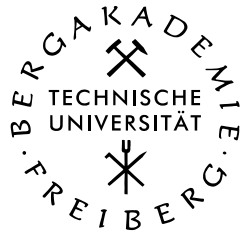
TECHNISCHE UNIVERSITÄT BERGAKADEMIE FREIBERG

The University of Resources. Since 1765.

Fakultät für Mathematik und Informatik (Fakultät 1)
Institut für Informatik
Lehrstuhl für Softwaretechnologie und Programmierungstechnik



Diploma Thesis

Web Applications Using the Google Web Toolkit

Michael von Wenckstern

Computational and Applied Mathematics
Specialization: Computer Science

Registration list: 50 455

April 30, 2013

Tutor/First Proofreader:
Prof. Dr.-Ing. habil. B. Steinbach
Room 224
Bernhard-von-Cotta-Straße 2
09596 Freiberg

Second Proofreader:
Dr.-Ing. G. Rudolf
Room 225
Bernhard-von-Cotta-Straße 2
09596 Freiberg

I Abstract

This diploma thesis describes how to create or convert traditional Java programs to desktop-like rich internet applications with the Google Web Toolkit.

The Google Web Toolkit is an open source development environment, which translates Java code to browser and device independent HTML and JavaScript.

Most of the GWT framework parts, including the Java to JavaScript compiler as well as important security issues of websites will be introduced.

The famous Agricola board game will be implemented in the Model-View-Presenter pattern to show that complex user interfaces can be created with the Google Web Toolkit.

The Google Web Toolkit framework will be compared with the JavaServer Faces one to find out which toolkit is the right one for the next web project.

Kurzfassung

Diese Diplomarbeit beschreibt die Erzeugung desktopähnlicher Anwendungen mit dem Google Web Toolkit und die Umwandlung klassischer Java-Programme in diese.

Das Google Web Toolkit ist eine Open-Source-Entwicklungsumgebung, die Java-Code in browserunabhängiges als auch in geräteübergreifendes HTML und JavaScript übersetzt.

Vorgestellt wird der Großteil des GWT Frameworks inklusive des Java zu JavaScript-Compilers sowie wichtige Sicherheitsaspekte von Internetseiten.

Um zu zeigen, dass auch komplizierte graphische Oberflächen mit dem Google Web Toolkit erzeugt werden können, wird das bekannte Brettspiel Agricola mittels Model-View-Presenter Designmuster implementiert.

Zur Ermittlung der richtigen Technologie für das nächste Webprojekt findet ein Vergleich zwischen dem Google Web Toolkit und JavaServer Faces statt.

II Contents

I Abstract ii
II Contents iii
III Acronyms and Glossary v
III.I Acronyms . . . v
III.II Glossary . . . vii
IV Credits x
1 Introduction 1
2 Basics 3
2.1 Development of the World Wide Web . . . 3
2.2 Hypertext Markup Language . . . 4
2.3 Cascading Style Sheets . . . 6
2.4 JavaScript . . . 7
2.5 Hypertext Markup Language Document Object Model . . . 8
2.6 Asynchronous JavaScript and XML . . . 11
3 GWT toolbox and compiler 12
3.1 GWT in action . . . 12
3.2 A short overview of the toolkit . . . 12
3.3 GWT compiler and JSNI . . . 12
3.3.1 Overview of GWT compiler and JSNI . . . 12
3.3.2 Deferred binding and bootstrapping process . . . 19
3.3.3 GWT compiler steps and optimizations . . . 24
3.4 Java Runtime Environment Emulation . . . 38
3.5 Widgets and Panels . . . 39
3.5.1 Overview of GWT Widgets . . . 39
3.5.2 Event handlers in GWT Widgets . . . 46
3.5.3 Manipulating browser's DOM with GWT DOM class . . . 49
3.5.4 GWT Designer and view optimization using UiBinder . . . 50
3.6 Remote Procedure Calls . . . 52
3.6.1 Comparison of Remote Procedure Calls with Remote Method Invocations . . . 52
3.6.2 GWT's RPC service and serializable whitelist . . . 54
3.7 History Management . . . 62
3.8 Client Bundle . . . 65
3.8.1 Using ImageResources in the ClientBundle interface . . . 65
3.8.2 Using CssResources in the ClientBundle interface . . . 68
4 Model-View-Presenter Architecture 75
4.1 Comparison of MVP and MVC . . . 75
4.2 GWT Model-View-Presenter pattern example: Agricola board game . . . 78
4.3 Extending the Agricola web application with mobile views . . . 88
4.4 Introducing activities in the Agricola Model-View-Presenter pattern enabling browser history . . . 94
5 Comparison of the two web frameworks: GWT and JSF 100
5.1 Definitions of comparison fields . . . 100
5.2 Comparison in category 1: Nearly completely static sites with a little bit of dynamic content, e.g. news update. . . . 100
5.3 Comparison in category 2: Doing a survey in both technologies. . . . 104
5.4 Comparison in category 3: Creating a forum to show data. . . . 106

5.5 Comparison in category 4: Writing a chat application. 109
5.6 Comparison in category 5: Writing the speed game Snake. 114
5.7 Summary 117
6 Security 119
6.1 Download Tomcat 119
6.2 Dynamic Web Application Project with GWT and Tomcat 119
6.3 Establish HTTPS connections in Tomcat 123
6.3.1 Create a pem certificate 123
6.3.2 Convert pem certificate into a key store object 124
6.3.3 Configure Tomcat's XML files to enable HTPPS 125
6.4 Establish a database connection in Tomcat 125
6.4.1 Create TomcatGWT user and schema, and add the table countries 125
6.4.2 Configure Tomcat's XML files to get access to the database connection 127
6.4.3 PreparedStatements avoid MySQL injections 130
6.5 Login mechanism in Tomcat 132
6.6 SafeHtml 135
7 Presenting a complex software application written in GWT 138
8 Conclusions 145
8.1 Summary 145
8.2 Future work 146
A Appendix A 1
A 1 Configure the Google Web Toolkit framework in Eclipse . . . . . A 1
A 1.1 Install the Java Developer Kit . . . . . A 1
A 1.2 Download Eclipse . . . . . A 1
A 1.3 Install the GWT plugin in Eclipse . . . . . A 1
A 1.4 Create first GWT Java Project . . . . . A 2
A 2 Figures . . . . . A 7
A 3 Listings . . . . . A 50
A 3.1 Source code of the Agricola board game . . . . . A 113
A 3.2 Source code of GWT and JSF comparison . . . . . A 269
A 4 Tables . . . . . A 310
R Lists and References R 1
R 1 Lists . . . . . R 1
R 1.1 List of Tables . . . . . R 1
R 1.2 List of Figures . . . . . R 2
R 1.3 List of Listings . . . . . R 5
R 2 References . . . . . R 12
R 2.1 Books . . . . . R 12
R 2.2 Online resources . . . . . R 12

III Acronyms and Glossary

III.I Acronyms

AJAX Asynchronous JavaScript and XML.

API Application Programming Interface.

ARPANET Advanced Research Projects Agency Network.

ASP Microsoft's Active Server Pages.

AST Abstract Syntax Tree.

CERN Conseil Européen pour la Recherche Nucléaire.

CGI Common Gateway Interface.

CM Communicating Module.

CSS Cascading Style Sheets.

DOM Document Object Model.

Eclipse RAP Eclipse's Rich Ajax Platform.

GPU Graphics processing unit.

GUI Graphical User Interface.

GWT Google Web Toolkit.

HTML HyperText Markup Language.

HTML DOM HyperText Markup Language (HTML) Document Object Model.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IE Internet Explorer.

JDT Java development tools.

JRE Java Runtime Environment.

JREE Java Runtime Environment Emulation.

JSF JavaServer Faces.

JSNI JavaScript Native Interface.

JSP JavaServer Pages.

MVC Model-View-Controller.

MVP Model-View-Presenter.

PHP Hypertext Preprocessor.

POJO Plain Old Java Object.

RAM Random-Access Memory.

RFC Request for Comments.

RIA Rich Internet Application.

RMI Remote Method Invocation.

RPC Remote Procedure Call.

SDK Software Development Kit.

SQL Structured Query Language.

SSI Server Side Includes.

SSL Secure Sockets Layer.

SVG Scalable Vector Graphics.

SWT Standard Widget Toolkit.

TLS Transport Layer Security.

UML Unified Modeling Language.

URL Uniform Resource Locator.

XML Extensible Markup Language.

III.II Glossary

Adobe Flash is a runtime environment allowing web browsers streaming videos and drawing shapes. Many internet games and video players are built upon Flash.

Android "is an open-source software stack for mobile phones and other devices." [Goo02a].

Application Programming Interface describes how software components should interact with each other.

BlackBerry is a mobile operating system developed by RIM (Research In Motion).

Cascading Style Sheets are used to describe the layout of a webpage.

Common Gateway Interface "The earliest Hypertext Transfer Protocol (HTTP) servers did not include any build-in mechanism for generating response dynamically. Instead, interfaces were provided for calling other programs to translate requests into run-time content. The first standard for dynamic web content was based on Common Gateway Interface (CGI), which specified a mechanism for web servers to pass request information to external programs, which were then run by the web server to generate responses at run-time." [FK00, p. 3].

Conseil Européen pour la Recherche Nucléaire is a nuclear research center in Switzerland, Europe.

Cookie "A small piece of data sent by websites and stored in a user's web browser to remember the website's state or past user activity." [GHV].

Drag and Drop is the action of selecting one or more (virtual) objects and move them from one place to another with the mouse.

ECMAScript is the standardized client-side scripting language in the web. 'JavaScript, JScript and ActionScript are well-known dialects.' [Wik03].

Extensible Markup Language "XML was designed to transport and store data. HTML was designed to display data." [Reff] More information are available at [Wor11].

Flex Adobe "Flex is a powerful, open source application framework that allows you to easily build mobile applications for iOS, Android, and BlackBerry Tablet OS devices, as well as traditional applications for browser and desktop using the same programming model, tool, and codebase." [Ado].

Graphics processing unit A graphic card has more, but less powerful, GPUs than a processor has CPUs, today. Many parallel algorithms like matrix multiplications are running faster using thousand GPUs instead of some CPUs.

GWT-Ext is a "Google Web Toolkit (GWT) wrapper around an earlier, 2.0.2 version of Ext JS. Being based on Ext JS, it has a very similar look and feel to" GXT. "Developing with GWT-Ext is more difficult [...], because the library is a wrapper around JavaScript and the Java debugger cannot help when there is a problem in the JavaScript code. Manual debugging is required. [...] Active development came to an end" in 2008. [Vau10, p. 9].

GXT is the shortform of Ext-GWT, which stands for Extended GWT; we are using GXT instead of Ext-GWT to avoid confusion with GWT-Ext (it is also a GWT extension framework).

Hypertext Preprocessor is todays most common serverside scripting language.

iPad is Apple's tablet computer; it is like an iPhone, but it has a bigger screen and you can not call people.

iPhone is a smartphone with a touchscreen which is designed by Apple, it is running with Apple's iOS mobile operating system.

Java development tools The Eclipse **Java development tools** "project provides the tool plug-ins that implement a Java IDE supporting the development of any Java application". [tea].

JavaServer Faces is Oracle's replacement for JavaServer Pages (JSP), for more information see section 5.

JUnit "JUnit is a simple framework to write repeatable tests" in Java.

Microsoft's Active Server Pages "ASP support multiple scripting languages, including PerlScript, JScript and VBScript. PerlScript is based on Perl and JScript based on JavaScript. But the default scripting language is VBScript, a subset of Microsoft's popular Visual Basic programming language." [FK00, p. 6].

Model-View-Controller The model consists application data, view displays the model data to the user, controller receives the interaction events of the user and changes the model and view.

MySQL "MySQL is named after co-founder Monty Widenius's daughter, My." [Orab] SQL stands for Structured Query Language.

Plain Old Java Object The given object is an ordinary (and not special) Java object.

RSA RSA, which is the abbreviation of the designers' names Ron Rivest, Adi Shamir and Leonard Adleman, is a public-key cryptography algorithm.

Scalable Vector Graphics "is a markup language for describing two-dimensional graphics applications and images". [Wor06].

Secure Sockets Layer are used to protect the traffic between server and web browser from a third party.

Server Side Includes The syntax is similar to Hypertext Preprocessor (PHP) (more information under http://www.apacheweek.com/features/ssi).

Servlet 'Java Servlets are similar to CGI, you have a HTTP request as an input from the browser, and a Java program will produce a dynamic output.' [FK00, p. 7].

Smart GWT Smart GWT is a "framework that wraps the Smart Client JavaScript library in a similar way that GWT-Ext wraps Ext JS. Smart GWT has the advantage that it is still being actively developed." [Vau10, p. 9].

Standard Widget Toolkit is Eclipse's graphical user interface for java programming, a short comparison between Swing and Standard Widget Toolkit (SWT) can be found at [Son].

Struts The main idea of Apache Struts is to write web applications with the Model-View-Controller (MVC) pattern. It is mostly based on Servlets and JSP. [The08a].

Swing is the primary Java Application Programming Interface (API) for graphical user interfaces in desktop applications. It is part of the Java Runtime Environment.

Transport Layer Security The Transport Layer Security is the same like Secure Sockets Layer (SSL), but newer and more secure.

Uniform Resource Locator The syntax of Uniform Resource Locator is is: scheme://domain:port/path?query_string#fragment_id.

Vaadin Vaadin "is a server-side framework that uses a set of precompiled GWT components. [...] In Vaadin the browser client is just a dumb view of the server components and any user interaction is sent to the server for processing. [...] The main disadvantage of Vaadin is the dependency on the server. GWT or GXT's JavaScript can run in a browser without needing to communicate with a server. This is not possible in Vaadin." [Vau10, p. 9].

W3C The World Wide Web Consortium develops the new web standards.

Web Workers "Allow scripts to run in the background to handle computationally intensive tasks, without blocking the UI or other scripts that handle user interactions." [GHV].

WebGL "makes it possible to display amazing realtime 3D graphics in your browser but what many people don't know is that WebGL is actually a 2D API, not a 3D API." [Gre02].

WebOnSwing Swing based web development "framework that allows you to create web applications in the same way you develop a desktop one". [XOO09] You can use the MVC pattern and the Swingdesigner to develop your application.

Wicket "With proper mark-up/logic separation, a Plain Old Java Object (POJO) data model, and a refreshing lack of Extensible Markup Language (XML), Apache Wicket makes developing web-apps simple and enjoyable again." [The08b].

IV Credits

I would like to thank my parents Bettina and Bodo who supported me morally and financially during all nine semesters studying Applied Mathematics. Without them I would not have been able to finish it in such a short time with such a good result.

I owe special thanks to my supervisor Prof. Dr. Bernd Steinbach, who advised and supported me throughout the realization of this diploma thesis, and Dr. Peter Michel. Both of whom were the main reason for the topic of this work. I also want to thank them for the pleasant, more than three years long employment contract and for the trust and freedom they gave me to convert the research Java program into a web application with the framework of my choice.

Moreover, thanks go to my girlfriend Maren Lepke, who supported and helped me in the last months. She and Amanda Richardson improved this text in several proof-reading sessions.

1 Introduction

The research project "Trip-Matrix-Composite" of the TU Freiberg develops new high-performance steel composites. A user-friendly client program should save the research results of different institutes in a central database. The first software solution was implemented as Java Swing application. One drawback of desktop programs is that the user has to download and replace the application for each new feature or fixed bug. Web applications like Google Maps do not have this disadvantage, because the webserver always sends the newest version to the web browser. The main motivation to convert the existing Java application into a website was that some institute computers did not allow the installation of the Java Runtime Environment and so the database software could not run.

The Google Web Toolkit framework was the first choice as it has a Java to JavaScript compiler. This allows reusing Eclipse as integrated development environment and some of the program's Java code.

The aim of this diploma thesis is to answer the crucial question, whether it is possible to create a web application in an efficient way with the Google Web Toolkit, which has the same behavior as a normal Java desktop program.

In order to give a detailed answer to the above, the following issues must be discussed:

- Is it possible to extend the Google Web Toolkit (GWT) framework with JavaScript code in the same way a Java application can be extended with C or assembler code using the Java Native Interface? This is of great importance, because no framework is complete and other libraries have to be imported, which are often written in another language.
- Does GWT support techniques equivalent to Java reflection? This is necessary to control the runtime behavior for different operating systems or web browsers.
- Does GWT provide analog classes to the most important Java ones like `ArrayList`, `HashMap`, and so on? This would simplify the change from the Swing framework to the GWT one, because there is no need to learn many new classes.
- Two very important questions are: Are there Graphical User Interface (GUI) components in the GWT equivalent to Swing ones, and is the user event handling as powerful as in Swing? Answering these questions with yes, means that the web layout can be similar to the desktop one, and that both technologies have the same user experience.
- How can the GWT web application communicate with other Java and non-Java applications? And is it more difficult in GWT for the developer to exchange messages with others? This is significant so that distributed systems can be converted into a web application.
- Java applications can write the actual program state to disk. How does GWT save the web application state so that the user can restore it?
- Is GWT suitable for user interface design patterns? This is required when developing large scale applications containing several hundred thousand lines of source code.
- Java applications' client-server communication can be secured with the iSaSiLk [sys] library. Is there a way to protect the communication between web applications and any server using the GWT framework?

Compared to JavaServer Faces 2.0, no standard for GWT exists and so the following question arises: Can the JavaServer Faces (JSF) framework create the same desktop-like web applications?

The following outline gives answers to the above questions:

Section 2 describes the basic web technologies. The historical development of the World Wide Web introduces different technologies. Then the five most important web standards will be

introduced: HyperText Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript, HTML Document Object Model (HTML DOM) and Asynchronous JavaScript and XML (AJAX).

The third section is the biggest in this paper. It starts by showing existing web applications and games created with the Google Web Toolkit; most of them let the user believe that they are desktop applications running in the web browser. The rest of this section will take a very detailed look at the GWT toolbox while trying to answer most of the sub questions. This way an opinion can be formed as to how difficult it is for Java developers creating web applications with the GWT.

The next section explains how to structure large GWT applications. The famous Agricola board game is the example implementation for the Model-View-Presenter pattern. One advantage of this pattern is the opportunity to extend the desktop view with an optimized tablet one. Answers are given to the two issues, whether GWT is suitable for complex applications and whether the user can load saved game states into the web application.

The following section compares the Google Web Toolkit with JavaServer Faces. The comparison of the two web frameworks is done in five categories representing today's major website kinds. This helps to figure out, if GWT is the right toolkit for different web projects.

Section six is about security in web applications. The aim is to check, if applications are as secure as normal Java desktop programs.

Afterwards, the result transferring the database desktop program into a rich internet application will be shown. This way a decision can be made whether it was possible to write a web application, which has the same behavior as the already existing database program.

The last section summarizes the result of the previous pages, and answers all introduced sub questions in a detailed way. A list with advantages and disadvantages of web applications helps to decide which desktop program types can be converted into web applications. This section finishes with future work.

2 Basics

2.1 Development of the World Wide Web

This section starts by explaining the history of the World Wide Web. A very nice illustration about the evolution of the web can be found at [GHV]. The timeline shows that many new technologies have been invented since 2009. At the end of this section it can be noticed that HTML 5, supported by all major web browsers since 2012, is an important milestone for web applications.

'Originally the Internet was created in 1969 by Advanced Research Projects Agency Network (ARPANET) to connect the Universities of the Air force. The main idea of ARPANET was to create a net with shared nodes. The resolution was to use a decentralized structure and so the network could survive a military attack. The Web itself is younger than the internet, the basics started at Conseil Européen pour la Recherche Nucléaire (CERN) about 1990. Tim Berners-Lee developed the first webserver and defined the first fundamentals of HTML. His hypertext definition means connecting documents with each other in a network-like structure. The first website is still available under [Tim12]. This site contains only text and hyperlinks (see figure A13); the source code (see listing A3) is interesting, too. ' [HWM06, pp. 32–33]

FileUpload (HTML 2.0), Cookies, Secure Sockets Layer (SSL) and CSS were introduced ca. five years later. This was the beginning of dynamic websites and languages like Hypertext Preprocessor (PHP) in 1995 or JavaServer Pages (JSP) in 1999. Rather nice definitions of static and dynamic web content can be found at [FK00, pp. 2–3].

A static website is requested from the web browser after the user has entered its address. The web server looks for the corresponding file and if it exists, the file content will be returned. Each document link in the HTML text will result in an extra file request by the web browser. "Documents that are requested never change regardless of who requested them, when they were requested, or which (if any) additional parameters are within the request. New versions of the documents might be placed on the server, but at any time, every request for those documents returns exactly the same results." [HWM06, pp. 32–33] The main characteristic of static web pages is that the server only detects the local files and uploads their contents unmodified.

But nearly all data is dynamic today. Examples are the current news pages or the own shopping card site at Amazon. "Dynamic web content, then, requires that the web server does some additional processing of the corresponding request in order to generate a customized response." [FK00, p. 3] Additional Uniform Resource Locator (URL) parameters are necessary to identify customized page requests. The timeline of [GHV] shows many new technologies about 1995. Some of them have been described above. From now on, not only static websites but also dynamic ones exist. This was the beginning of "Web 2.0" [Alb07].

Many new standards like Drag and Drop, Web Workers, WebGL, and File System Application Programming Interface (API) have been introduced since 2009. This is the turning point from traditional dynamic web pages to rich internet applications. For example Google Documents [Goob] is a web application having nearly the same behavior as Microsoft Office with Word, Excel and PowerPoint. The Aviary [Avia] web application, shown in figure A21, is a replacement for Microsoft Paint.

The beginning of the traditional dynamic web site was the birth of technologies and languages like Common Gateway Interface (CGI), Servlets, Server Side Includes (SSI), Microsoft's Active Server Pages (ASP), PHP, JSP and many more. Eclipse's Rich Ajax Platform (Eclipse RAP), GWT, Wicket, JSF, Flex, Struts, WebOnSwing (more can be found at [AW11, p. 43]) are now available to create rich internet applications.

Despite not needing to write any HTML and JavaScript code in GWT, it is useful to learn the basics of web languages and technologies. This allows to understand the GWT compiled output files and to extend the functionality of some widgets.

2.2 Hypertext Markup Language

The purpose of the Hypertext Markup Language is to structure documents and to create links between them. The layout and explicit view of a web page is done by the layout language Cascading Style Sheets and not by HTML. HTML can be written as plain text, and it does not have been compiled to a binary representation. This has the advantage that everyone can directly read the structure of a web page. Figure A14 shows the HTML code of the website tu-freiberg.de. The most important parts of this markup language are the HTML tags describing the structure. There are two types of HTML elements: the paired and unpaired ones. The paired ones start with <tag> and end with </tag>. As the name already says the unpaired elements have no corresponding ending tag, their only content is <tag>. Each HTML element can have attributes describing extra information on it. The attributes are always listed in the start tag and they appear as name/value pairs. The general start tag structure is <tag attributeName1=attributeValue1 ... attributeNameN=attributeValueN>. The HTML tags and attribute names are case insensitive, but some of the values are case sensitive, e.g. the value of the title attribute. It is possible to nest tags between other open and end tags. This would look like the following <tag1 attributeName11=attributeValue11 ... attributeName1N=attributeValue1N> <tag2 attributeName21=attributeValue21 ... attributeName2N=attributeValue2N> </tag2> </tag1>. Please notice that the closing sequence of the end tags is the opposite order of the opening sequence of the start tags. Some parent attributes are inherited to their child elements; e.g. if tag1 contains the attribute pair color="blue" and tag2 does not contain the attribute name color, then it inherits it from tag1. In order to have a separation of concerns, neither styling tags like <font> nor styling attribute names like color should be used in the HTML code. The HTML style should always be declared in CSS. Table 1 shows the most important HTML tags.

Table 1: Most important HTML tags

HTML tag	**Description**	**Example**
html	start tag represents the beginning of the HTML language and the end tag finishes the markup language	<html lang="de">...</html> means the entire content of the page is given in the German language.
head	contains meta information about the website	<head>...</head>
meta	information for the web browser and the web server	<meta charset="UTF-8"> means that the web browser should use the Unicode character to display the content. This allows the usage of special signs like ä,ö directly instead of writing ä and ö.
body	contains the content which will be displayed in the web browser, it can be divided into the following parts <header>, <nav>, <section>, <article>, <aside> and <footer>	<body> <header> HTML information </header> <nav> Tags:ab</nav> <section> Information to tag ... </section> <footer>(c) Michael von Wenckstern</footer> </body>

h1 ... h6	headings of pages, h1 is the main heading, and h2 to h6 are subheadings	Operators in Java Unitary operators Not operator Binary operators And operator Or operator
a	for internal and external links	 search engine is external link Back to main page is internal link to a different page jump back to top of page is internal link to a specific point on the same page Heading sets an anchor on this page
li,ol,ul	list item in ordered list or unordered one	- programming languages - C - C++ - Java - PHP - search engines - Google - Bing generates 1. programming languages ◦ C ◦ C++ ◦ Java ◦ PHP 2. search engines 1. Google 2. Bing
table, thead, tfoot, tbody, th, tr, td	defines a table th is an abbreviaton for table header data tr for table row td for table data	<table border="1"> <thead> <tr> <th>Header 1</th> <th>Header 2</th> </tr> </thead> <tfoot> <tr> <td>Remark1</td> <td>Remark2</td> </tr> </tfoot> <tbody> <tr> <td>(1,1)</td> <td>(1,2)</td> </tr> <tr> <td>(2,1)</td> <td>(2,2)</td> </tr> </tbody> </table> generates Header 1 Header 2 (1,1) (1,2) (2,1) (2,2) Remark1 Remark2
div, span	describes an area, div tags can contain other div and span tags, span tags are inline areas and so these cannot contain other elements	Main area Area 1 Area 2 Area 2.1 Area 2.2 are often used with class or id attributes so that they can be styled later with CSS.
p	paragraph	This is the first paragraph. And this is the second one.

br	forces line break	Line 1 Line 2
input	for user input commands, the following types are allowed: • button • checkbox • color • date • datetime • datetime-local • email • file • hidden • image • month • number • password • radio • range • reset • search • submit • tel • text • time • url • week	<form name="input" action="html_form_action.asp" method="get"> Username: <input type="text" name="user" value="Michael"> Password: <input type="password" name="pwd" value="123"> Sex: <input type="radio" name="sex" value="male" checked>Male <input type="radio" name="sex" value="female">Female <input type="checkbox" name="lang" value="En" checked>English <input type="checkbox" name="lang" value="De" checked>German <input type="checkbox" name="lang" value="Fr">French <input type="button" name="help" value="Help" onclick="alert('Please insert your personal information here.');"> <input type="reset" value="Reset"> <input type="submit" value="Submit"> </form> generates Username: Michael Password: ••• Sex: ◉ Male ○ Female ☐ English ☐ German ☐ French Help Reset Submit

2.3 Cascading Style Sheets

Cascading Style Sheets are used to describe the specific layout of websites. It can be used to customize the HTML code. There are four ways to define CSS styles:

1. tag level: HTML-tag[attribute condition]: { ... }
2. class level: .class-Name: { ... }
3. id level: #id-Value: { ... }
4. inline: style=" ... "

All CSS properties can be found at [Refa]. Listing 1 shows an example of how to style HTML code with CSS. Figure 1 displays the result.

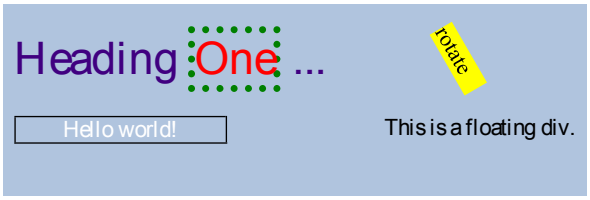


Figure 1: CSS example

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <style>
5  body { background-color:#b0c4de; }
6  h1 { color: rgb(64,0,128); }
7  h1 .redBox {color: red; border: 5px dotted green;}
8  #d1 {color: white; float:left; border: 1px black solid; width: 150
       px; text-align: Center;}
9  #d2 {float:right}
10 </style>
11 </head>
12 <body>
13 <h1>Heading <span class="redBox">One</span> ...</h1>
14 <p><div id="d1">Hello world!</div><div id="d2">This is a floating
       div.</div></p>
15 <div style="transform:rotate(60deg);background-color:yellow;width
       :50px;position:absolute;left:300px;top:30px;">rotate</div>
16 </body>
17 </html>
```

Listing 1: CSS example

2.4 JavaScript

'ECMAScript is the standardized scripting language on the web. It is more commonly known as JavaScript.' [Ecm03] Erik Meijer gives an appropriate definition of JavaScript: "JavaScript is an assembly language. The JavaScript + HTML generate is like a .NET assembly. The browser can execute it, but no human should really care what's there." [Han]. In the same way a good C developer knows some background information about the lower-leveled machine language, a GWT developer can write better and more optimized code having knowledge about JavaScript. JavaScript is a dynamic, weakly typed, imperative, multi-paradigm and functional language. The syntax is C like. Different browsers implement different JavaScript dialects, e.g. Internet Explorer 9 implements Jscript 9, Google Chrome uses the V8 JavaScript Engine, and Mozilla Firefox uses the SpiderMonkey engine. One problem is that every JavaScript engine interprets the code differently. The conformity of the browser scripting engines to ECMAScript 5.1 can be tested at [Ecm03]. Internet Explorer 10 failed 8, Chrome 26 failed 15 and Firefox 20 failed 193 from 11573 tests (see figures A15, A16 and A17). Assuming that each of these browsers failed different tests, then the JavaScript engine difference between these three programs is 1.87%.

[Refe] is an introduction about JavaScript. Listing 2 and figure 2 show a simple calculator example written in JavaScript. The next subsection explains how to manipulate HTML elements in JavaScript to create dynamic user interfaces.

Figure 2: Simple JavaScript calculator example

```
<!DOCTYPE html>
<html>

<head><script>
function calc() {
  v1 = document.getElementById("operand1").value;
  v2 = document.getElementById("operand2").value;
  res = document.getElementById("result");
  switch(document.getElementById("operator").value) {
    case "m": res.value = v1 - v2; break;
    case "t": res.value = v1 * v2; break;
    case "d": res.value = v1 / v2; break;
    default: res.value = parseInt(v1) + parseInt(v2); break;
  }
}
</script></head>

<body>
<input type="number" id="operand1" value="2">

<select id="operator">
<option value="p" selected>+</option>
<option value="m">-</option>
<option value="t">*</option>
<option value="d">/</option>
</select>

<input type="number" id="operand2" value="1">
<input type="button" value="=" onclick="javascript:calc();">
<input type="number" id="result" value="3" disabled>
</body>
</html>
```

Listing 2: Simple JavaScript calculator example

```
<!DOCTYPE html>
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href="">My link</a>
    <h1>My header</h1>
  </body>
</html>
```

Listing 3: Simple HTML code for Document Object Model (DOM) model

2.5 Hypertext Markup Language Document Object Model

The browser parses all HTML files and creates a Document Object Model before it displays the website's content. The advantage of this intermediate step is that JavaScript can modify the

created DOM. This allows creating real dynamic web pages by adding, changing or removing elements of the Document Object Model. To get a better understanding of the HTML DOM structure, figure 3 shows the DOM model of the HTML code displayed in listing 3.

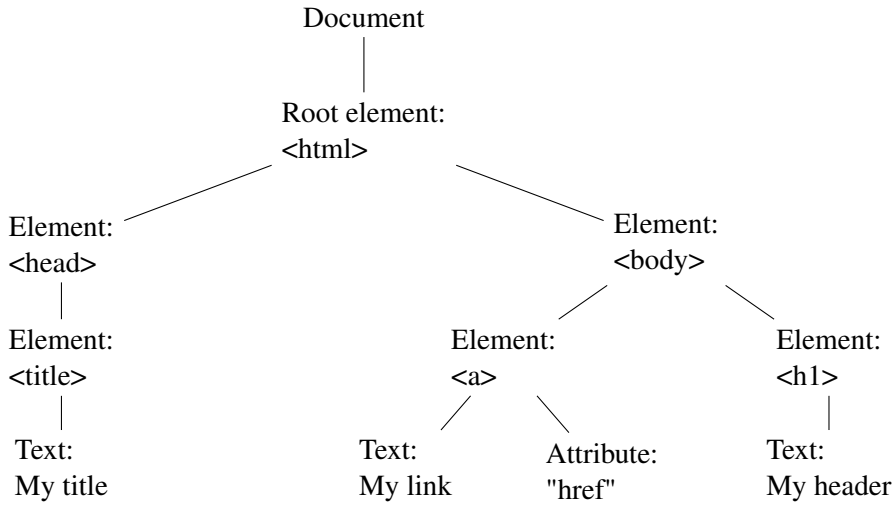


Figure 3: HTML DOM of listing 3. Copied from [Refd].

There are three ways to access the elements in the Direct Object Model:

- getting all elements with a given tag name, like h1
- getting all elements with a given class name
- getting the only element with a given id

Node hierarchy

'According to the official W3C standard, everything in an HTML document is a node:

- the complete HTML document is a document node
- every HTML element like is an element node
- the text inside HTML elements are text nodes which can be accessed by the node's innerHTML property
- every HTML attribute is an attribute node
- even comments are comment nodes in the DOM model

The nodes in the DOM tree are related to each other:

- the top node is called root element which is always the <html> tag
- every node except the root node has exactly one parent element
- a node can have no, one, or many children
- siblings (sisters or brothers) are nodes with the same parent node

In figure 3 <html> is the root node. It has the <head> node as first child and the <body> node as last child. The parent node of <head> is the <html> node. The child nodes <head> and <body> are siblings to each other.' [Refb]

Example manipulating the HTML DOM

Figure A18 and listing A4 show the complete source code and the corresponding screenshot of this example. The most important parts of the source code are explained below.

Line 21 defines an array containing the shape properties, which can be changed by the user later.

```
21 var shapeArray = [ ['Rectangle','rect',['x','y','width','height
     ']], ['Circle','circle',['cx','cy','r']],['Ellipse','ellipse
     ',['cx','cy','rx','ry']],['Line','line',['x1','y1','x2','y2
     ']],['Text','text',['x','y','html-text']] ];
```

Listing 4: HTML code of DrawDomExample.html (line 21).

Lines 30 until 41 create dynamically the radio buttons by manipulating the browsers DOM hierarchy.

```
30   var index;
31   for(index = 0; index < shapeArray.length; ++index) {
32     var radioInput = document.createElement('input');
33     radioInput.setAttribute('type', 'radio');
34     radioInput.setAttribute('name', 'shapes');
35     radioInput.setAttribute('onclick', 'showProps();');
36     radioInput.setAttribute('id', shapeArray[index][1]);
37     s.insertBefore(radioInput,b);
38     s.insertBefore(document.createTextNode(shapeArray[index][0]),b
         );
39     s.insertBefore(document.createElement('br'),b);
40   }
41   document.getElementById(shapeArray[0][1]).checked = true;
```

Listing 5: HTML code of DrawDomExample.html (lines 30-41).

A new HTML input element is created in line 32. The next lines 33 to 36 add some attributes to this element. The generated HTML code for index=0 is <input type='radio' name='shapes' onclick='showProps();' id='rect'>. The radio button will be added to the div element having the id "shapes" in line 37. At this point the radio button is visible in the web browser, because the created input node is available from the root node. The next two lines insert text and a line break into the div node. Before the loop is executed, the HTML code is Shapes:
 <input type="button" id="add" value="add" onclick="add()">
 . After two loop iterations with index=0 and index=1, the generated HTML code looks like Shapes:
 <input name="shapes" id="rect" onclick="showProps();" type="radio">Rectangle
<input name="shapes" id="circle" onclick="showProps();" type="radio">Circle
 <input id="add" onclick= "add()" type="button" value="add">
 .

The lines between 89 and 121 create a new Scalable Vector Graphics (SVG) shape, set the properties of the new shape and add it to the SVG draw container. The most important source code lines are:

```
90   var svg = document.getElementById('svg');
95       shape = document.createElementNS('http://www.w3.org/2000/svg
           ', shapeArray[index][1]);
109          shape.appendChild(document.createTextNode(n.value));
111          shape.setAttribute(n.id, n.value);
121  svg.appendChild(shape);
```

Listing 6: HTML code of DrawDomExample.html (lines 90, 95, 109, 111, 121).

The function changeProps modifies the HTML DOM model by iterating over all attributes of the selected shape element and changing the attribute value to the new one in the input field. The id of the input field is the same as the name of the changed attribute name. The interesting code lines are:

```
147   var a = shape.attributes;
148   for(var i=0; i<a.length; ++i) {
149     if(propAttr.indexOf(a[i].nodeName) != -1) {
150       a[i].nodeValue = document.getElementById(a[i].nodeName).
              value;
151     }
155   }
```

Listing 7: HTML code of DrawDomExample.html. (lines 147-151)

2.6 Asynchronous JavaScript and XML

Asynchronous JavaScript and Extensible Markup Language (XML) are used to load server data in the background. This way the web browser will not block until the data is received. The advantage of AJAX in conjunction with HTML DOM is the ability to update a part of the website without reloading it completely again. Imagine the website is a large HTML page with many pictures and the latest news should be displayed in the left part of the web page. AJAX allows loading the news information in background; after the browser has received the data, the website's news part can be showed using HTML DOM. This way there is no need to reload the entire page, which may take several seconds to show up. Listing A5, figures A19 and A20 display the complete example. The AJAX code has been the same for all browser versions since 2007. The previous browsers Internet Explorer (IE) 5 and IE 6 had to create the AJAX object with xmlhttp=new ActiveXObject("Microsoft.XMLHTTP").

The timer will continue counting, when the website receives the latest news, at the bottom of the page. This proves that the page is not completely reloaded. Line 6 creates the AJAX request object. The next line assigns a function, which should be called after the request finishes, to the AJAX object. Line 12 sets the URL containing the actual data to the object. The last line of listing 9 starts the request by calling the send method. As earlier mentioned the data is shown to the user by manipulating the DOM model in line 9.

3 GWT toolbox and compiler

3.1 GWT in action

This section begins by showing web applications and games created with the Google Web Toolkit. [Goo11] displays the GWT showcase. The left frame contains several groups with the names of the basic GWT components. The behavior of the selected component can be tested in the right frame (see figure A22). The GWT user interface widgets have similar names and behaviors as the components used in Swing or Standard Widget Toolkit (SWT). However, the GWT library does not include advanced panel components with different layouts like GXT's BorderLayoutContainer displayed in figure A24.

This is the reason why it is useful to extend the Google Web Toolkit with other libraries like GXT, GWT-Ext, Smart GWT or Vaadin. The unique characteristic of GXT is that it extends GWT using Java code. All the other above listed libraries only wrap their JavaScript classes into a Java one. The main disadvantage of wrapper classes is that they cannot be debugged in Eclipse, and this makes it more difficult to extend any widget of these libraries.

The GXT showcase is available at [Sen08b]. Many GXT widgets have a desktop-like layout and behavior. The advanced toolbar component shown in figure A23, is very similar to Microsoft's ribbon bar. The GXT library also contains many layout panels such as AccordianLayout, BorderLayout, CardLayout, HorizontalLayout, and VerticalLayout. This means GWT applications can use the same layout patterns as desktop ones.

Both showcases do not only display the components, they also show the source code of the selected example next to it. This makes both of them perfect learning places for new components. The GXT web desktop shown in figure A25, demonstrates the "power" of GXT. The web application can be tested at [Sen08a].

Figure A26 displays the GWT Quake 2 project, which can be found at [Ste]. The ego shooter online game uses WebGL to render the graphics, WebSocket to communicate with the other players, and WebStorage to save the actual games state. PlayN is a "GWT/Java based" [Plab] project "writing games that compile to" [Plac] five different platforms. The popular online game Angry Birds [Plaa] is created with the PlayN framework.

The mgwt [Dan06] toolkit creates websites which look and behave like native phone or tablet apps on iPhone, iPad, Android, or BlackBerry. Figure A27 shows a showcase [Dan] screenshot of mgwt. [api] demonstrates how to include Google Maps in GWT (see figure A28). Further information and the official site can be found at [Kei09] and [Bra].

The aim of this subsection was to give an overview of web applications. This may help to decide which framework should be used to convert a desktop program into a website. Another goal of demonstrating the different web pages is to motivate to learn a new API and how the GWT compiler works.

3.2 A short overview of the toolkit

GWT provides several tools moving desktop applications into the browser. Table 2 shows the most important parts of the toolbox.

3.3 GWT compiler and JSNI

3.3.1 Overview of GWT compiler and JSNI

The first part of this section explains how the compiler works in general and where the compiler gets the corresponding JavaScript code from the Java one. This knowledge also allows extending the GWT toolkit with third-party JavaScript libraries.

Table 2: Overview of GWT toolbox

Java to JavaScript compiler	GWT compiler converts Java source code (no byte code) into optimized browser and language specific JavaScript code.
JavaScript Native Interface (JSNI)	'JSNI allows direct integration of JavaScript code into Java source code and vice versa.' [Goo06b]
Java Runtime Environment (JRE) Emulation (JREE)	It provides a small (most commonly used) part of the real JRE. Only Java classes of the JREE can be used, because these can be compiled to JavaScript.
GWT API	
Widgets and Panels	These are the components displayed in the GWT showcase in section 3.1.
Internationalization (I18N)	Techniques to provide multiple language support in a static and dynamic way. This paper will not cover I18N, for more information see [HT07, pp. 47–49] or [Goo06g]
Remote Procedure calls (RPC)	Mechanism to exchange information between JavaScript (browser side) and Java (server side), see section 3.6.
XML Parser	It converts XML code (e.g. from Hypertext Transfer Protocol (HTTP)-Requests) into an object model (DOM tree). GWT will call the browser's API to do the conversion as native code execution for speed improvement.
History Management	Mechanism allowing the user to take advantage of the browser's history; e.g. bookmarking a website's state, using back and forward button.
Client Bundle	Mechanism which enables caching of resources to improve the loading time and increase the application speed.
JUnit Integration	Allows writing automated tests for GWT code. This theme is not part of this diploma thesis, further information is available under [HT07, pp. 560-587]

This section assumes that both Eclipse and GWT are downloaded, installed and configured as shown in appendix A 1.

One way to understand the compiler basics is to compare some parts of the Java code with the generated JavaScript one in an example. After creating a new GWT Project[1] and a GWT module[2], Eclipse generates an example code as shown in listing 8. This code creates a web page containing only one button with the text "Click me!". After the button is clicked, a modal window with the message "Hello, GWT World!" pops up as shown in figure A29. The function `onModuleLoad` is equal to the standard Java function `main`, it is the starting point of the web application. Line 32 creates a normal button. It is added to the website in the line below. The `rootPanel` variable can be compared with the `contentPane` one of the main frame object in Swing. Line 34 sets the text of the button to "Click me" and in the next line a click handler is added, which shows the popup-window with the message "Hello, GWT World!" (line 37). After compiling this project[3], the file ImageViewer.html and the folder de.tu_freiberg.informatik.vonwenckstern.ImageViewer can be found in the war directory. The

[1] Files -> New -> Project ... -> WindowBuilder -> GWT Designer -> Model -> GWT Java Project; project name: DA_GWTCompile

[2] Module name: ImageViewer, Package name: de.tu_freiberg.informatik.vonwenckstern

[3] Google icon in toolbar -> GWT compile project ... -> Output style: Detailed

```
29   public void onModuleLoad() {
30     RootPanel rootPanel = RootPanel.get();
31
32     clickMeButton = new Button();
33     rootPanel.add(clickMeButton);
34     clickMeButton.setText("Click␣me!");
35     clickMeButton.addClickHandler(new ClickHandler(){
36       public void onClick(ClickEvent event) {
37         Window.alert("Hello,␣GWT␣World!");
38       }
39     });
40   }
```

Listing 8: Excerpt from auto generated example source code by Eclipse; complete code (see listing A6)

file ImageViewer.html contains all the static content for the web application and a reference to the dynamic content[4] loader.

Opening this JavaScript file in a text editor shows that the web content is loaded dynamically, depending on which browser version is used (see listings A6 and A8)[5]. This is necessary because every browser has a different layout and JavaScript engine[6]. This is one big advantage of the GWT framework, because the Java code only has to be written once and the compiler will handle the browser-specific code.

Generally GWT compiles by substituting the Java code with the given JavaScript one using JavaScript Native Interface (JSNI). Listing 9 displays the JSNI implementation of window.alert, which can be opened by pressing the control key and clicking with the mouse on the alert function invocation in Eclipse.

```
610 public static native void alert(String msg) /*-{
611     $wnd.alert(msg);
612   }-*/;
```

Listing 9: JSNI implementation of window.alert

This means the line Window.alert("Hello, GWTWorld!") will be translated to $wnd.alert ('Hello, GWT World!')[7] which can be found e.g. in Opera's specific website[8] at line 341.

The next paragraphs give more details about JavaScript Native Interface. JSNI is the bridge between Java and JavaScript. Because only the web browser is able to execute JavaScript code, JSNI cannot be used on the server-side part. This passage shows how to pass Java objects to JavaScript, load external JavaScript libraries, wrap JavaScript code as Java classes, and execute Java code from JavaScript. This section intends to give an understanding of the lowest structure of GWT components. It also explains how different libraries like the Smart GWT one work. Another aim of this subsection is to describe JSNI in such a way that Google's implementations of different functions are readable. This removes the mystery factor from the compilation process.

[4] de.tu_freiberg.informatik.vonwenckstern.ImageViewer\
de.tu_freiberg.informatik.vonwenckstern.ImageViewer.nocache.js

[5] This means if Firefox (user.agent = gecko1_8) is requesting ImageViewer.html then the dynamic content loader will return D86C78425EBF3471F02A10176A0C7644.cache.html and if this web page has been viewed with Internet Explorer 9 (user.agent = ie9) the site D2314070AA3C3D47EA100703B721615D.cache.html will be loaded.

[6] e.g. Firefox uses SpiderMonkey [Moz08] and IE9 uses Chakra [Mica]

[7] JavaScript uses single quotation for strings instead of double ones

[8] 67FDEAF3397887CF1E2008A6BC06B53D.cache.html

It is not recommendable at all to write much JavaScript code, because it is like assembly. This means the JSNI code will not run on all browsers or it will be interpreted differently from browser to browser. The JavaScript code cannot be debugged in Eclipse, it can introduce memory leaks, and due to its weak typed nature, GWT cannot perform any syntax check on JSNI. Another drawback of JSNI code is that the compiler is not able to optimize it as well as Java code, e.g. by changing global variable names to shorter ones, which will shorten the output code and speed up the entire application. Some books advise using (plain) JSNI to implement new browser functions in GWT; but if JavaScript Overlay Types are available, then these should be favored over (plain) JSNI. Overlay Types will not be covered in this paper[9]. GWT 2.5 uses this technique to implement new HTML5 features[10].

JSNI uses the Java Native Interface ("JNI enables the integration of code written in the Java programming language with code written in other languages" [Ora03]) to implement JavaScript code. For this reason the function declaration contains the additional keyword native, and "the body of a native method is given as a semicolon only, indicating that the implementation is omitted, instead of a block" [Gos07, p. 224]. This is why the JavaScript code is inside a Java comment block. /*-{ and }-*/ are the start and end points of the JavaScript method, so the compiler will notice it as JavaScript code instead of a usual comment. If the return type of the native method is not void, then the JavaScript code has to return an object of the correct type. Returning a newly created Java object in the JavaScript code as well as a long number is not allowed. Table 3 shows all possible return types. Now the other way round, passing objects from

Table 3: Passing JavaScript values into Java code (copied from [Goo06b])

Outgoing Java type	**What must be passed**
String	JavaScript string, as in return 'boo';
boolean	JavaScript boolean value, as in return false;
long	disallowed
Java numeric primitive	JavaScript numeric value, as in return 19;
JavaScriptObject	native JavaScript object, as in return document.createElement('div')
any other Java Object (including arrays)	Java Object of the correct type that must have originated in Java code; Java objects cannot be constructed from "thin air" in JavaScript

Java to JavaScript, will be explained. Table 4 lists all possibilities of transferring Java values to JavaScript. Since JavaScript does not expect any types, typing incompatibility can occur. Problems which can be caused by a weakly-typed language are shown in listing 10 and figure A30.

```
3  function addWithOut(a,b) {
4    var c = a + b;
5    alert('Result without type conversion is: ' + c);
6  }
7  function addWith(a,b) {
8    var c = Number(a) + Number(b);
9    alert('Result with type conversion is: ' + c);
10 }
```

Listing 10: Excerpt from code showing typing conversion in JavaScript; complete code (see listing A9)

[9] further information at [Goo06c]
[10] see [Goo06e] and [Goo06f]

Table 4: Passing Java values into JavaScript (copied from [Goo06b])

Incoming Java type	**How it appears to JavaScript code**
String	JavaScript string, as in `var s = 'my string';`
boolean	JavaScript boolean value, as in `var b = true;`
long	disallowed
other numeric primitives	JavaScript numeric value, as in `var x = 42;`
JavaScriptObject	JavaScriptObject that must have originated from JavaScript code, typically as the return value of some other JSNI method.
Java array	opaque value that can only be passed back into Java code
any other Java Object	opaque value accessible through special syntax

This is why it is important to be careful in the JavaScript world. Whenever parameters need to have a special type, the type conversion must be done manually as explained in [Gar05].

Java variables inside JSNI functions can be accessed by the following pattern: **objectName.@fully_qualified_class_name::name_of_variable** and **@fully_qualified_class_name::name_of_variable** for static fields.

Calling java functions from JavaScript works with almost the same pattern: **objectName.@fully_qualified_class_name::method_name(parameter_signature)(arguments)**. Table 5 shows the most common parameter signatures. Listing A10 and figure A31 display a

Table 5: Parameter signature of most common Java types (copied from [HT07, p. 289])

Type signature	**Java type**
Z	boolean
B	byte
C	boolean
S	char
I	int
J	long
F	float
D	double
L fully qualified class;	Fully qualified class
[type	type[] (an array)

complex example with many comments on how the interaction between Java and JavaScript works. It is also possible to send and receive information from the server by using JSNI, more information is available at [HT07, pp. 323-331].

After becoming familiar with the basics of JSNI, the next passage explains how to load and wrap a JavaScript library so that it can be used in GWT without any JavaScript code later. All necessary steps are shown by a simple example. The highlighter JavaScript library will be used to emphasize keywords of different programming languages. Firstly, the library has to be downloaded from [Ale05] and unzipped. Secondly, the syntaxhighlighter_3.0.83[11] folder will be copied into the Eclipse's web directory[12]. Thirdly, the library must be loaded by either including the lines showed in listing 11 in the ImageViewer.html document or by adding the code from listing 12 in the application's module file[13]. Wrapping the constructor of the JavaScript

[11] maybe a newer version have been downloaded and so the folder name is slightly different

[12] mostly named war or WebContent

```
9     <script type="text/javascript" src="syntaxhighlighter_3.0.83/
          scripts/shCore.js"></script>
10    <script type="text/javascript" src="syntaxhighlighter_3.0.83/
          scripts/shBrushJScript.js"></script>
11    <script type="text/javascript" src="syntaxhighlighter_3.0.83/
          scripts/shBrushJava.js"></script>
12    <script type="text/javascript" src="syntaxhighlighter_3.0.83/
          scripts/shBrushSql.js"></script>
13    <link type="text/css" rel="stylesheet" href="
          syntaxhighlighter_3.0.83/styles/shCoreDefault.css"/>
```

Listing 11: Excerpt from ImageViewer.html; complete code of ImageViewer.html (see listing A12)

```
<script src="syntaxhighlighter_3.0.83/scripts/shCore.js"> </script
    >
...
<script src="syntaxhighlighter_3.0.83/scripts/shBrushSql.js"> </
    script>
<stylesheet src="syntaxhighlighter_3.0.83/styles/shCoreDefault.css
    "/>
```

Listing 12: Example module file configuration for loading an external JavaScript library

library object into Java functions will be done in three steps: writing the implementation class, creating the JavaScriptObject class and packing the JavaScriptObject class in a Java GWT widget. The implementation class will contain all the JSNI code for interacting with the JavaScript library. The example JavaScript code explains how to use this library; unfortunately the example contains only HTML code (see listing 13) and no JavaScript one.

```
15 <pre class="brush:␣js;">
16 function helloSyntaxHighlighter()
17 {
18   return "hi!";
19 }
20 </pre>
```

Listing 13: Example code using syntaxhighlighter_3.0.83 library (excerpt from index.html)

This means the HTML code must be 'converted' into JavaScript as listing 14 illustrates.

```
1  <div id='d1'><div> <!-- place holder for the text -->
2  <script type="text/javascript">
3          var syntaxHighlighter = document.createElement('pre');
4          var d1 = document.getElementById('d1');
5          d1.appendChild(syntaxHighlighter);
6          syntaxHighlighter.className = 'brush: js';
7          syntaxHighlighter.innerHTML =
8          'function helloSyntaxHighlighter()\n' +
9          '{\n' +
10         '\treturn \"hi!\";\n' +
11         '}';
12 </script>
```

Listing 14: Corresponding JavaScript code of the HTML code in listing 13

[13] de.tu_freiberg.informatik.vonwenckstern/ImageViewer.gwt.xml

Line 3 creates the pre object and two lines later it is added to a given div element. Lines 6 and 7 set the properties of this object: the code and the language (in this example it is JavaScript) which should be highlighted. Listing 15 shows the code of the implementation class.

```
5  public class SyntaxHighlighterImpl {
6    public native SyntaxHighlighter create(Element el) /*-{
7      var synHighPre = document.createElement('pre');
8      el.appendChild(synHighPre);
9      return synHighPre;
10   }-*/;
11
12   public native void setLanguage(SyntaxHighlighter sh, String
         language) /*-{
13     sh.className = 'brush: ' + language + ';';
14   }-*/;
15
16   public native void setCode(SyntaxHighlighter sh, String code) /*
         -{
17     sh.innerHTML = code;
18   }-*/;
19 }
```

Listing 15: SyntaxHighlighterImpl.java in LoadJavaScript library's project(implementation class)

Line 6 defines the Java create method, line 7 creates the JavaScript object and line 9 returns it. Lines 13 and 17 set the properties of the JavaScript object which was passed as an argument of the functions setLanguage and setCode. Now the JavaScriptObject class SyntaxHighlighter, which is similar to a JavaScript handle for passing around in Java code, will be created. The handle is an opaque object in the Java world, because it is neither possible to see its properties nor is any Java code able to call its methods. The handle's properties and execution of its methods is only available in JavaScript code. Listing A13 demonstrates the code of the JavaScriptObject class. This class just contains normal Java functions which invoke the corresponding JSNI methods of SyntaxHighlighterImpl. Finally, the widget class having DOM elements as placeholders will be created. These placeholders will be passed as arguments to the create function of the implementation class. The DOM elements in the widget class can be compared to the empty div-tag in listing 13 at line 1. Since this case is so simple and only one placeholder is needed, the widget extends the HTML component and passes its own element as a placeholder to the implementation class (see line 18 at listing A14). This widget also contains the other two functions to change the JavaScript object properties. After having wrapped the SyntaxHighlighter JavaScript library into GWT, the widget can be used as a normal Java object. Listing A15 shows how to use the wrapped SyntaxHighlighterWidget, figure A32 displays the result. Whenever the SyntaxHighlighterWidget is being used, it is not obvious that a JavaScript library is doing all the work. This is an advantage of GWT, because the developer can stay in its Java world by using JavaScript libraries after they have been wrapped successfully.

The Smart GWT library works in the same way; it is a wrapper for the SmartClient JavaScript framework. This is the reason why it is very difficult to modify widgets of the Smart GWT library, because the code has to be written in JavaScript. In general the GWT compiler substitutes Java code by JavaScript one or replaces Java function calls by direct JavaScript method invocations to a library, and all this by using the JSNI interface. However, there are still some questions which have still not been answered:

- How does the compiler generate several browser outputs? (Until now, no browser-specific JavaScript or Java code has been defined.)

- Why do the output files have such strange names and what is meant by .cache.html and .nochache.js?
- What optimizations does the GWT compiler carry out (Java and/or JavaScript code)?

These questions will all be answered in the next subsections.

3.3.2 Deferred binding and bootstrapping process

Deferred Binding, which is the analogue to Java reflection, "is a technique used by the GWT compiler to create and select a specific implementation of a class based on a set of parameters." [Goo06a] By default GWT generates a specific JavaScript version for each browser. Other parameters can be for example the language and device type. If the project supports desktop, tablet and mobile versions in the languages English, German, French, Russian, Italian, Chinese and Spanish, then the compiler will generate $6 \cdot 3 \cdot 7 = 126$ specific website versions (see figure 4). The compiler generates a huge number of files in order to reduce the download size

o = compiler generated file for the parameter configuration: Firefox, German and mobile device

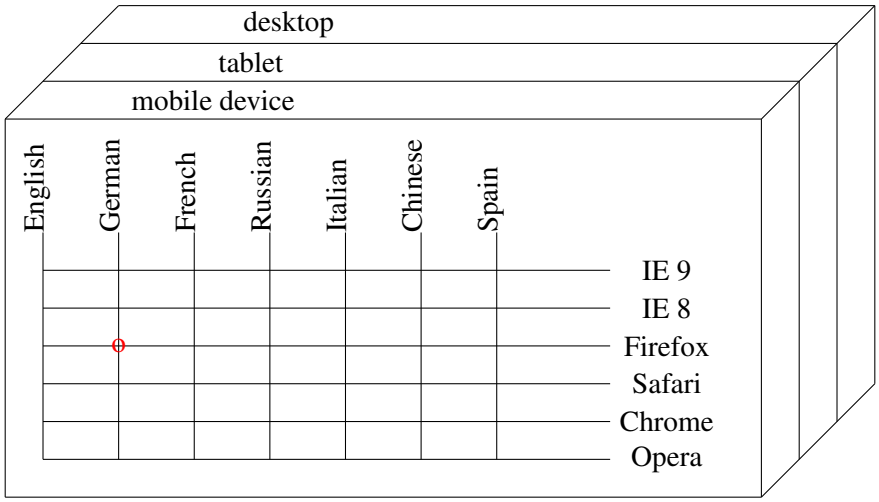


Figure 4: Files the compiler generated

and increase the execution speed. Because if it created just one file, the mobile phone user would have to download all the JavaScript code for every browser, the code for all the languages as well as the widgets and images in three possible sizes: small for mobile, medium for tablet and large for the desktop version. This would be in spite of only one browser and language being used and there not being use of the medium and large version on a mobile phone. It is very unlikely that a customer visiting a sale website with a mobile phone will come back when the site forces the browser to download all 126 specific versions. This download process would slow down the loading process and it would take several minutes to build the website.

Another drawback of delivering just one file to all web browsers is that this file contains many conditions which allow the JavaScript engines to execute the specific code depending on the parameter set. Listing 16 gives an example code, which must be executed every time a new item is shown. In this example eight instructions are needed, because the program executes seven condition tests for all languages and one assignment. If the server delivers the language-specific code, then only the one assignment instruction description = 'Intensivere Farben. Schärfere Kontraste. Geringere Blickwinkelabhängigkeit.' will be executed. Hopefully this extremely theoretical example has shown the advantage of deferred binding.

```
var description = '';
if(language == 'en') {
        description = 'It delivers rich color and deep contrast
            from every angle';
} else if(language == 'fr'){
        description = 'Des couleurs profondes. Un contraste marque
            . Des angles de vue plus larges.';
} ...
else if(language == 'de'){
        description = 'Intensivere Farben. Schärfere Kontraste.
            Geringere Blickwinkelabhängigkeit.';
}
```

Listing 16: Example JavaScript code choosing the right language

The drawback of this technique is the long compilation time to create the 126 HTML files. This section part will start with deferred binding by replacement. "Replacement means overriding the implementation of one java class with another that is determined at compile time." [Goo06a] This mechanism will be described in an example which creates a special message depending which browser is used (see figure A33). At first the general interface class (see listing 17) will be created.

```
public interface Status {
  public void getStatus();
  public Widget asWidget();
}
```

Listing 17: Status.java in deferred binding with replacement project (package name and imports are omitted)

The following classes StatusLabel (see listing A17), StatusLabelIE9 (see listing 18) and StatusLabelFF (see listing 19) will implement this interface.

```
public class StatusLabelIE9 extends Label implements Status {
  @Override
  public void getStatus() {
    this.setText("You␣are␣using␣Internet␣Explorer␣9.");
  }
}
```

Listing 18: StatusLabelIE9.java in deferred binding with replacement project (package name and imports are omitted)

```
public class StatusLabelFF extends Label implements Status {
  @Override
  public void getStatus() {
    this.setText("You␣are␣using␣Firefox.");
  }
}
```

Listing 19: StatusLabelFF.java in deferred binding with replacement project (package name and imports are omitted)

The code Status status = GWT.create(Status.class) creates the Status object in the ImageViewer (complete code see listing A16) class. In order to use the compiler's deferred binding mechanism, the new keyword, as shown in the following code line Status status = new

Status(), cannot be used to create any specific code. The binding rules are defined in ImageViewer.gwt.xml (see listing 20).

```
5
6   <!-- Fall through to this rule is the browser isn't IE 9 or
        Mozilla -->
7   <replace-with class="de.tu_freiberg.informatik.vonwenckstern.
        client.StatusLabel">
8       <when-type-is class="de.tu_freiberg.informatik.vonwenckstern
            .client.Status"/>
9   </replace-with>
10
11  <!-- Mozilla has a different implementation -->
12  <replace-with class="de.tu_freiberg.informatik.vonwenckstern.
        client.StatusLabelFF">
13      <when-type-is class="de.tu_freiberg.informatik.vonwenckstern
            .client.Status" />
14          <any>
15              <when-property-is name="user.agent" value="gecko
                    "/>
16              <when-property-is name="user.agent" value="
                    gecko1_8" />
17          </any>
18  </replace-with>
19
20  <!-- IE 9 has a different implementation -->
21  <replace-with class="de.tu_freiberg.informatik.vonwenckstern.
        client.StatusLabelIE9">
22      <when-type-is class="de.tu_freiberg.informatik.vonwenckstern
            .client.Status"/>
23          <when-property-is name="user.agent" value="ie9" />
24  </replace-with>
```

Listing 20: ImageViewer.gwt.xml in deferred binding with replacement project (module definition is omitted)

Lines 7 to 9 describe the default case with the standard replacement. Lines 12 to 18 say that the interface should be replaced with the Mozilla specific implementation if the user agent is either gecko or gecko1_8. The next block declares that the Status interface should be replaced by Internet Explorer 9 specific implementation if the user agent is ie9.

The passage explains deferred binding using generators. "Generators are classes that are invoked by the GWT compiler to generate a Java implementation of a class during compilation." [Goo06a] Instead of writing a different class for each browser as in the replacement example above, a generator class - which automatically writes the source code classes for the different browsers - will be created at compile time. The advantage of this technique is that there is no need to copy one file several times and just change a little part of it. The disadvantage is that it is more complex and strange errors can occur during the compilation process.

The best way to understand this method is to show an example. For reason of comparability, the same example as above is used. This is why the files ImageViewer.java (see listing A16), Status.java (see listing 17) and StatusLabel.java (see listing A17) are the same. The module's project file is shown in listing 21. Lines 5 to 7 say that the compiler should generate the browser-specific Java file for the Java type Status. Instead of implementing the classes StatusLabelFF (see listing 18) and StatusLabelIE9 (see listing 19), the Generator class - which creates these classes - will be defined. The StatusGenerator class is displayed in listing 22. It is important that the generator class is not in the client package, because this code cannot be translated to JavaScript.

```
5   <generate-with class="de.tu_freiberg.informatik.vonwenckstern.
        StatusGenerator">
6     <when-type-assignable class="de.tu_freiberg.informatik.
          vonwenckstern.client.Status" />
7   </generate-with>
```

Listing 21: ImageViewer.gwt.xml in deferred binding with generator project (module definition is omitted)

```
15 public class StatusGenerator extends Generator {
22   public String generate(TreeLogger logger, GeneratorContext
         context,
23     String typeName) throws UnableToCompleteException {
27       userAgent = context.getPropertyOracle().
             getSelectionProperty(logger, "user.agent").
             getCurrentValue();
33       SourceWriter sw = getSourceWriter(typeName, context,
             logger, userAgent);
38       sw.println("@Override");
39       sw.println("public void getStatus() {");
40       sw.println("this.setText(\"You are using " + browser + "
             .\");");
41       sw.println("}");
45       sw.commit(logger);
46       System.out.println("class '" + typeName + userAgent + "
             Generated' was created succesfully");
47       return typeName + userAgent + "Generated";
52     }
73     public SourceWriter getSourceWriter(String typeName,
         GeneratorContext context,
74       TreeLogger logger, String userAgent) throws
           NotFoundException {
80       String simpleName = classType.getSimpleSourceName();
82       simpleName = simpleName + userAgent + "Generated";
86       composer.setSuperclass("com.google.gwt.user.client.ui.
             Label");
88       composer.addImplementedInterface("de.tu_freiberg.
             informatik.vonwenckstern.client.Status");
90       composer.addImport("com.google.gwt.user.client.ui.Label");
99         SourceWriter sw=composer.createSourceWriter(context,
             printWriter);
100        return sw;
103    }
104 }
```

Listing 22: StatusGenerator.java in deferred binding with generator project (complete code see listing A18)

In line 15 the class StatusGenerator extends the standard Generator class which generates the source code during the deferred binding process. Because the standard Generator class is abstract, the generate method must be implemented. In line 22 it creates the specific class and returns the name of the new class, which the compiler uses instead of the requested one. The TreeLogger class is an abstract class, which logs several messages during deferred binding. The GeneratorContext interface provides several metadata for deferred binding. The metadata are:

- The parameter set, which is used to generate the class.
- The typeName is the name of the requested type, which will be substituted. In this case is typeName = 'Status', because the following expression Status status = GWT.create (Status.class) in the ImageViewer.java file invoked the deferred binding generator.

Line 27 returns the user agent for which the compiler generates the specific file. Line 33 calls the function getSourceWriter, which creates an empty class, e.g. Statusgecko1_8Generated, in line 82. The method also includes all required imports in line 90. The created class extends the Label class (see line 86) and implements the Status interface (see line 88). Line 100 returns the SourceWriter object. The result is used to add the function implementation of the generated class in lines 38 to 41. The command sw.commit(logger) flushes all the generated codes into a Java file in line 45, and the logger notes that the file is completely created. Line 47 returns the generated file name. Compiling this project creates the output as shown in listing 23. The console text displays all six browser-specific generated Java files, which will be used in the compilation process to generate JavaScript code later.

```
4 class 'de.tu_freiberg.informatik.vonwenckstern.client.
      Statusgecko1_8Generated' was created succesfully
5 class 'de.tu_freiberg.informatik.vonwenckstern.client.
      Statusie6Generated' was created succesfully
6 class 'de.tu_freiberg.informatik.vonwenckstern.client.
      Statusie8Generated' was created succesfully
7 class 'de.tu_freiberg.informatik.vonwenckstern.client.
      Statusie9Generated' was created succesfully
8 class 'de.tu_freiberg.informatik.vonwenckstern.client.
      StatusoperaGenerated' was created succesfully
9 class 'de.tu_freiberg.informatik.vonwenckstern.client.
      StatussafariGenerated' was created succesfully
10    Compiling 6 permutations
```

Listing 23: Compiler output in deferred binding with generator project (complete code see listing A19)

Figure A34 shows the compiled result in different web browsers. In the second subfigure Internet Explorer 9 was running in compatibility mode. It displays the message 'Internet Explorer6' instead of its real compatibility user agent 'Internet Explorer7', because the GWT 2.4 compiler creates only specific web pages for the user agents ie6, ie8, ie9 and more, but no website for ie7. This is why the bootstrapping process loads the specific code for ie6.

The next paragraph explains the bootstrapping process in more detail, and tries to explain the meaning of generated filenames.

```
1 <html>
2   <body onload='alert("w00t!")'>
3     <img src='bigImage0.jpg'></img>
4     <script source='externalScript0.js'></script>
5     <img src='bigImage1.jpg'></img>
6     <img src='reallyBigImage2.jpg'></img>
7     <script src='myApp/myApp.nocache.js'></script>
8     <script src='externalScript1.js'></script>
9   </body>
10 </html>
```

Listing 24: HTML example describing the order of loading a web page (copied from [Goo06h])

Listing 24 shows a bootstrapping sequence example. Figure 5 illustrates the loading sequence of listing 24. Firstly the entire HTML file will be downloaded and parsed. If it contains links to other documents like images or scripts then these files will be transferred in order of their appearance. There are two rules: Image files will not block the downloading process, but script files will block downloading and the process will only continue if the script document has completed transferring.

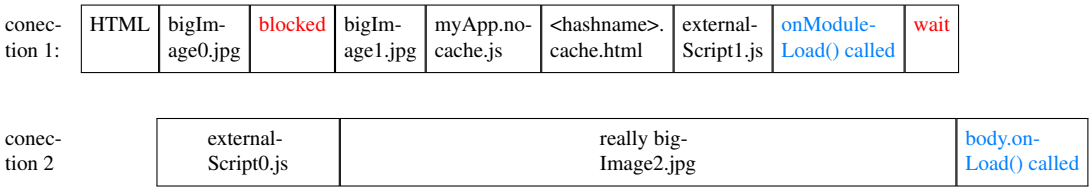


Figure 5: Loading sequence of a website (summary from [Goo06h])

The <hashname>.cache.html files contain the actual JavaScript program code written in an HTML file. The JavaScript code is wrapped, because some browsers do not support compression of pure JavaScript. "Since the GWT mantra is no-compromise, high-performance AJAX code" [Goo06d] the compiler wraps the program code in an HTML file to avoid sending the application code uncompressed. The <hashname> is the md5 sum of their content. This guarantees the web browser that this content will never change and so the file can and should be cached, which is the reason for their .cache.html extension. Modifying the Java code of any project will change the corresponding JavaScript and the <hashname> will be different. This means that the browser has to download a different file and will not use the cached one. The myApp.nochache.js contains the code to find the browser-specific program version and all other required parameters. After the parameter set has been figured out, the browser starts loading the associated <hashname>.cache.html file. This is the place where deferred binding occurs at runtime. As the filename indicates, this JavaScript file should never be cached as the GWT compiler generates different contents under the same filename every time. This means if the browser cached this file, then it would use the old cached content instead of downloading the new one. The <hashname>.gwt.rpc file contains all types which implement java.io.Serializable interface and are allowed to be serialized from client to server or vice versa due to security reasons. The <hashname>.cache.png files contain several images which are bundled together to one big image. As above, the hashname is the md5 hash of its content, so this name is unique and can be cached by the browser. The advantage of image bundles will be discussed in section 3.8.1. The last compiler generated file is the hosted.html one, which contains the HTML and JavaScript code for the hosted mode. This code is looking for and connecting to the GWT plugin, which interprets Java code as JavaScript, and displays errors from the GWT plugin in the web browser. Since the GWT compiler does a lot of optimizations like creating cached and no-cached files, compiled AJAX applications are much faster than handwritten ones[14].

3.3.3 GWT compiler steps and optimizations

The aim of this subsection part is to describe the compiler steps and its optimizations, which keep the application relatively small although many libraries were imported. This allows downloading and starting the web application in a short period of time.

Since the GWT compiler is a normal Java program, the compilation process can be debugged in Eclipse to understand the compilation steps in more detail. This paragraph describes the

[14] for "real" programs containing several forms and controls and not just only one button, where the bootstrapping overhead is larger than the application code itself

setup to start debugging the compilation process. Firstly a normal GWT project with the name "DA_GWTCompiler_Optimizations" will be created, secondly the Eclipse debug configuration will be opened, a double click on Java Application will be done and "Compiler" will be entered as debug configuration name. In the Main tab the Project field is set to "DA_GWTCompiler_Optimizations", the Main class to "com.google.gwt.dev.Compiler" and check the option "Stop in main". The following text "-logLevel ALL -style PRETTY de.tu_freiberg.informatik.vonwenckstern.ImageViewer" will be entered into the Program arguments field of the Arguments tab. In the same tab the working directory has to be "default". User Entries have to be selected in the Classpath tab. After the Advanced ... button on the right side has been toggled and Add External Folder selected in the popup dialog, the ".../DA_GWTCompiler_Optimizations/src" folder can be chosen in the tree. After pressing the Debug button, the Eclipse debugger will stop at the `main` function. This is the starting point to explore the compilation process.

Abstract Syntax Tree and Visitor Pattern

Before discovering the GWT compiler, it is better to mention Abstract Syntax Tree (AST). The best way to understand AST is by playing around with the Eclipse AST plugin (download site at [Tea]), which shows the AST tree of a selected Java file. AST is a "precise and fully resolved compiler parse tree" [Aes] representing the syntax of a Java source file. Some Eclipse AST node types are:

- MethodDeclaration for a function,
- JavaDoc for the `/** ... */` comments over the method declarations,
- Modifiers are keywords like `public`, `static`, `native` and so on.

All Eclipse AST node types are listed in table A3. For a better understanding of the Abstract Source Tree, figure A35 shows the AST of the simple Java class shown in listing A20. During the compilation process the AST will be changed several times, e.g. after deleting unused variables and methods. Implementing the compiler can be done using different patterns like the inheritance pattern, visitor pattern or the compiler matrix pattern. Since the inheritance pattern is the naive way of implementing the compiler and Google uses the visitor pattern for their compilation process, the matrix pattern (for more information see paper [XW]) will not be described in this subsection. Using the inheritance pattern means (1) declaring an abstract interface for all AST nodes containing virtual methods as node operations (like code error checking, optimizations, code generation and obfuscation) and (2) defining a class for each AST node, which implements the methods inherited from the super node class. Figure A39 shows an Unified Modeling Language (UML) diagram using this pattern. The disadvantage of this pattern is that each node class implements many methods with different algorithms. As a result algorithms such as error checking are spread across numerous nodes, having to introduce global variables or pass many arguments by reference for each method call. It is also nearly impossible to change or add any algorithm later because all the functions have to be updated in different node classes by considering the calling hierarchy order. A better solution would be the separation of the algorithm from the object structure. The double dispatch pattern solves the separation by encapsulating all methods performing one operation into a single class.

This way all required variables for executing this algorithm are in one class and so there is no longer need for global variables or passing references. Another advantage is that the classes defining the AST structure were not modified, and so new operations -by creating new classes- can be added. This is especially important when there are no writing rights in the operation classes, e.g. in shipped JAR archives. Figure A40 displays the UML diagram, which manipulates Java AST using the inheritance pattern and the double dispatch one. Since the

double dispatch pattern is not natively supported in Java[15] it must be emulated by an extensive usage of the instanceof operator (see shape example at listing A21) or by the visitor pattern. To implement the visitor pattern a Visitor interface or a superclass must be created. This would contain different visit methods, and a Visitable interface or an extra class containing an accept method which receives the Visitor as argument.

It is important that each class inheriting the accept method needs to override this one, even if these are always the same lines of code return visitor.visit(this), because this is the point where double dispatching occurs. More precisely: Firstly the accept method of element A is invoked with parameter B, e.g. A.accept(B). The specific method invocation is chosen by the dynamic type of the element A (single dispatch) and the static type of the visitor B. Secondly when the visitor calls the associated visit(this) method, its implementation is chosen by the dynamic type of the visitor B and the static type of the element A known from the implementation of accept(this), which is the same as the dynamic type of the element. This means the double dispatch mechanism is emulated by executing single dispatching twice in succession.

Listing A22 implements the shape example using the visitor pattern. In the example code there are two shapes shape1 = new Triangle() and shape2 = new Heptagon(), which have the same static type ShapeVisitor. The line shape1.accept(shape2) is the starting point of the double dispatch pattern. Firstly Java does the single dispatch, this means Triangle::accept (ShapeVisitor shape2) is called, because shape1 has the dynamic type Triangle and shape2 has the static type ShapeVisitor. As mentioned above, every accept function calls only visitor.visit(this). The this variable has the static type Triangle now, because the Triangle::accept (ShapeVisitor) method has been invoked and the Triangle class overrides the accept method to change the static type from the this variable to its own class type Triangle. The line visitor.visit(this) in the Triangle::accept(ShapeVisitor visitor) method, where visitor is equal shape2, invokes the function Heptagon::visit(Triangle), because the dynamic type of visitor is the dynamic type of shape2, being Heptagon, and the static type of this is Triangle. This means after the visitor pattern has been executed, the code shape1.accept(shape2) invokes the function Heptagon::visit(Triangle), which represents the double dispatch function call of the dynamic types of the two shape objects.

The visitor pattern also has disadvantages, for example after adding a new node type, all visitor classes have to be changed. For Google this is not really a disadvantage, because the AST node types represent the Java language specification and this will not recently change. On the other hand the advantage of using the visitor pattern is great, because it is easy adding new compiler steps or changing existing ones.

This paragraph finishes by showing how to use the Eclipse AST library together with the visitor pattern. The example shown below modifies an existing Java source code. The example can be seen as a little prototype showing how a compiler may work. Figure A41 displays the UML diagram. Listing 25 gives a sample implementation of the accept method for the classes extending ASTNode. Source code for all classes, which extend ASTNode, is available at [Ecl07]. The traversing process starts every time by calling CompilationUnit::accept(ASTVisitor). The next lines describe the process for the FieldPrinter. This means firstly FieldPrinter::visit (CompilationUnit) is called which is not implemented in the FieldPrinter class. This means ASTVisitor::visit(CompilationUnit) is invoked, which will do nothing else than returning true, and so in the accept method all the children nodes of CompilationUnit node[16] will accept the FieldPrinter as visitor. This means once FieldPrinter::visit(PackageDeclaration), many

[15] Java supports only the single dispatch mechanism, which invokes the method at runtime depending at the object's type which is calling the function. Java ignores the runtime types of the arguments of a method by invoking it. Method overloading will be done by matching the argument types for function calls at compile time and not at runtime.

[16] having the types PackageDeclaration, ImportDeclaration and TypeDeclaration (see figure A35 for ASTNode hierarchy)

```
1 void accept(ASTVisitor visitor) {
2   boolean visitChildren = visitor.visit(this);
3   if (visitChildren) {
4     // visit children in normal left to right reading order
5     for(ASTNode child: children) {
6       child.accept(visitor);
7     }
8   }
9 }
```

Listing 25: Prototype implementation of the accept method for the classes extending ASTNode

times FieldPrinter::visit(ImportDeclaration) and several times FieldPrinter::visit(TypeDeclaration) are called. If FieldPrinter::visit(P) returns true, then FieldPrinter::visit(S) will be called for all children nodes S of the parent node P, otherwise the entire sub tree of the parent node P will be skipped and will not be visited. Now it is known how the abstract source tree is traversed and in what order the visit functions are called.

```
52      ASTParser parser = ASTParser.newParser(AST.JLS3);
53      parser.setKind(ASTParser.K_COMPILATION_UNIT);
54      parser.setSource(content.toCharArray()); // set source
62      ASTNode node = parser.createAST(null);
69      ASTPrinter.exec(node);
70      ASTOptimizer.exec(node);
71      ASTRenamer.exec(node, "de.tu_freiberg.informatik.TestClass2");
77     String s=org.eclipse.jdt.internal.corext.dom.ASTFlattener.
           asString(node);
```

Listing 26: Main.java in AST project.(complete code see listing A23)

In lines 52 to 54 in listing 26 the ASTParser will be created. This parses the entire Java file (K_COMPILATION_UNIT) and returns its content as char array containing the text of the Java file. This means the entire Java file will be parsed. The parser starts analyzing the Java file, creating the AST structure of the content and returning the AST root node by calling createAST. The methods ASTPrinter.exec(node), ASTOptimizer.exec(node) and ASTRenamer.exec(node) will print some information of the abstract source tree to the console, delete unused private functions and fields, and change the package and class name of the Java file by manipulating the corresponding AST nodes. After having manipulated the Java AST, Java source code will be created and written into a different Java file by calling ASTFlattener.asString(node). The explanation on how to manipulate Java AST will be given at the ASTRenamer class (see listing 27).

Since the AST nodes should visit the functions to manipulate them, the ASTVisitor class has to be extended as shown in line 14. The static method exec calls CompilationUnit::accept and passes the root AST node to it. This is the starting point of the traversing process. If traversing reaches the PackageDeclaration node, the method ASTRenamer::visit(PackageDeclaration) is invoked and it changes the package declaration name to the wanted one. If the TypeDeclaration node is called (this means the node is the declaration of an interface or a class – see table A3) a test will be done. The check tries to find out whether the class or interface is declared as public to match the name of the Java file later. If the result is positive, the interface or class name will be changed to the desired one. Both visit functions return false, because neither the child nodes of the package declaration nor the type declaration should be traversed. Listening A23 shows the complete example including source code and console output.

```
14 public class ASTRenamer extends ASTVisitor {
17   public static void exec(ASTNode node, String newClassName) {
18     CompilationUnit cu = (CompilationUnit) node;
19     cu.accept(new ASTRenamer(newClassName));
20   }
26   public boolean visit(org.eclipse.jdt.core.dom.PackageDeclaration
         packageDecl) {
27     String pName = className.substring(0,className.lastIndexOf
         (".") );
28     packageDecl.setName(packageDecl.getAST().newName(pName));
29     return false;
30   }
32   public boolean visit(org.eclipse.jdt.core.dom.TypeDeclaration
       typeDeclaration) {
33     if(Modifier.isPublic(typeDeclaration.getModifiers())) {
34       String name = className.substring(className.lastIndexOf(".")
           +1);
35       typeDeclaration.setName(typeDeclaration.getAST().
           newSimpleName(name));
36     }
37     return false;
38   }
```

Listing 27: ASTRenamer.java in AST project.(complete code see listing A26)

General overiew of compilation steps

This paragraph summaries the code of the GWT compiler and therefor many explanations are copies from Google's codes and comments of the package and sub packages of com.google.gwt.dev, especially com.google.gwt.dev.jjs.

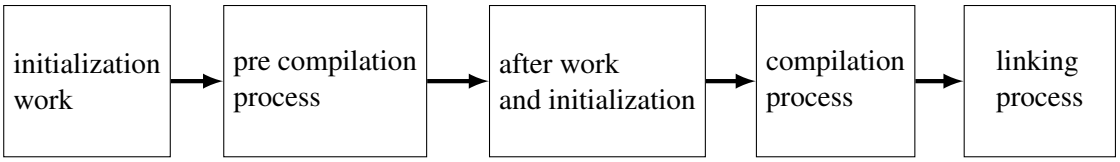


Figure 6: General steps of the GWT compiler

Figure 6 shows an overview of the entire compilation process. At the initialization step, command line arguments were tested and passed to the CompilerOptions class (see figure A42 for the command line parameters) and if any parameter is missing then all parameter options will be printed to System.out. A GUI or a normal PrintWriter logger will be created – which displays the output at a given detail (Error, Warn, Info, Trace, Debug, Spam, All). After the logger is created, the function CompileTaskRunner.doRun is called, which starts the other steps of the compilation process, catches all exceptions during the compilation and gives these to the logger. In the first step the compiler also checks once a day if a new version of the GWT API is available and if so it notifies the logger. Later on, all needed modules will be loaded and a working directory for the compilation will be created at System.getProperty("java.io.tmpdir")[17]. At a later point, a GWT unit cache is loaded which contains compilation artefacts to speed up the compilation process. Opening the unit cache file in a text editor, shows that it is a combination of several source files. In this example it contains 2275 different code files. Figure 7 illustrates

[17] the folder name was: C:\Users\«User Accoutname»\AppData\Local\Temp\gwtc1813510714541001888.tmp

the sub steps of the pre compilation process, which will be explained later in more detail. In the step after the pre compilation process, identical permutations are merged together in order to save later one or more permutations and unique ids for the permutations are generated. In the same phase the initialization for the compilation process starts by creating a working factory, which listens for external workers at port 51460 to get the compilation results. This means the compilation permutations are executed by a given amount of workers, which can be set by the –localWorkers flag. Figure 8 illustrates the sub steps of the compilation permutation, which will be described later. The compilation process returns a number of JavaScript artefacts, which will be merged together in the linking step. The Link class uses different Visibility states to determine whether the client needs the generated outputs or if these files are only used for the deploying process on the server.

Figure 7: General steps of pre compilation process

Figure 7 shows additional details about the pre compilation process. Firstly the compiler looks for all possible rebinds of the declared entry points. These are all the building paths the compiler has to take care of later. Afterwards it searches for all JavaScriptObjectImpl types, which are not often used directly in the Java source code. This is to protect them from being removed in the Abstract Syntax Tree when the dedicated Eclipse compiler compiles the Java project. The compiler will look for correlations in the source files to find out which classes should be included for a defined compilation path.

Now the Eclipse Java development tools (JDT) compiler compiles the Java source code using GWT specific concepts like JSNI and deferred binding to create the JDT AST parse tree. The compiler then looks through a list of compiled units in order to find any errors which occurred during the JDT compilation. After this step the compiler creates raw, unfinished and unlinked AST nodes for types, methods, fields and parameters by mapping JDT AST nodes to these newly created AST nodes and parsing all JSNI source codes. Afterwards the compiler checks for syntactic JSNI errors and aborts the compilation if any have occurred. The compiler records all super classes and interfaces of a given class including this class itself, e.g. checking the class String would return as super classes Object and String and as super interfaces Serializable, CharSequence and Comparable<String>. The compiler creates GWT AST from JDT AST by combining information from JDT type nodes and the type map to generate a JProgram structure and to carry out auto boxing of Java types. After this the compiler can check the GWT AST for semantic JSNI errors and aborts if any occur. The next step is the normalization of Java AST. This contains several sub steps: (1) Complex unboxing of expressions like x++ or x+=3 will be done in two steps, firstly x++ will be converted to x=x+1 and secondly auto boxing will change the result to x=box(unbox(x)+1). (2) All assertions from the AST will be removed to speed up the code. (3) The compiler replaces GWT.create calls with the creating call defined in deferred binding by swapping out GWT.create nodes with JGWTCreate ones. (4) GWT.runAsync will be replaced with a fragment loader, which downloads the JavaScript code when it is needed. (5) An initial loading sequence of split points will be chosen by identifying split points, which must be loaded before other ones. This means the compiler takes care of code dependencies. (6) Entry points will be resolved and non-static ones will be rebounded. (7) The compiler creates a field representing implementations of class literals. Now the normalized Java code will be optimized, which will be explained in the next paragraph. The compiler records all available

rebinds again, as some rebinds could have vanished during the optimization. The pre compilation process finishes creating a unified, non-permutation specific AST using the GWT Java AST.

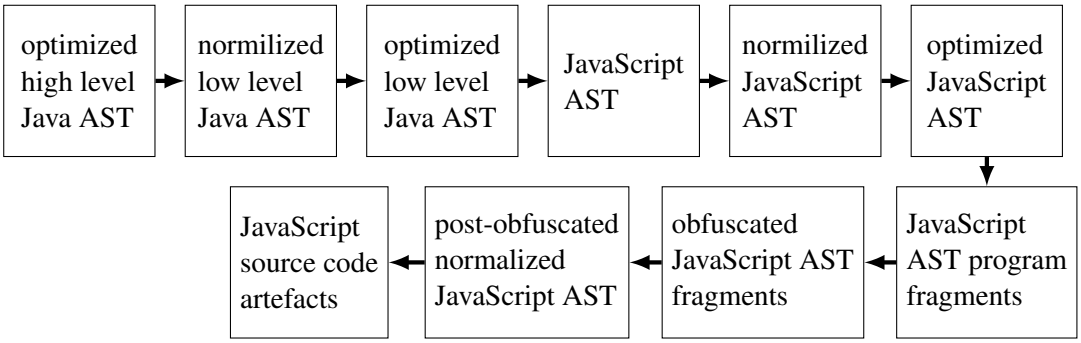


Figure 8: General steps of one compilation permutation

After explaining the pre compilation step, figure 8 gives more information about the compilation process. Firstly the compiler exception class for unexpected and unsupported state of operations is loaded. Later all compiler options for this permutation like ClientBundle.enableInlining = true and compiler.emulatedStack=false and the unified AST from the pre compilation are loaded. Afterwards the compiler resolves rebinds by inserting permutation-specific deferred binding code. Java AST code will be optimized again with permutation-specific optimization options. Calls to empty super constructors will later be removed. Now the high-level Java tree will be normalized into a lower-level tree, which is more suited for JavaScript code generation. The code generation is done in several sub steps: (1) The compiler creates functions for single runtime dynamic dispatch and then it matches dynamic dispatches to static ones. This is done if the compiler does not know the object type and so it has no idea what overwritten virtual function should be called. That is why an if statement block is generated. This contains all possible opportunities to call the method depending on the object type. (2) Multi-catch blocks will be merged into a single block, e.g. listing 28 will become listing 29.

```
try {...}
catch(NullPointerException e) {...}
catch(IOException e) {...}

.
```

Listing 28: Multi-catch block

```
try {...}
catch(Exception e) {
  if(e instanceof NullPointerException){}
  else if(e instanceof IOException) {...}
}
```

Listing 29: Single-catch block

(3) Problematic type compound assignments (e.g. long assignments) will be replaced with a sequence of more simple operations and these will be optimized by the compiler to prevent calculating operations twice, e.g. x = l+w; w =2*(l+w) will become temp = l+u; x = temp; w =2*temp. (4) The compiler handles casts from Java type long and long operations are replaced with calls to the emulation library, because JavaScript one does not support the long type. (5) Cast and instanceof operations are substituted for the Cast class. This is needed because JavaScript is typeless and the Cast class contains a queryId telling the JavaScript code what type it should be. The queryId = 0 tells the JavaScript engine it was Java type Object and casting will always succeed. (6) The compiler replaces array access and instantiations like c = new char[3] with calls to the Array class, because JavaScript creates arrays as c = new Array("Anita", "Ema"). (7) Java's equal operator (==) will be changed to JavaScript's identical operator(===); but if the equal operator returns the same as JavaScript's identical operator, then the equal operator will be used in order to save one character each time. More information about JavaScript's equal and identical operator is available at [Cha06, pp. 80–82].

After the normalization step the compiler will carry out further optimizations by removing globally unreferenced classes, interfaces, methods, parameters and fields form AST and by deleting JavaScriptObject single implementations when the implementer is no longer in the abstract source tree. Now the major step creating JavaScript AST from GWT Java AST will be executed. After this the JavaScript AST will be normalized in several sub steps: (1) Semantic JavaScript errors, which were introduced by converting Java AST to JavaScript AST, will be fixed. (2) All unresolved JavaScript name references will be resolved. (3) The compiler moves all function definitions to the top of the file, so it does not have to worry about function definitions in blocks. That means lexical scoping can be used instead of dynamic (nested) scoping. An advantage of lexical scoping is that the browser can invoke function calls faster, because it has only to search the function names from top to bottom instead of parsing the complete context. Now the optimizeJS function is called in order to optimize the JavaScript code which will be explained in the next paragraph. After this all case labels will be combined with identical bodies, e.g. listing 30 will become listing 31.

```
switch(x) {
case 0: y = 17; break;
case 1: y = 17; break;
case 2: ...
}
```

Listing 30: Uncombined case labels

```
switch(x) {
case 0: case 1: y = 17; break;

case 2: ...
}
```

Listing 31: Combined case labels

In the next step the GWT compiler emulates JavaScript stack, providing useful stack traces on browsers which do not offer stack information. The JsProgram AST is divided into multiple fragments, because the initial fragment (which is all except for anything called in a callback of GWT.runAsync which will be downloadable via AsyncFragmentLoader.inject) is sufficient to run all of the program's functionality. JavaScript AST will be obfuscated in order to decrease download size later. After this the JavaScript AST will be rewritten to handle references from one code fragment to another better. Before generating JavaScript code from JavaScript AST, large var statements will be divided into smaller ones, e.g. var a,b,c,...,x,y,z becomes var a,b,c,...,l,m; var n,o,...,y,z. This is needed because very long var statements cause trouble on some browsers. Later on, the compiler creates the permutation properties, a serialized symbol map for allowed remote procedure calls, and the permutation result object containing JavaScript code as bytes. If the –soyc, which is an abbreviation for story of your compile, flag is used, then different information about the compilation process will be added to the permutation result object.

GWT compiler optimizations

Almost all Java code optimization is done in the com.google.gwt.dev.jjs.JavaToJavaScriptCompiler::optimize function. This method contains a loop, which calls several times the optimizeLoop in order to do one optimization step. The number on loop runs depends on the optimization level x: If x is between one and eight, then at most x loop runs will be done. But if the optimization level is equals nine (maximum), then the loop will be executed as long as the optimizeLoop will have no optimization effect on the source code. Numerous loop executions are done, because the previous run can unlock better optimizable code for the next one. If the aggressive optimization option has not been disabled by setting the compiler flag -XdisableAggressiveOptimization, then the compiler does data flow optimization after the optimization loop.

Figure 9 shows the optimization phases, which are done in the optimizeLoop function. All the implementations of the optimization classes can be found in the com.google.gwt.dev.jjs.impl

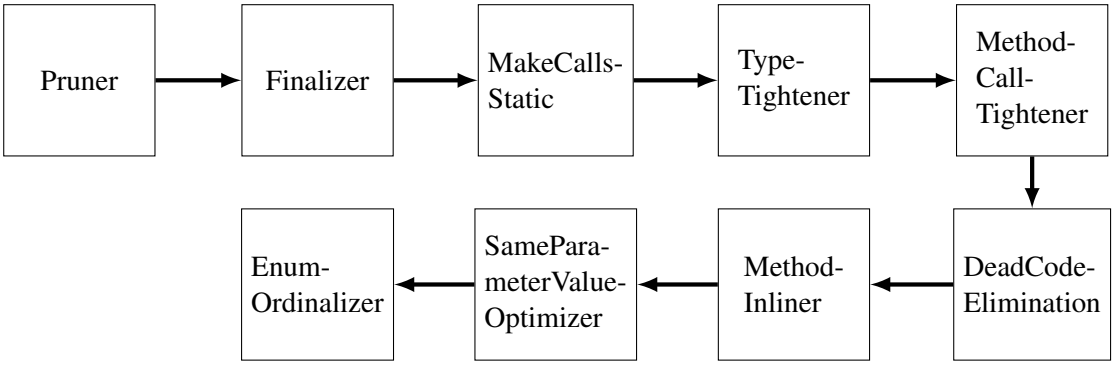


Figure 9: Optimization order in optimizeLoop(), names are the GWT class names perfoming the optimization phases

package. Java code pruning means deleting globally unreferenced AST nodes by determining their reachability from the root nodes (entry points) based on class creations with the new operator and function invocations. The pruner will do no local code flow observation, it will only remove classes and interfaces in addition to fields, methods and parameters. This is the most important optimization phase, because it will delete all included but unused GWT library classes. If this step did not exist, then the generated JavaScript code would not be downloadable at all, because the server would transfer all the GWT widgets, which are automatically included, even if these are not used at all. The finalizer detects and marks all nodes that are 'effectively' final; e.g. classes that are not inherited, functions which have never been overridden, and only once assigned variables. The next phase makes final functions to static ones containing the original method's body and an instance functions delegating to this new static method; e.g. a.add(b) will be delegated to add(a,b). The main reason for doing this is to save later the this keyword in the compiled JavaScript code making it shorter. The type tighter substitutes general types like List with more specific ones like ArrayList in order to reduce instanceof checks and dynamic type castings. Examples: final List foo = new ArrayList() will become final ArrayList foo = new ArrayList() and Collection bar() { return new LinkedHashSet(); } will become LinkedHashSet bar() { return new LinkedHashSet(); }. The main issue doing this optimization is that the next phase "method call tightening" can remove useless generated run-time dispatch code for virtual function invocations. This means the compiler knows the concrete class and so it is able to invoke the specific method call, now. This operation also sets the type of a variable to null, if it was never initialized, or only once null assigned and was not reassigned. This allows the compiler to do further optimizations. "Method call tightener" replaces polymorphic method calls to more specific ones, e.g. List foo = new ArrayList(); foo.add("bar") will become ArrayList foo = new ArrayList(); foo.add("bar") by type tightening and so instead of List::add the function ArrayList::add is called. This means no JavaScript single dispatch replacement code must be generated. The dead code elimination step does even more than just eliminating code - table 6 shows the different operations.

The next paragraph gives further explanations of table 6. In (1) simple expressions like null == null occur, because the type tightening also sets variable types to null. In (8) nearly all if and else keywords are removed in order to get better compressed JavaScript code later. (9) Substitutes post operations with pre operations to speed up runtime, because the browser can save one copy operation of the variable. The "method inline" phase inserts function bodies containing no more than two lines of Java code and no other method invocations. This way the application speed can be increased, because the JavaScript interpreter does not have to jump to the methods and in most cases embedding maximal two code lines into multiple places is shorter than the extra method declaration plus the calls to it. The "same parameter value optimizer" substitutes

Table 6: Dead code elimination grouped in different steps

dead code elimination step	**examples**

(1) updating logical expressions	if (true && isWhatever()) -> if (isWhatever())
	if (false && isWhatever()) -> if (false)
	if (true || isWhatever()) -> if (true)
	if (false || isWhatever()) -> if (isWhatever())
	null == null -> true
	if(!a == b) -> if(a!=b)
	null != null -> false

(2) shorten xor expressions	true ^ x -> !x false ^ x -> x y ^ true -> !y y ^ false -> y
(3) shorten numerical expressions	j+0 -> j, 0+j -> j, j-0 -> j, 0-j -> -j j*1 -> j, 1*j -> j, -1*j -> -j, j*-1 -> -j j*0 -> 0, 0*j -> 0, j/1 -> j, j/(-1) -> -j j « 0 -> j, j » 0 -> j, j »> 0 -> j
(4) shorten assignments	a = 3, b = 3 -> a = b = 3
(5) concat strings	s = "Hallo " + "Tom" -> s = "Hallo Tom"
(6) removing blocks with no effect	if(x == 0) { y = 0; } -> if(x == 0) y = 0;

(7) removing useless loops	do { ... } while(false); -> { ... }
	for (X; false; Y) {...} -> Y;
	while(false) { ... } -> {}

(8) remove useless and shorten if code	if(true) x=3; else x=4; -> x=3;
	if(false) x=3; else x=4; -> x = 4;
	(cond ? true : else) -> cond || else
	(cond ? false : else) -> !cond && else
	(cond ? then : true) -> !cond || then
	(cond ? then : false) -> cond && then
	(!cond ? then : else) -> (cond ? else : then)
	if (!cond) foo else bar -> if (cond) bar else foo
	if(x) return 3; else return 4; -> return x ? 3 : 4;
	if (x) s1 else s2 -> x ? s1 : s2;
	if (x) s1 -> x && s1;

(9) replacing post operator with pre operator if possible	x++ -> ++x, x– -> –x
(10) optimize switch blocks	switch(x) { case 1: {} case 2: {} case 3: { s1 } } -> switch(x) {case 3: { s1 } }
(11) deleting catch blocks whose exception type will never be thrown and prune empty try statements	
(12) static evaluation of literals	s = 3+8 -> s = 11 s = true && false -> s = false -(-x) -> x

the method's parameters with literals, which are used in method calls in the function body. Listing 32 shows an example code before this optimization phase and listing 33 after. In line 3 the true literal substituted the log parameter, because the execute function was always called (see lines 8 and 9) with log equals true. The useless parameter log and the if condition will be deleted later. This is the reason why the optimizeLoop is called several times as mentioned above.

```
1 void execute(...,boolean log) {
2   ...
3   if(log) {
4     logger.log(...);
5   }
6 }
7 ...
8 execute(a,b,...,true);
9 execute(x,y,...,true);
```

Listing 32: Same parameter value optimizer: **before**

```
  void execute(...,boolean log) {
    ...
    if(true) {
      logger.log(...);
    }
  }
  ...
  execute(a,b,...,true);
  execute(x,y,...,true);
```

Listing 33: Same parameter value optimizer: **after**

The "enum ordinalizer" phase replaces enum constants with the corresponding ordinal values by changing their field constants to integer variables. The enum class will be removed later. Listing 34 displays the code before this optimization phase and listing 35 (compare lines 7 and 8) afterwards.

```
1 public enum DAY {
2   SUNDAY, MONDAY, TUESDAY,
3   WEDNESDAY, THURSDAY,
4   FRIDAY, SATURDAY
5 }
6 public static void main(String
      []as){
7   Day firstDay = new EnumTest(
        DAY.MONDAY);
8   Day thirdDay = new EnumTest(
        DAY.WEDNESDAY);
9 }
```

Listing 34: Enum ordinalizer optimizer: **before**

```
public enum DAY {
  SUNDAY, MONDAY, TUESDAY,
  WEDNESDAY, THURSDAY,
  FRIDAY, SATURDAY
}
public static void main(String[]
    as){
  Day firstDay = new EnumTest(1);
       // Day.SUNDAY = 0
  Day thirdDay = new EnumTest(3);

}
```

Listing 35: Enum ordinalizer optimizer: **after**

After the optimizations in the optimizeLoop have been explained, the next paragraphs describe the dataflow optimizer. 'Dataflow optimization is the code improvement by transformations which are justified by assertions starting with the weakest one (true) and computing strongest postconditions. Listing 36 shows constant propagation and folding done with dataflow analysis.

```
          { true }
x = 9;
          { x == 9 }
y = 3:
          { x == 9 && y == 3 }
z = x - y;
```

Listing 36: Simple dataflow analysis example

This means in the assignment z = x - y the assertion x == 9 justifies replacing x with 9, this means z =9 - y. The other assertion y = 3 justifies substituting 3 for y, as the result we are getting the assignment z = 6. This means the aim of dataflow optimization is to find valid assertions allowing code improving transformations. Assertions are characterized as dataflow facts relating to traditional program logic. A data flow fact is

either a program state or continuation assertion. A lattice is a set of dataflow facts. To ensure that the analysis ends, it is enough if every fact has only finitely different facts above it. Mostly a different lattice of dataflow facts is used by different analyses.

A continuation assertion is an assertion about paths from a specific program point to a procedure end, or some other kind of ending; also known as backward dataflow analysis. Information about forward dataflow analysis can be found at [NR08, p. 3]. An edge in a control flow graph represents a program point. Edges connect nodes representing an assignment, a label or a control transfer. A transfer function computes dataflow facts on outgoing edges of a node out of incoming edges.' [NR08, pp. 1-3] The aim of this short introduction to dataflow optimization is to allow a better understanding of the graph interface (public interface Graph<NodeType, EdgeType, TransformerType>) and the integrated analysis interface (public interface IntegratedAnalysis<NodeType, EdgeType, TransformerType, G extends Graph<NodeType, EdgeType, TransformerType>, AssumptionType extends Assumption< AssumptionType>).

The integrated analysis interface contains a reference to the integrated flow function, which either interprets the node or produces node transformations based on already computed assumptions. The already computed assumptions are stored in the interface as map (AssumptionMap) combining assumption and edge types; these assumptions are quasi the input and output of the integrated flow function. To give an example, listing 37 shows the source code of unreachable analysis.

```
public class UnreachableAnalysis implements IntegratedAnalysis<
    CfgNode<?>,
     CfgEdge, CfgTransformer, Cfg, UnreachableAssumptions> {
  private static final UnreachabeIntegratedTransformationFunction
  INTEGRATED_FLOW_FUNCTION=new
      UnreachabeIntegratedTransformationFunction();

  public IntegratedFlowFunction<CfgNode<?>, CfgEdge,
      CfgTransformer, Cfg,
                                  UnreachableAssumptions>
  getIntegratedFlowFunction() {
    return INTEGRATED_FLOW_FUNCTION;
  }

  public void setInitialGraphAssumptions(Cfg graph,
       AssumptionMap<CfgEdge, UnreachableAssumptions> assumptionMap
           ) {
    AssumptionUtil.setAssumptions(graph.getGraphInEdges(),
        UnreachableAssumptions.REACHABLE, assumptionMap);

    AssumptionUtil.setAssumptions(graph.getGraphOutEdges(),
        UnreachableAssumptions.UNREACHABLE, assumptionMap);
  }
}
```

Listing 37: Source code of UnreachableAnalysis.class in GWT Software Development Kit (SDK) 2.4 in gwt-dev.jar (package com.google.gwt.dev.jjs.impl.gflow.unreachable)

Cfg is the abbreviation for control flow graph. The Cfg class implements the Graph class and is the control flow representation for the entire gflow framework. This class contains array lists for the nodes as well as edges coming in and going out of the graph. It also contains a transformer function changing the AST structure of this graph. CfgNode is the class type of the nodes in the Cfg class. This class has two collections: one for the edges, which are coming into the node, and

one for the edges which are going out. The template parameter of this class extends the JNode type, which is the base class for all GWT AST nodes. For general understanding, it is enough to say that the CfgNodes of the Cfg graph are a subset of all GWT AST nodes. CfgEdge is the class type of the edges in the Cfg class containing a start node and an end node of one edge. The UnreachableAssumptions implements the Assumption<UnreachableAssumptions> interface and acts like an enum class having two fields REACHABLE and UNREACHABLE. The actual implementation of the transformation can be found in the UnreachabeIntegratedTransformationFunction class. Listing 38 shows the source code of this class. With the

```
1  public class UnreachabeIntegratedTransformationFunction implements
2      IntegratedFlowFunction<CfgNode<?>, CfgEdge, CfgTransformer,
           Cfg,
3                              UnreachableAssumptions> {
4    public Transformation<CfgTransformer, Cfg>
5    interpretOrReplace(CfgNode<?> node, Cfg graph,
6        AssumptionMap<CfgEdge, UnreachableAssumptions> assumptionMap
             ) {
7      UnreachableAssumptions in = AssumptionUtil.join(
8          graph.getInEdges(node), assumptionMap);
9
10     if (UnreachableAssumptions.isReachable(in)) {
11       AssumptionUtil.setAssumptions(graph.getOutEdges(node),
             UnreachableAssumptions.REACHABLE, assumptionMap);
12       return null;
13     }
14
15     if (node instanceof CfgNopNode) {
16       AssumptionUtil.setAssumptions(graph.getOutEdges(node),
             UnreachableAssumptions.UNREACHABLE, assumptionMap);
17       return null;
18     }
19
20     return new DeleteNodeTransformation(graph, node);
21   }
22 }
```

Listing 38: Source code of UnreachabeIntegratedTransformationFunction.class in GWT SDK 2.4 in gwt-dev.jar (package com.google.gwt.dev.jjs.impl.gflow.unreachable)

join operation in lines 7 and 8 the maximum of all edges coming into this node are calculated ($\max(\text{REACHABLE}, \text{UNREACHABLE}) = \text{REACHABLE}$). If the input maximum is reachable, then all output edges are reachable (lines 10 until 13). If the node carries out no operation, e.g. containing an empty body, then all output edges are unreachable (lines 15 to 18). If the maximum is unreachable and the node is not a no-operation node, then the DeleteNodeTransformation class tries to delete unused statements in line 20. The DataflowOptimizer class executes ConstantsAnalysis, CopyAnalysis and LivenessAnalysis besides UnreachableAnalysis.

Constant analysis replaces constant local variables and constant parameters; constant propagation and folding (see listing 36) are parts of this analysis. Copy analysis works are similar to constant analysis, except that it is not allowed for variables to be equal in assertions. Listing 39 shows such an example. This means the compiler will replace z=5+y by z=5+x and will delete the useless assignment y=x later. The analysis is called so, because the compiler recognizes that one variable is just a copy of another one and so the optimizer will substitute the copy with the original variable. In liveness analysis the compiler calculates at each program point the life of

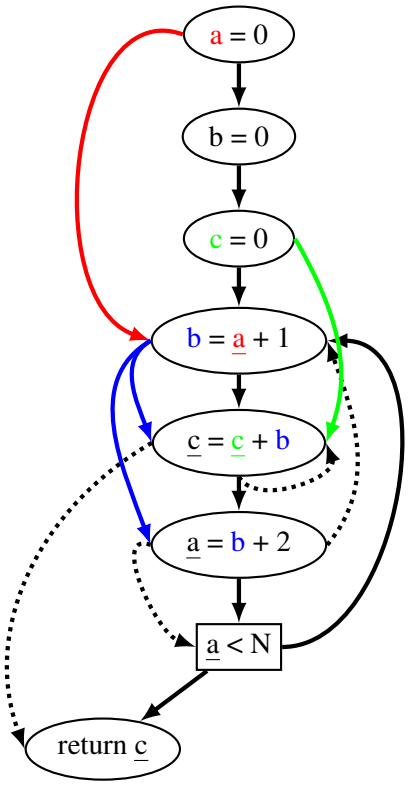


Figure 10: Control flow graph of liveness analysis

```
                { true }
y = x
                { y == x }
z = 5 + y
```

Listing 39: Simple copy analysis example

```
1       a = 0
2       b = 0
3       c = 0
4  L1 : b = a + 1
5       c = c + b
6       a = b + 2
7       if a < N goto L1
8       return c
```

Listing 40: Simple example code for liveness analysis

variables, which is the timespan where a variable may be potentially read before its content will change. The life of variables is used to remove useless assignments. This analysis is a backward dataflow. To get a better understanding listing 40 shows an example and figure 10 the corresponding control flow graph. Since the node b=0 contains no lines to other nodes, the lifespan of the variable b in line 2 is zero. As a result the optimizer will delete this node. The variables a and c have at least two life ranges, e.g. the first lifespan of a is from line 1 to line 4 and the second one is from line 6 to line 7 or line 4 (if it is extended by the loop). Any executed jump creates a new live range of the variable a in line 6[18]. The demonstrated liveness example is a modification from [Pro04, pp. 4-5]. Exact definitions and an algorithm to determine the liveness of variables can be found at [Pro04, pp. 7-13].

As mentioned at the beginning of this paragraph, most of the optimizations are done in the optimize function. But one more Java optimization step, which removes all calls to super constructors with empty bodies, is executed in the compilePermutation function after the optimize function has been called.

After explaining the Java AST optimization steps in great detail, the JavaScript AST optimizations will be only touched on. Firstly the JsStaticEval (JavaScript static evaluation executor) will do the same optimizations on JavaScript code as the dead code elimination step (see table 6) does on Java code. Secondly, the JsInliner optimization is running and it can be compared with the MethodInliner for Java code. Lastly the JsUnusedFunctionRemover will delete JavaScript methods, which are not referenced in the program. All the optimizations mentioned can be found in the optmizeJs function in the JavaToJavaScriptCompiler class.

The explanation of the GWT compiler steps is finished, it is now known what the compiler does, why the compilation process does several permutations and why compiling a GWT ap-

[18] If it is not obvious at the first look, think a is an array then the code in listing 40 would be: a[0] = 0; index=1 (line 1), b = a[index-1] + 1 (line 4), a[index++] = b + 2 (line 6). And so a[0] until a[array length] are the different live ranges.

plication needs so much time. Testing a GWT application can also be done in the developer mode, which does not invoke the GWT compiler, in order to save time (for more information see appendix A 1).

3.4 Java Runtime Environment Emulation

The Java Runtime Environment Emulation (JREE) is a library emulating the most commonly used part of the real Java Runtime Environment (JRE). The emulated java source files can be found in the gwt-user.jar library at com.google.gwt.emul. Since the package explorer shows only *.class files of java archives, the gwt-user.jar[19] archive must be opened with WinRAR or some similar program in order to change them. Figures A43, A44 and A45 show the UML diagrams of the JREE. They list all the JREE classes of GWT 2.4, because it is useful to know what classes are allowed to be used. The tricky thing is, that the development mode uses the standard Java JRE, because the code will be interpreted by the GWT code server and will not be compiled using the GWT JREE. As a result, JRE classes, which do not belong to the JREE, can be used in development mode. These classes like java.util.GregorianCalendar work in the web application running in the development mode the same way as they would work in a normal Java program. Listing 41 and figure 11 show an example of this. Only an error message like "Errors in 'file:/C:/GWT/workspace/DA_JREECalendar/src/de/tu_freiberg/informatik/vonwenckstern/client/Today.java': [ERROR] [de.tu_freiberg.informatik.vonwenckstern.ImageViewer] - Line 8: No source code is available for type java.util.GregorianCalendar; did you forget to inherit a required module?" indicates that the web application does not use a JREE class.

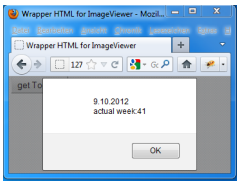


Figure 11: Simple example code for using no JREE class

```
20  Window.alert(Today.getToday());
6
7   public static String getToday() {
8     GregorianCalendar calendar = new GregorianCalendar();
9     String s = calendar.get(Calendar.DAY_OF_MONTH) + "." +
10                Integer.toString(calendar.get(Calendar.MONTH) + 1)
                      + "." +
11                calendar.get(Calendar.YEAR);
12    s += "\nactual␣week:" + calendar.get(Calendar.WEEK_OF_YEAR);
13    return s;
14  }
```

Listing 41: Simple example code for using no JREE class
See listing A27 and A28 for the complete source code.

Ignoring this error causes GWT to abort the compilation process. Since the compiler does not have any source code for the GregorianCalendar class, it will show an error list containing a lot of unresolved fields and methods of this class. The JREE source files show, that the basic code is written with JSNI (see section 3.3.1 for more information about JSNI, JavaScriptObject and GWT.create). This is why the Java JRE functions sometimes behave a little differently to the compiled JavaScript JREE functions. This phenomenon will be explained at the JSNI function String::split(String regex, int maxMatch). This means that the regular expression parameter will be interpreted in the web browser as JavaScript regular expression, since JSNI replaces Java regex calls with JavaScript ones. The problem is that the Java regular expression is slightly

[19] If GWT has been installed via the GWT plugin as shown in appendix A 1, this file is located under <eclipse>/plugins/com.google.gwt.eclipse.sdkbundle_2.4.0.vxxx/gwt-2.4.0/ gwt-user.jar

different from the JavaScript one. The two online regular expression tester [Goy] for JavaScript and [Reg] for Java help to figure out the difference.

The next paragraph explains the main differences. The ^ sign means in Java that a special class will be negated in order to match; in JavaScript it signifies that the string must match the regular expression from the beginning on. Java supports quote matching with \Q ... \E, but JavaScript does not. Java matches strings only if they begin with the first regular expression character and end with the last one. In JavaScript this behavior can be forced by starting the regular expression with ^ and ending it with $. It is advisable to avoid JavaScript specific regular expressions like $', which allow the replacement of characters, to have the same result in compiled and debug mode. Table 7 compares Java and JavaScript regular expressions.

Table 7: Examples to show the difference of String::match in the Java vs. JavaScript world.

Regular expression	Input string	Java matches	JavaScript match
\d+	a123b	no	yes
\d+	123	yes	yes
^\d+	a123b	no	no
^\d+	123b	no	yes
^\d+$	123b	no	no

Listing 42 displays a quote function for regular expressions working on both Java and JavaScript. The dissimilarities between the JRE and JREE have been explained in more detail

```
1 public static String quoteStringForRegex(String str) {
2   String res = "";
3   for(int i=0; i<str.length(); i++) {
4     String hex = Integer.toHexString(str.charAt(i));
5     res = res + "\\x" + (hex.length() < 2? "0":"") + hex;
6   }
7   return res;
8 }
```

Listing 42: Source code of a quote function for regular expressions in JS and Java
e.g. "\x64" means there should be used the sign with the ASCII hex code 64, which is "d"

to help understand some irregular occurrences during the GWT development.

The above paragraph could create the impression that the JREE classes are not very helpful, because of the trouble which can occur. But an active Java developer will be pleased to use the String, ArrayList and HashMap classes, because in most cases they work the same way as in the JRE. [Ros03] gives a tutorial on how to emulate missing JRE classes in GWT.

The answer to the question whether GWT provides the most important Java classes can only be a partial yes. The main reason why the question is not answered with a clear yes is that some very important classes like the Calendar emulation are missing and this results in an extensive usage of the depreciated Date class.

3.5 Widgets and Panels

3.5.1 Overview of GWT Widgets

The showcase in subsection 3.1 introduced the most important GWT widgets. This subsection gives a hierarchical overview of the most relevant GWT widgets. The next subsection describes how these components handle user inputs.

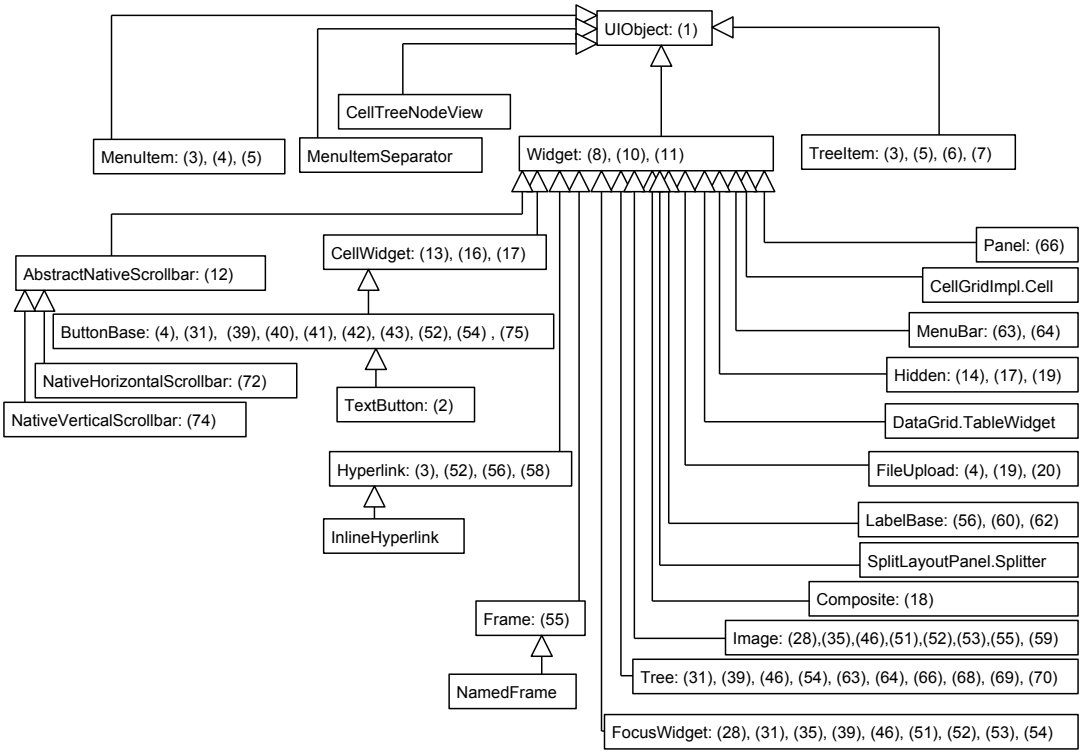


Figure 12: Widget class hierarchy; source: API of GWT 2.5.0 RC 2

The book GWT in Action gives a nice definition for widgets: "Widgets are the visible components of a GWT application that a user can see on the browser page." [HT07, p. 110]. Widgets are the analogue to the Java Swing user interface components like JButton or JTextField. A list of all GWT widgets can be found at [Goo10d]. As illustrated in figure 12, all widgets inherit from the UIObject. This class wraps HTML DOM properties (see section 2.5 for information) in such a way, that they can be manipulated in Java code. The most important methods of the UIObject class are addStyleName(String) to add a new CSS class to this widget, getAbsoluteLeft and getAbsoluteTop to get the absolute position in pixels of the element in the web browser, getOffsetHeight and getOffsetWidth returns the size in pixels of the displayed object, isVisible tells whether the user can see the widget, and the getElement method handles the corresponding DOM object to manipulate the HTML DOM directly by adding or removing DOM nodes. More information about modifying the HTML DOM with GWT can be found in the API of the com.google.gwt.dom.client.Node class at [Goo10c].

Since there is no information about the operations of a class during JavaScript runtime, the widgets implement many interfaces describing the methods of the widgets. A check with Java's instanceof operator tells whether the widget contains a specific method. For example the Panel widget implements the HasWidgets.ForIsWidget interface, that means it supports the operations add and remove to add or delete widgets. The numbers behind the classes in figure 12 represent the interfaces, which the widgets implement. The corresponding interface with all important methods can be found in figures A46 and 19. The last image contains the events which support the widget. GWT events and the corresponding handler interfaces will be discussed in the next subsection.

The last picture shows that the Widget class extends the UIObject and implements EventListener and the HasAttachHandlers interface to support low-level browser events and notifications. These are fired when the object is being attached or detached to the browser's document. The MenuItem class generates a unique id in the document DOM model in order to support

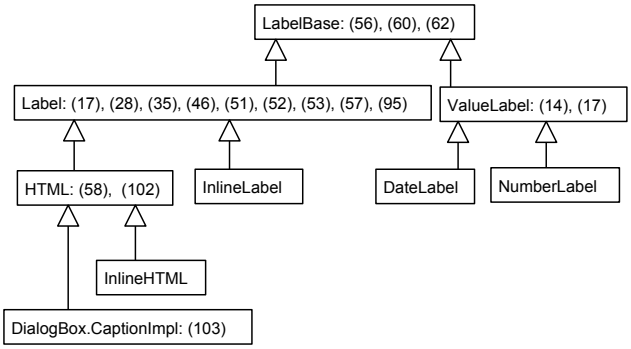


Figure 13: Label widget class hierarchy; source: API of GWT 2.5.0 RC 2

ARIA (Annotations for assistive technology products). It is not a widget, because creating the entire menu of the given menu items and handling the events is done by the MenuBar widget.

The next paragraph gives more details about the CellWidget. As mentioned before, normal widgets extend the UIObject class, which means all the widget properties are set by DOM manipulation. And cell widgets are "high-performance, lightweight widgets composed of Cells for displaying data" [Goo10d]. These widgets are used in tables, trees and lists. The rendering process is so fast, because it generates a HTML string with all the properties of the widgets and their children. This generated string is set to the DOM model's property innerHTML. Instead of using var div = document.createElement('div'); div.width = 200; var span=document.createElement('span'); span.className = 'red'; span.appendChild(document.createTextNode('Error')); div.appendChild(span); document.getElementById(parent).appendChild(div), which needs seven DOM manipulations, cell widgets need only one DOM manipulation with the following code var html = " Error "; document.getElementById(parent).innerHTML = html. The CellWidget class is a wrapper widget, which takes a lightweighted cell widget and turns it into a normal widget. This way cell widgets can be reused outside a CellTable. This class can also be used to create widgets which the browser can render quickly. This is the reason why some class names like ButtonBase are listed twice in the UML diagrams. The ButtonBase class extending the CellWidget generates light weighted buttons, and the ButtonBase class extending the FocusWidget produces heavy weighted DOM buttons with more properties.

The standard GWT widgets can be divided into four types: Label, FocusWidget, Composite and Panel. Figure 13 shows the hierarchy of the LabelBase class. Label widgets display information to the user. The Label class becomes a tag in the HTML document and the InlineLabel class becomes a tag.

FocusWidgets (figure 14 shows the hierarchy) are widgets which can receive user inputs. Nearly all of them are used in forms to collect user information. Most of the widgets are Java classes for different HTML <input> types. An example should show what pages can be created with these widgets. Listing A29 and figure A47 display the complete example.

The RichTextArea widget shows formatted (bold, underlined, italic, colored) text. The method setHTML(String) sets the styled text. In this example the HTML string "<u>GWT</u> stands for Google Web Toolkit." was passed as parameter. This results in the following output: "**GWT** stands for *G*oogle *W*eb *T*oolkit.", The tags u, b, i are abbreviations of underlined, bold and italics. The RadioButton widget takes as constructor two parameters: the first one is the group name and the second one is the text displayed after the radio button. Since this class extends CheckBox implementing the HasValue<Boolean> interface, the function setValue(true) selects the button. The PushButton widget accepts as parameter either a string or an image object. If a string value is passed, then

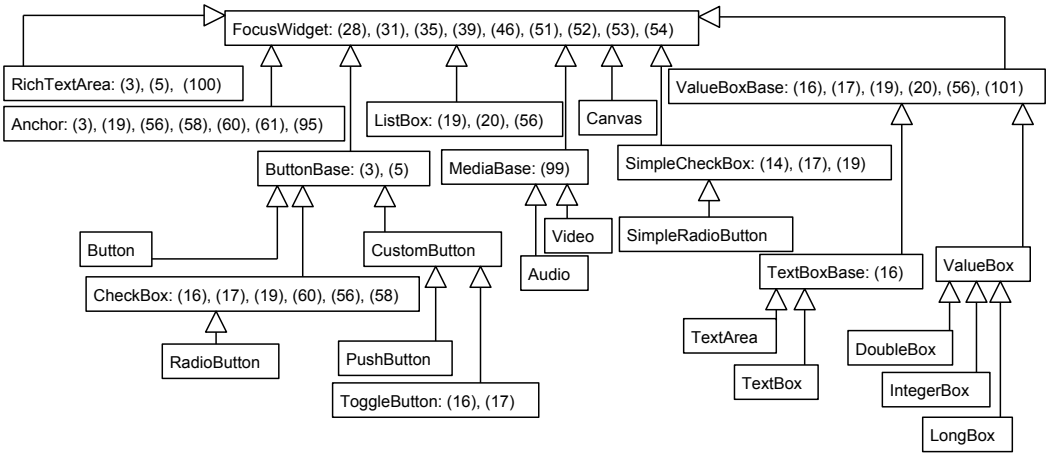


Figure 14: FocusWidget widget class hierarchy; source: API of GWT 2.5.0 RC 2

the button displays normal text; but otherwise it draws an icon. In this example the national flag of Germany: Image im = new Image("http://upload.wikimedia.org/wikipedia/...); new PushButton(im) was passed as parameter. The most important method of the ListBox widget is addItem to add new selectable options. The ListBox appears as drop-down list, if the parameter 1 is set to the function setVisibleItemCount(int); otherwise it displays the number of visible items. The classes Audio, Video and Canvas are created by calling the static method createIfSupported,as older web browsers do not support the new HTML5 tags. If the browser does not support the new HTML elements, then this function will return null. Both Audio and Video widgets support setting the source by either calling the setSrc(String) or the addSource(String, String) method. The first function should only be invoked if every browser (Chrome, Firefox, Internet Explorer, Safari, ...) supports the specified file type. Otherwise the second method should be used, because it allows the specification of different file formats. This gives the browser the opportunity to select the URL to the media type, which it supports. The media control bar, which allows the user to start, stop and pause the audio or video playback, can be enabled by invoking the method setControls(true). After the Canvas object has been initialized, the line Context2d context = canvas.getContext2d() creates the draw context. A function call similar to context.setFillStyle(CssColor.make(255, 0, 0).value()) sets the drawing color of text and different shapes. The method fillRect(int, int, int, int) draws a rectangle and takes the start point as the first two parameters and the size as the last two . The function fillText(String, int, int) prints text to the draw context and accepts the text as the first and the position as the last two parameters.

The difference between the TextArea and the TextBox is that the first one accepts line breaks and the last one does not. This means the TextArea widget allows the user to input multiple lines.

Composite widgets can wrap one or more widgets. Figure 15 shows the DateBox widget having a TextBox in the top and a PopupPanel, which contains a DatePicker, on the bottom. GWT ships many standard composite widgets to the developer. Figure 16 displays the hierarchy. The example shown in figures A48, A49 and listing A30 introduces these widgets. The NotificationMole class creates a popup information display. The information shown to the user is set by the method setMessage(String). If a number greater zero has been set to setAnimationDuration(int), then the popup needs so much time in milliseconds to show up completely. During the appearance process the information will become visible from top to bottom. CaptionPanel is a normal panel, which contains other widgets with a HTML header text, which is the first parameter in the constructor. DateBox (see figure 15) works like a ValueBoxBase widget, which

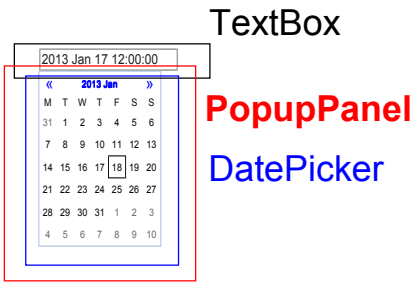


Figure 15: DateBox composite contains TextBox, PopupPanel and DatePicker widgets

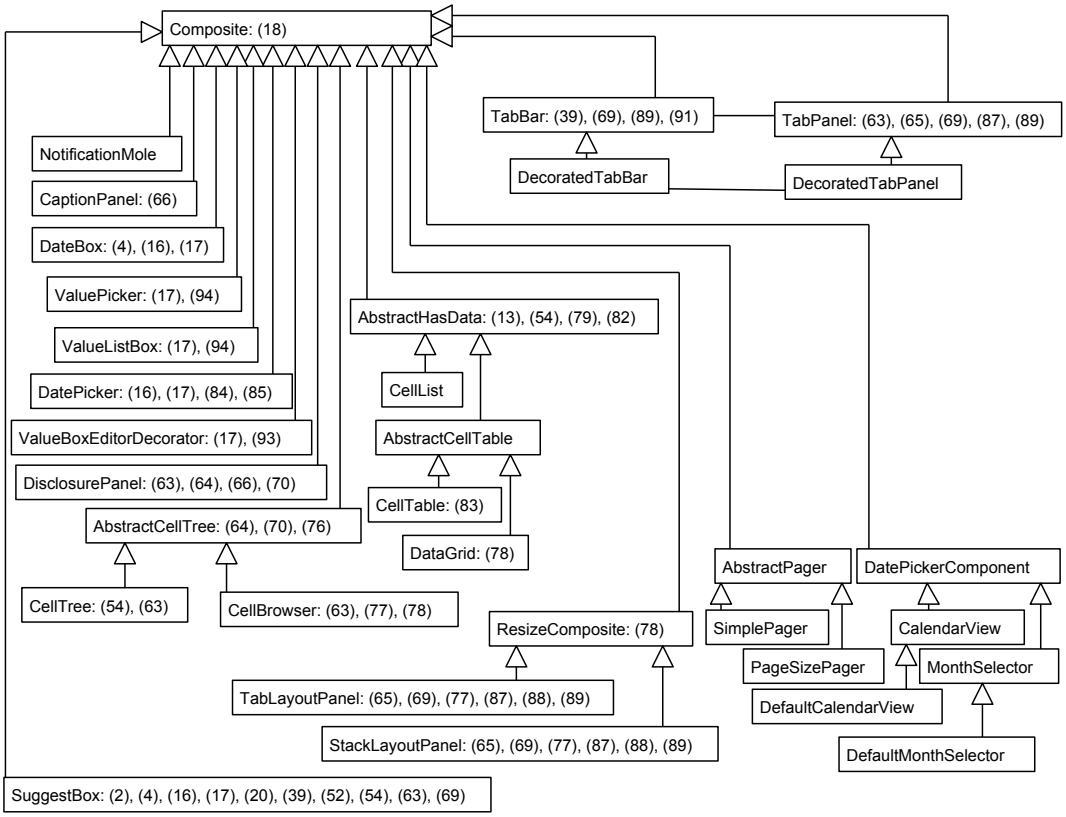


Figure 16: Composite widget class hierarchy; source: API of GWT 2.5.0 RC 2

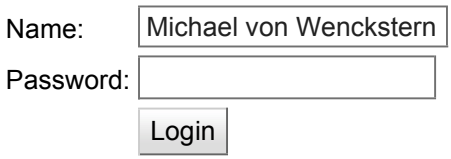


Figure 17: The own Composite widget

lets the user enter a date value. ValuePicker is comparable to the non-dropdown ListBox. The only difference is that the operation setAcceptableValues(Collection) sets the possible values. ValueListBox is similar to the dropdown ListBox. But both widgets have an advantage over the normal list box, because it is possible to specify a value renderer interpreting the list data. The DisclosurePanel is a panel with a header, which shows and hides its body content after the user has clicked on the header. As default the body part is not visible; passing true to the setOpen(boolean) method makes the body visible. The CellTree works like a Windows Explorer tree. It shows the parent's children items after clicking on it. This class needs as first constructor parameter a TreeViewModel containing two methods getNodeInfo(T) and isLeaf(Object). The first function returns the children items of a given parent item and the other one returns true, if the parent item has no children to display. CellTable widget is used to present large data to the user in a table form. The addColumn(Column, String) operation adds a table column to this object. The first parameter object needs to implement the getValue(T) method of the abstract Column class, which returns the value contained in this column from a given record data. After all the table columns have been defined, the row data (=different records) can be set to the table. This is done by invoking the table's setRowData method, and passing the data as first argument. The CellList widget displays its data as list. The SimplePager object can control the data range which is displayed in the web browser, of either the CellList or the CellTable widget. The connection between the pager and the cell widget is done by passing the cell widget as argument to the pager's setDisplay(HasRows) function. The StackPanel shows all the children headers but only one body widget at once. Similar to the DisclosurePanel, the childrens' body part can be made visible by clicking on its header. The TabPanel works the same way as the StackPanel. The only difference is the appearance: the TabPanels headers are horizontally aligned and always above or below the selected body part. In contrast the StackPanel's headers are vertically aligned and the visible body part can be located between two headers. The SuggestBox auto completes words while their initial letters will be typed. The MultiWordSuggestOracle contains the recommendation for the auto complete and is the only constructor parameter of this widget.

```
1  class LoginComposite extends Composite {
2    private TextBox name = new TextBox();
3    private PasswordTextBox pwd = new PasswordTextBox();
4    private Button btn = new Button("Login");
5
6    public LoginComposite(String defaultName) {
7      name.setValue(defaultName);
8
9      FlexTable tbl = new FlexTable();
10     tbl.setText(0, 0, "Name:␣");
11     tbl.setWidget(0, 1, name);
12     tbl.setText(1, 0, "Password:␣");
13     tbl.setWidget(1, 1, pwd);
14     tbl.setWidget(2,1, btn);
15
16     // initWidget must be always in constructor
17     initWidget(tbl);
18   }
19 }
```

Listing 43: Example project for own login composite widget

Listing 43 and figure 17 demonstrate how to create a login composite widget. It contains a TextBox, one PasswordTextBox and a Button widget. Firstly, the widget extends the Com-

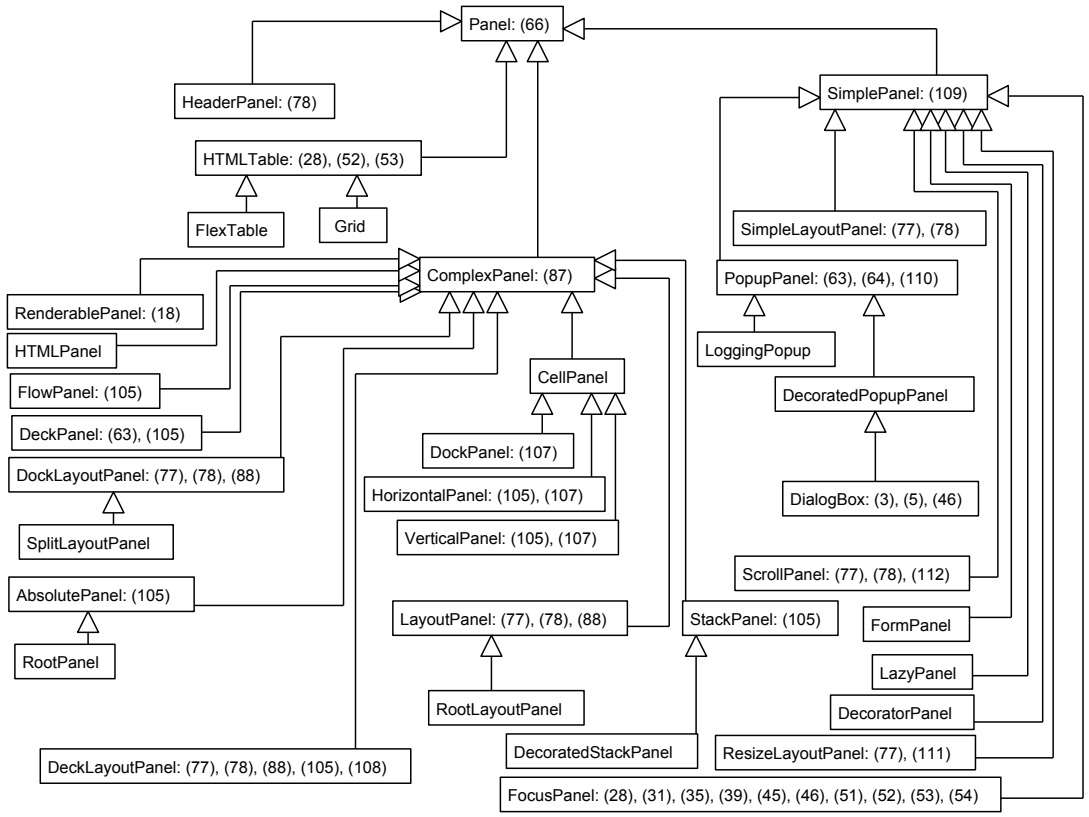


Figure 18: Panel widget class hierarchy; source: API of GWT 2.5.0 RC 2

posite object. Secondly, the widget view will be built by adding several other widgets to the table in the same way they have been added to any application view. Despite adding the FlexTable to the RootPanel, it will be inserted into the composite widget by calling the initWidget(Widget) method inside the composite's constructor.

Panels are widgets containing one or more other widgets. These are used to structure the layout of a website. Figure 18 gives an overview of the different panel types. As always this paragraph starts with an example (see listing A31). Figure A50 illustrates the Panel widgets in action. FlexTable user interface component creates a table supporting cell merging. New cell data can be added to the object by either calling setText(int row,int column, String text) or setWidget(int row,int column, Widget w). The code line DOM.setStyleAttribute (formatter.getElement(row, col), "border", "1px solid black") makes the cell lines hidden by default visible. After the cells distance has been set to zero by calling setCellPadding(0) and setCellSpacing(0), a nice table grid has been created. The FlexTable does not really support cell merging, but it can be emulated by:

- displaying one cell over several rows and columns. The methods setRowSpan(int row, int column, int rowSpan) and setColSpan(int row, int column, int colSpan) of the table's FlexCellFormatter class (FlexTable.getFlexCellFormatter()) will do this work.
- removing the cells, which should become merged together with the expanded ones. This is done by calling the table's removeCell(int row, int column) method.

The difference of the HeaderPanel widget to the CaptionPanel one is that the first one allows setting a header widget to the header part instead of only plain HTML text. The header widget may even contain other widgets. The HeaderPanel also allows setting a footer widget. The

advantage of the HTMLPanel is that the container layout can be set directly as constructor argument. Afterwards the widgets will be inserted into given div-id place inside the container. This is done by calling HTMLPanel's add(Widget widget, String divId) method. The FlowPanel just creates a environment and inserts every added widget into it, this means the floating layout is handled by the web browsers like line breaking and overflowing. DockLayoutPanel and DockPanel behave the same, the only difference is that the DockLayoutPanel generates its layout by using HTML div elements and the DockPanel uses a HTML table. The functions addNorth(Widget widget, double size), addSouth(Widget widget, double size), addWest(Widget widget, double size), addEast(Widget widget, double size) or add(Widget) add the widgets into the upper, lower, left, right or center part of DockLayoutPanel. The add(Widget widget, DockPanel.DockLayoutConstant direction) method inserts the widget into the DockPanel. The second parameter controls the place where new widgets will be inserted. The HorizontalPanel widget adjusts its children in a line next to each other. The VerticalPanel places its children in a column one below the other. ScrollPanel widget automatically creates scrollbars, if the content of its child widget is larger than the panel's screen size. FormPanel component wraps its child into a HTML <form></form> tag. The method submit sends the formula data to the URL specified as parameter in the setAction(String URL) function. DialogBox widget creates an information window, which can be moved around in the HTML website. The method setText(String text) sets the displayed text. A Button or any other widget can be added to the dialog by calling the setWidget(Widget) function. If the DialogBox should act like a modal window, then true has to be passed to the setGlassEnabled(boolean) method.

This section showed that there are a lot of GWT widgets. These ones, together with the widgets of other third party libraries like the GXT one, are an equivalent substitute for every Swing user interface component. Since GWT does not compile byte code to JavaScript, the source code for every widget is available. That makes it easier to create customized widgets by extending the existing ones. This subsection answered one of the two important questions as to whether the GUI components in GWT are equivalent to the Swing ones in a clearly positive way.

3.5.2 Event handlers in GWT Widgets

Event handling in GWT works very similarly to event listening in Swing. Both of them use the delegation model:

1. Interested listeners register their class to the object, which should trigger the event. This causes the listener to be added to a subscription list.
2. When the state has changed, the source class generates and fires an event containing the actual state.
3. Now every listener in the subscription list for this event type receives the event. The listeners are called synchronically one after another.
4. Finally the implementation method of the registered class is invoked, and so the listener can start processing the event.

To receive events in GWT the following things have to be done concretely:

1. Implement the handler interface for the event, which should be received.
2. Pass the handler reference to itself or to the widget to add it to the subscription list of the given event type.

Figure 19 shows the most important event handlers, which should be overridden in order to receive the events. Listing 44 illustrates that event handling can be done very intuitively in

```
Button btn = new Button("Click␣Me");
btn.addClickHandler(new ClickHandler() {
  public void onClick(ClickEvent event) {
    // handle the click event
  }
});
```

Listing 44: Very simple example of registering a ClickHandler to receive Button click events

GWT. On the surface many anonymous inner classes could be used to receive all the click events from several buttons. However this would result in a creation of a large number of separate ClickHandler objects and the result would be a need of too much memory. The better version is to create just one ClickHandler instance and add this object to every Buttons' subscription list. Since every GWT event extends com.google.gwt.event.shared.GwtEvent<H extends EventHandler> it inherits the getSource method, which returns the widget which fired the event. Listing 45 displays the correct Java code for handling click events of many buttons.

```
Button btn1,...,btn99;
final ClickHandler clickHandler = (new ClickHandler() {
  public void onClick(ClickEvent event) {
    // we now that we registered this handler only to
    // Button classes
    Button sender = (Button) event.getSource();
    if(sender == btn1) {
      // process ClickEvent from btn1
    } ... { else if(sender == btn99) {
      // process ClickEvent from btn99
    }
  }
};
btn1 = new Button("Button␣1");
btn1.addClickHandler(clickHandler);
...
btn99 = new Button("Button␣99");
btn99.addClickHandler(clickHandler);
```

Listing 45: Example how to handle click events of many buttons

Like the more complex EventHandler example in listing A32 and figure A51 the class containing the buttons implements the ClickHandler interface and the event registration is done by calling addClickHandler(this).

Some components like the Label widgets do not fire keyboard events. In this case the widget has to be wrapped inside the FocusPanel, and this panel fires keyboard events. A widget will only fire the keyboard if it contains the input focus.

This subsection finishes by answering the second important question as to whether the event handling is as powerful as in Swing. Even through GWT event handling is very similar to the Swing event listening, it cannot handle as many events as the Java Swing library. This limitation is the result of the stricter security issues which a JavaScript program has to deal with in comparison to a normal desktop application. For example a web application does not have any access to the Windows clipboard, and so the data must be extracted from a contenteditable div element. Section 7 shows how to write data to the clipboard and how to read pasted content. Another drawback of JavaScript and GWT is that no drag and drop download event exists. It

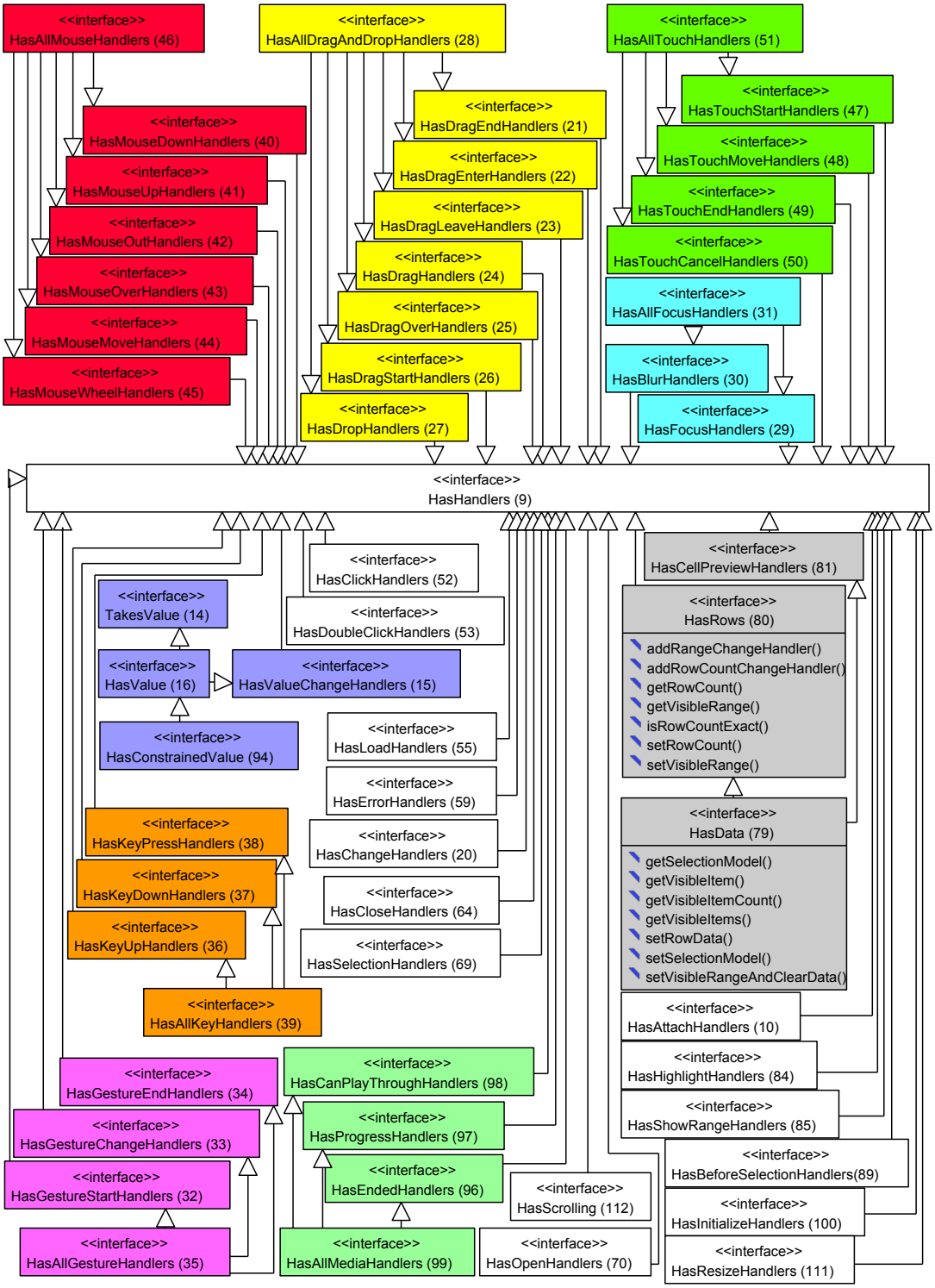


Figure 19: General handler interfaces which implement the widgets to listen to special events; source: API of GWT 2.5.0 RC 2

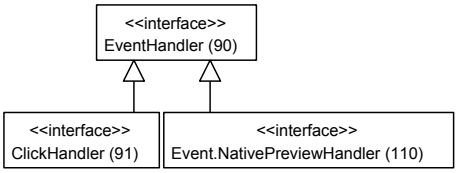


Figure 20: Special handler interfaces which implement the widgets to listen to special events; source: API of GWT 2.5.0 RC 2

is possible to upload files per drag and drop, but until now files cannot be downloaded per drag and drop. Without the security and not available JavaScript functionality issues, the second important question can be answered with yes. But if the web application needs to write binary data into the clipboard, if it needs hard disk access, or if the user should be able to download several files at once, then there is no equivalent GWT functionality available. This means in such a case it is not possible to create a web application which has the same user comfort as the equivalent desktop program.

3.5.3 Manipulating browser's DOM with GWT DOM class

Section 2.5 showed that programs written in JavaScript mostly manipulate the HTML DOM structure to change the website's layout. Because of the different DOM browser specifications, GWT provides many widgets (see section 3.5.1 to get an overview) wrapping up this functionality. But sometimes it is necessary to have direct access to the HTML DOM model. Luckily GWT provides an almost browser independent DOM (API is available under [Goo10a]) class for manipulating HTML directly. As mentioned in the previous part of this section, every widget extends UIObject and inherits the getElement method. This method returns the underlying DOM element node. For example the function widget.getElement().getStyle() accesses the element's style properties. These can be manipulated as shown in listing 46. Listing 47 shows how the restyling can be done in a more compact way using the GWT DOM class.

```
RichTextArea rt1 = new RichTextArea();
Style style = rt1.getElement().getStyle();
style.setBorderColor("blue");
style.setBorderStyle(Style.BorderStyle.SOLID);
style.setBorderWidth(3, Unit.PX);
```

Listing 46: Manipulating the widget's border style with GWT Style class

```
DOM.setStyleAttribute(rt1.getElement(), "border", "3px␣blue␣solid"
    );
```

Listing 47: Manipulating the widget's border style with GWT DOM class

Before changing an element's properties, it is necessary to find them. The next paragraph explains how to find the corresponding element which triggered a browser event. In the example the table cell or row in which the user clicked should be found. Normally this would be the starting point of the editing process, but to keep the example simple the selected cell or row will only be highlighted. Listing A33 and figure A52 show the complete example. The onPreviewNativeEvent(NativePreviewEvent event) function is called every time a browser event occurs. The code line NativeEvent e = event.getNativeEvent() returns the browser event. Since the web application is only interested in the user's click down event, the program

filters it out with the following code if("mousedown".equalsIgnoreCase(e.getType())). Comparing the event against "click" would not be successful, because this handles left clicks but no right ones. The expression Element el = e.getEventTarget().cast() returns the HTML DOM element on which the user clicked. An InlineLabel widget creates a tag, HTML class will become a element and the FlexTable widget produces <table>, <tr> and <td> tags. With this knowledge the test to find out whether the user clicked into a table cell is if("div".equalsIgnoreCase(el.getTagName())). In this small example, the verification can be done against the HTML tag name. In larger projects it is more secure doing it against an earlier set class name by calling el.getClassName(). The following conditions e.getButton() == NativeEvent.BUTTON_LEFT and e.getButton() == NativeEvent.BUTTON_RIGHT distinguish between a left and a right mouse click on the cell. If the user presses the left mouse button down, the selected cell will be colored yellow by manipulating its background color with the following expression DOM.setStyleAttribute((com.google.gwt.user.client.Element) el, "backgroundColor", "yellow").

Selecting the entire row is slightly more complicated. The information that the tag, which represents the clicked cell, is inside a <td> one and this again is inside the wanted <tr> tag, which represents the row, will be used. Since the <tr> tag has no style attribute, the background color of every cell inside this row has to be changed. Because a row may contain many cells, the <tr> tag may also have a lot of <td> DOM children. A loop iterates over all children, which can be accessed with the following code Element child = (Element) parent.getChild(i), of the <tr> tag. After the <td> cell container has been received, the element can be obtained by executing child.getFirstChildElement(). Finally the background color of the div element will be changed to red.

3.5.4 GWT Designer and view optimization using UiBinder

This subsection part shows that the GWT Designer in Eclipse can be used in the same way as the Swing GUI Builder in Netbeans. The GWT Designer is very interesting for new GWT developers, because it displays the text and icon of all available widgets. In order to use the designer, the Java project must be a GWT Designer one[20]. After a designer Composite class[21] was created, a click at the Java file's bottom Design tab opens the designer perspective. The view layout can be created by drag and drop. Figure A53 shows creating the login composite panel with the GWT Designer. A right click on the button opens a popup. A ClickHandler can be added by doing a right click on the button and selecting AddEventHandler -> click -> OnClick in the popup menu. The designer generates Java code out of the view and partial vice versa. This means it is possible to add widgets as sequential Java code and the designer will insert them to the view, but creating widgets in loops even with static parameter, e.g. for(int i=0; i<10; i++), will be ignored in the designer version 1.5.

Creating large user interfaces results in a lot of generated and confusing Java code. That is why GWT allows describing user interface with XML. Another advantage of designing views with HTML/XML is that GWT uses the innerHTML attribute to create the views. This means the declarative layout will be converted into an HTML string, which will be inserted into the DOM element's innerHTML attribute. This technique avoids executing many DOM calls. Page 41 discusses the speed advantage of innerHTML over DOM manipulations by comparing normal (heavyweighted) with cell (lightweighted) widgets.

This passage shows contrasting pairs of normal Java widget code against the corresponding XML one (see listings 48 and 49). The XML code must be inserted into a UiBinder[22] file.

[20] File -> New -> Project ... -> WindowBuilder -> GWT Designer -> Model -> GWT Java Project
[21] File -> New -> Other ... -> WindowBuilder -> GWT Designer -> GWT Java UI -> Composite
[22] File -> New -> UiBinder

```
<g:ListBox visibleItemCount="3">
  <g:item value="1">1st</g:item>
  <g:item value="2">2nd</g:item>
  <g:item value="3">3rd</g:item>
</g:ListBox>
```

Listing 48: ListBox in XML

```
ListBox lb = new ListBox();
lb.setVisibleItemCount(3);
lb.addItem("1st", "1");
lb.addItem("2nd", "2");
lb.addItem("3rd", "3");
```

Listing 49: ListBox in Java

The g in <g:...> is always defined as com.google.gwt.user.client.ui, which is the package of all GWT widgets. Own widgets can only be used, after the complete package namespace has been defined inside the <ui:UiBinder> tag. The code line xmlns:own="urn:import:de.tu_freiberg.informatik.vonwenckstern.client" represents an example package namespace definition. Afterwards the own components can be used in the following way: <own:MyComponent> </own:MyComponent>. The comparison of listings 50 and 51 shows the large advantage of

```
<ui:style>
  .top { letter-spacing: 10px; }
  .can { background-color: navy;
    font: italic; color: white; }
  .styled { border: dotted 3px
    lime;
    padding: 15px; }
</ui:style>
<g:HTML styleName="{style.top}">
  To<b>p</b>
    <span class="{style.can}">
    can
  </span> get
  <span class="{style.styled}">
    styled
  </span>
</g:HTML>
```

Listing 50: Styled HTML in XML

```
HTML html = new HTML("To<b>p</b>"
    +
  "<span>can</span> get " +
  "<span>styled</span>");
Element el = html.getElement();
DOM.setStyleAttribute(el,
  "letterSpacing", "10px");
Element el2 = (Element) el.
    getElementsByTagName("span").
    getItem(0);
DOM.setStyleAttribute(el2,
  "backgroundColor", "navy");
DOM.setStyleAttribute(el2,
  "font", "italic");
DOM.setStyleAttribute(el2,
  "color", "white");
Element el3 = (Element) el.
    getElementsByTagName("span").
    getItem(1);
DOM.setStyleAttribute(el3,
  "border", "dotted 3px lime");
DOM.setStyleAttribute(el3,
  "padding", "15px");
```

Listing 51: Styled HTML in Java

XML layout over Java layout. It is important that braces have been set, when passing CSS styles to widgets in the UiBinder. The curly brackets tell the designer to use the above defined style classes and not the class name of the defined styles in the project's CSS files. Listing A34 and figure A54 show a more complex user interface declared in XML.

The attribute ui:field allows access to the widgets in Java files. This is done by adding the @UiField annotation before the variable declaration. The name of the variable must be the same as the value of the ui:field attribute. It is important that the visibility of the variable is not private, because otherwise the UiBinder generator has no access to the variable. Listing 52 shows how to handle the click event of the button defined in the XML file.

Creating a declarative layout view with the GWT Designer can be done in the same way as creating Java code. The only difference is that a GWT UiBinder Composite[23] is needed to tell the designer it should describe the view in XML.

[23] File -> New -> Others ... -> WindowBuilder -> GWT Designer -> GWT UiBinder -> Composite

```
<ui:UiBinder xmlns:ui="urn:ui:com.google.gwt.uibinder"
        xmlns:g="urn:import:com.google.gwt.user.client.ui">
  <g:Button ui:field="btn">Click me</g:Button>
</ui:UiBinder>
```

```
@UiField
Button btn;
public UiBinderExample() {
  initWidget(uiBinder.createAndBindUi(this));
  btn.addClickHandler(new ClickHandler() {
    @Override
    public void onClick(ClickEvent event) {
      Window.alert("XML and Java file are now connected!");
    }
  });
}
```

Listing 52: Example how to get access to the widgets defined in the XML
Top: UiButton.ui.xml, Bottom: UiButton.java

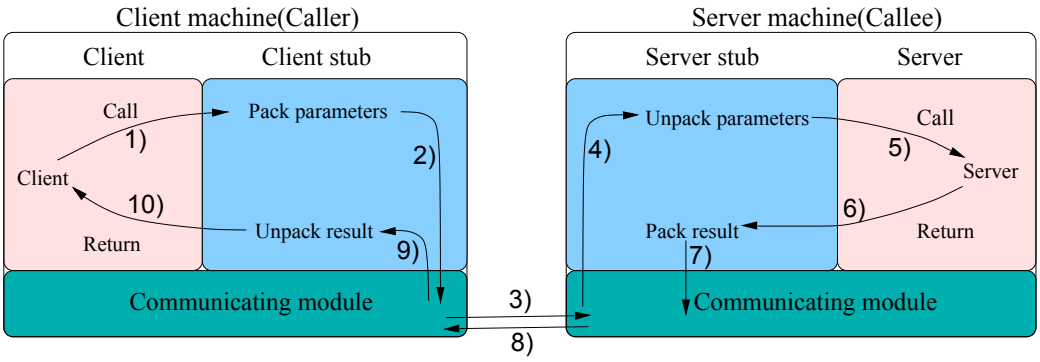


Figure 21: Calls and messages of an invoked Remote Procedure Call, source [Sha, pp. 9-19]

3.6 Remote Procedure Calls

3.6.1 Comparison of Remote Procedure Calls with Remote Method Invocations

1981 Birrell and Nelson described the Remote Procedure Call as the following: A process running on machine M1 calls a procedure P on machine M2. M1 is suspended as long as M2 executes P. When M2 returns then M1 continues working. The aim of both people was that

- the call of any remote procedure should be very similar to calling a local function and
- most importantly, neither message passing from M1 to M2 and vice versa nor the variable serialization or any other input/output operations at all should be visible to the programmer.

Most programming languages support Remote Procedure Calls (RPCs) by using a special-purpose Interface Definition Language for describing the interfaces between client and server. Since 1981 the operating systems' architecture has not supported RPCs, it must be simulated by tools using local procedure calls. This means the tools create a stub component, which the user invokes as a local call. Now, this stub is responsible for all the details (opening, closing the connection, and (de)serialization of parameters), which are necessary for the communication between client and server to work. Figure 21 shows the RPC process. Firstly, the client procedure

calls the local generated client stub function. Secondly, the client stub packs the parameters, creates a message and gives all this to the Communicating Module (CM). Thirdly, the client's CM sends the received message to the remote computer's CM. Fourthly, the remote's CM passes the message to the server stub. Fifthly, the server stub analyses the message and invokes the server procedure with unpacked parameters. Sixthly, the server function does the task and returns the result, an okay or any error notification to the server stub. Seventhly, the server stub packs the result in a message and passes it to the CM. Eighthly, the remote's CM sends the message to the client's CM. Ninthly, the client's CM gives the message to the client stub where the result will be unpacked. Tenthly, the local-invoked client stub function returns the result to the client procedure, and the client procedure can then continue its work. Remote Procedure mechanism handles reference parameters as call-by-values. This means the entire referenced data is copied and transferred to the server. The RPC interface definition is very similar to a set of function declarations in C. It contains the signature of several procedures. The function name and parameter list, which defines the types and notes whether the parameters are input or output ones, belong to the signature.

After the RPC mechanism has been explained, the question arises: How does the client stub know where the server stub is located to exchange the messages. The answer is that the client stub does not know it, unless the remote procedure is registered somewhere. Mapping the service name to a registered server address is called binding. The binding address can be either compiled into all clients or it can be specified at the program start. The program receives the remote procedure location in the following way: The client stub asks the binder service, which address is known, to lookup for the remote procedure name in its table. If the remote procedure name has been registered before, the binding process returns the procedure's address and the remote call can then start.
(These paragraphs are a summary of [Sha].)

Remote Method Invocation (RMI) is the object orientated approach of RPC. This means the object and its method name have to be known to invoke a remote method, instead of only having knowledge of the procedure name in RPC. RMI also supports calls by remote references, where the parameters, which are Remote interfaces, can be passed as arguments. More information on the differences between these technologies can be found at [dif].

Figure A57 and listing 53 shows a Java RMI example, where typed messages can be sent either synchronously or asynchronously to the server. Sending them asynchronously has the advantage that the user interface is not blocked until the response is received. The next passage takes a closure look at the differences between asynchronous RMI calls and synchronous ones. The first difference is that the function, which invokes the synchronous call, returns the result directly and the method, which does an asynchronous call, returns the result via the ActionListener interface. Since the asynchronous version creates a new thread, the client can execute many RMI calls in parallel. Mostly this has advantages, but sometimes it makes handling sequential code more complex. Imagine there is a dropdown list where the user can choose the car brand like BMW. Later the program creates a list, where the user can select the model of the previously picked brand like 3er BMW. Finally, the application creates a list, where the user has to select the correct motorization like BMW 320d. The list can only be created after the user has chosen the model. If all RMI calls are synchronous, then the code could look like the one displayed in listing 54. But if the calls are asynchronous, the code is as complex as shown in listing 55. The last listing shows that the asynchronous model has to use nested ResultListeners to process several steps of data, which depend on RMI results. While it is possible to process synchronous RMI calls in a loop, there is no way to use loops for the asynchronous model. Iterations, which can cause stack overflows, are the substitutes for loops.

```
31  /** wrapper making synchronous RMI asynchronously */
32  public void getMessage(final String msg, final ActionListener
        listener) {
33    if(listener != null) {
34      Runnable run = new Runnable() {
35        @Override
36        public void run() {
37          String result = getMessage(msg);
38          listener.actionPerformed(new ActionEvent(this, 0, result
                ));
39        }
40      };
41      Thread t = new Thread(run);
42      t.start();
43    }
44  }
45
46  /** sending a synchronous RMI message to the server */
47  public String getMessage(final String msg) {
48    try {
49      obj = (RmiServerIntf)Naming.lookup("//localhost/RmiServer");
50      return obj.getMessage(msg);
51    } catch (Exception e) {
52      System.err.println("RmiClient exception: " + e);
53      e.printStackTrace();
54
55      return e.getMessage();
56    }
57  }
```

Listing 53: Excerption of RmiClient.java, see listing A35 for complete code

```
1 CarInfoIntf cars = ...
2 JListBox<String> brands = JListBox<String>(cars.getBrands());
3 // ListSelectionEvent handled on brands...
4 String brand = brands.getSelectedValue();
5 JListBox<String> models = JListBox<String>(cars.getModels(brand));
6 // ListSelectionEvent handled on models ...
7 String model = models.getSelectedValue();
8 JListBox<String> motors = JListBox<String>(cars.getMotors(model));
```

Listing 54: Loading car informations using synchronous RMI calls

3.6.2 GWT's RPC service and serializable whitelist

The subsection part starts with the confusing statement that GWT's Remote Procedure Call mechanism is in reality a Remote Method Invocation mechanism without the ability to send an object by remote reference. GWT calls remote methods on objects as shown in a second. GWT only allows making asynchronous RPC calls. In GWT's RPC service, the website is the client and the web page calls the server, which is always a Java servlet, methods per HTTP POST requests. The easiest way to create a new GWT RPC service is to open GWT RemoteService[24] in Eclipse. Afterwards a client package name like DA_GWTRPC/src/de.tu_freiberg.infor-

[24] File -> New -> Other -> WindowBuilder -> GWT Designer -> Model -> GWT RemoteService

```
1  CarInfoIntf cars = ...
2  cars.getBrands(new ResultListener() {
3    public void returned(Object result) {
4      String[] sBrands = (String[]) result;
5      final JListBox<String> brands = JListBox<String>(sBrands);
6      // ListSelectionEvent handled on brands...
7      cars.getModels(brands.getSelectedValue(), new ResultListener()
           {
8        public void returned(Object result) {
9          String[] sModels = (String[]) result;
10         JListBox<String> models = JListBox<String> (sModels);
11         // ListSelectionEvent handled on models ...
12         cars.getMotors(models.getSelectedValue(),new
               ResultListener(){
13          /* ... */
14        });
15     }    });
16   }  });
```

Listing 55: Loading car informations using asynchronous RMI calls

matik.vonwenckstern.client and a service name like CalendarService have to be chosen. After the new wizard is finished, Eclipse creates the following files[25]:

- CalendarService.java which is the RPC interface definition.

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2  import com.google.gwt.core.client.GWT;
3  import com.google.gwt.user.client.rpc.RemoteService;
4  import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;
5  @RemoteServiceRelativePath("CalendarService")
6  public interface CalendarService extends RemoteService {
7    /**
8     * Utility class for simplifying access to the instance of async
          service.
9     */
10   public static class Util {
11     private static CalendarServiceAsync instance;
12     public static CalendarServiceAsync getInstance(){
13       if (instance == null) {
14         instance = GWT.create(CalendarService.class);
15       }
16       return instance;
17     }
18   }
19 }
```

Listing 56: Generated CalendarService.java file by Eclipse

- CalendarServiceAsync.java is the asynchronous RPC interface. The GWT compiler uses this interface to create a wrapper class to make asynchronous RMI calls. The interface's

[25] Sometimes Eclipse's auto build function does not work correctly. If it is so, these files have to be created (see listings 56, 57, 58 and 59) manually.

functions can be compared with the public void getMessage(final String msg, final ActionListener listener) method in listing 53 at line 32.

```
1 package de.tu_freiberg.informatik.vonwenckstern.client;
2 public interface CalendarServiceAsync {
3 }
```

Listing 57: Generated CalendarServiceAsync.java file by Eclipse

- CalendarServiceImpl.java implementing the RPC interface definition.

```
1 package de.tu_freiberg.informatik.vonwenckstern.server;
2 import de.tu_freiberg.informatik.vonwenckstern.client.
      CalendarService;
3 import com.google.gwt.user.server.rpc.RemoteServiceServlet;
4 public class CalendarServiceImpl extends RemoteServiceServlet
      implements CalendarService {
5 }
```

Listing 58: Generated CalendarServiceImpl.java file by Eclipse

- Eclipse also registers the Java servlet in the DA_GWTRPC/war/WEB-INF/web.xml file by adding the following lines as shown in listing 59.

```
12 <servlet>
13   <servlet-name>CalendarService</servlet-name>
14   <servlet-class>de.tu_freiberg.informatik.vonwenckstern.server.
         CalendarServiceImpl</servlet-class>
15 </servlet>
16 <servlet-mapping>
17   <servlet-name>CalendarService</servlet-name>
18   <url-pattern>/de.tu_freiberg.informatik.vonwenckstern.GWTRPC/
         CalendarService</url-pattern>
19 </servlet-mapping>
```

Listing 59: Modified web.xml file by Eclipse

The CalendarServiceImpl class will be compiled to normal Java byte code. That means the implementation class can use every object of the Java Runtime Environment. Inserting a service interface method ReturnType name(ParameterType1 parameter1,..., ParameterTypeN parameterN) into the RPC definition file also requires adding the corresponding asynchronous interface method void name(ParameterType1 parameter1,..., ParameterTypeN parameterN, AsyncCallback<ReturnType> callback) in the asynchronous RPC class. The asynchronous interface uses object classes as return types, e.g. Integer or Void instead of int or void. The calendar service only calls the GregorianCalendar's get method to collect date information for the website. The interfaces in CalendarService and CalendarServiceAsync are public int get(Date date, int field) and public void get(Date date, int field, AsyncCallback<Integer> callback). After a new interface method has been created, Eclipse generates the asynchronous one by clicking on the error marker left of the synchronous interface definition and choosing generate method get in type CalendarServiceAsync. An easy way to produce the skeletons of the implementation functions is to open the CalendarServiceImpl.java file, click at

the error marker next to the class definition and select *add unimplemented methods*, now. This way Eclipse creates a default implementation of the get method. Listing 60 shows the server side implementation of the get function.

```
30 public int get(Date date, int field) {
31   GregorianCalendar calender = new GregorianCalendar();
32   calender.setTime(date);
33   return calender.get(field);
34 }
```

Listing 60: Implementation of CalendarServiceImpl's get method

The only thing which has to be done is to invoke the remote get method. Listing 61 displays the onModuleLoad function, which calls the server's get method twice to receive further date information. The AsyncCallback<T> interface contains two methods: The first function public

```
32 public final static int WEEK_OF_YEAR = 3;
33 public final static int DAY_OF_YEAR = 6;
34 public void onModuleLoad() {
35   CalendarServiceAsync calInfo = CalendarService.Util.getInstance
       ();
36   Date today = new Date();
37   final HTML html = new HTML("Today␣is:␣" + today);
38   RootPanel.get().add(html);
39   calInfo.get(today, WEEK_OF_YEAR, new AsyncCallback<Integer>() {
40     @Override
41     public void onSuccess(Integer result) {
42       html.setHTML(html.getHTML() + "<br>We␣have␣week␣number␣" +
           result + "␣now.");
43     }
44     @Override
45     public void onFailure(Throwable caught) {
46       caught.printStackTrace();
47       Window.alert("An␣error␣occured.");
48     }
49   });
50   calInfo.get(today, DAY_OF_YEAR, new AsyncCallback<Integer>() {
51     @Override
52     public void onSuccess(Integer result) {
53       html.setHTML(html.getHTML() + "<br>We␣have␣day␣number␣" +
           result + "␣now.");
54     }
55     @Override
56     public void onFailure(Throwable caught) {
57       caught.printStackTrace();
58       Window.alert("An␣error␣occured.");
59     }
60   });
61 }
```

Listing 61: Invoking remote calls to collect more date information

void onSuccess(T result) is called when the remote method could be invoked and the server does not throw any exception. T is the result type of the result variable. It contains the value, which the server implementation returned. The public void onFailure(Throwable caught)

function is called, when the server is not reachable, access is denied, or the remote function throws any exception. One important GWT RPC exception is the StatusCodeException, which is called when the HTTP status code was not 200 (OK). Table 8 lists the most important status codes.

Table 8: HTTP status codes. Copied from [Fie09]

200 OK	The request has succeeded.
4xx Client Error	The 4xx class of status code is intended for cases in which the client seems to have erred.
401 Unauthorized	The request requires user authentication, which is mostly set by a HTTP cookie.
404 Not Found	The server has not found anything matching the Request-URI. This also happens if the servlet has not been registered in the web.xml file.
5xx Server Error	Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has erred or is incapable of performing the re quest.
500 Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request. This happens e.g. when the Tomcat container crashes.

The above example showed that the error handling function of both RPC calls is the same. To avoid writing the same code several times, an abstract Result class, which does all the error handling for the application, can be created. Before this class is created, the server side get method throws some exceptions. Firstly, the interface method definition has to be changed to public int get(Date date, int field) throws Exception in CalendarService.java. Secondly, the get method in CalendarServiceImpl.java will be changed to the one shown in listing 62. Now

```
30 public int get(Date date, int field) throws Exception {
31   if(date == null)
32     throw new NullPointerException("the␣date␣object␣is␣not␣allowed
         ␣to␣be␣null");
33   if(field < 0)
34     throw new IllegalArgumentException("the␣field␣argument␣most␣be
         ␣not␣negative");
35   GregorianCalendar calender = new GregorianCalendar();
36   calender.setTime(date);
37   return calender.get(field);
38 }
```

Listing 62: CalendarServiceImpl's get method throws exceptions now

the Result class will be created as displayed in listing 63. The AbstractAsyncHandler class was generated to do general things, which have to be done after every RPC like logging. Another advantage of writing these 'between' class is, that the Result class has the same method names as the AsyncCallback interface. The remote calls in the onModuleLoad function have to be changed as shown in listing 64. The new source code (listing 64) is much more compact than the old one (listing 61). Another advantage is that error handling is not forgotten. This may occur when the application contains hundreds of remote calls and the method implementations of AsyncCallback have been created by Eclipse. This example showed that the program does not have to deal with object (de)serialization from the client to the server and vice versa. This is one of the big advantages of the Google Web Toolkit. The compiler analyzes the code and

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2  import com.google.gwt.core.client.GWT;
3  import com.google.gwt.user.client.Window;
4  import com.google.gwt.user.client.rpc.AsyncCallback;
5  abstract class AbstractAsyncHandler<T> implements AsyncCallback<T>
       {
6    @Override
7    public void onFailure(Throwable caught) {
8      GWT.log("RPC Failure ["+this.getClass().getName() + "]",
           caught);
9      handleFailure(caught);
10   }
11   @Override
12   public void onSuccess(T arg) {
13     GWT.log("RPC Success [" + this.getClass().getName() + "]",
           null);
14     handleSuccess(arg);
15   }
16   protected abstract void handleFailure(Throwable caught);
17   protected abstract void handleSuccess(T arg);
18 }
19 public abstract class Result<T> extends AbstractAsyncHandler<T> {
20   @Override
21   protected void handleFailure(Throwable caught) {
22     onFailure(caught);
23   }
24   protected void handleSuccess(T arg) {
25     onSuccess(arg);
26   }
27   public void onFailure(final Throwable caught) {
28     caught.printStackTrace();
29     if(caught instanceof NullPointerException) {
30       Window.alert("NullPointerException: \n" + caught.getMessage
             ());
31     } else if(caught instanceof IllegalArgumentException) {
32       Window.alert("IllegalArgumentException: "+caught.getMessage
             ());
33     }
34   }
35   public abstract void onSuccess(T arg);
36 }
```

Listing 63: Source code of Result.java

automatically produces serialization classes for the objects which will be sent over the wire. GWT does not use the powerful standard Java serialization mechanism and this is why not every JRE emulation class can be sent to the server. The following classes are serializable in GWT:

- primitive types such as short, int, byte, and so on; and their object wrappers
- date objects, enumeration classes and strings
- all throwables are serializable
- arrays, ArrayList, HashMap, Stack, Vector and a few more; if these collections and maps only contain serializable types

```
30 callInfo.get(null, WEEK_OF_YEAR, new Result<Integer>() {
31   @Override
32   public void onSuccess(Integer result) {
33     html.setHTML(html.getHTML() + "<br>We have week number " +
           result + " now.");
34   }
35 });
36 callInfo.get(today, DAY_OF_YEAR, new Result<Integer>() {
37   @Override
38   public void onSuccess(Integer result) {
39     html.setHTML(html.getHTML() + "<br>We have day number " +
           result + " now.");
40   }
41 });
```

Listing 64: Using the abstract Result class instead of the AsyncCallback interface as RPC parameter

The general java.lang.Object is not serializable. If the Object is a template parameter e.g. in an ArrayList, then the ArrayList is only serializable if the objects are serializable, otherwise the application receives RPC exceptions during runtime. A class is serializable, if all its attributes except final and transient ones are serializable types. As with the normal Java serialization, the class must provide a no parameter constructor. Most times a RPC exception occurs, then a class does not have a default constructor. In order to serialize an ArrayList object containing own defined class types, these class types have to be added to the GWT's whitelist. The most convenient way is to create a SerializableWhitelist class, which has all class types as attributes. Later, a remote method, which accepts SerializableWhitelist as parameter and as return type, will be created. Listing 65 shows such an example. This example allows one to serialize an ArrayList, which contains a FilePath[] object, but it does **not** allow one to add a TablePath[] object, because only a TablePath attribute has been added to the SerializableWhitelist class.

```
public class SerializableWhitelist implements IsSerializable {
  FilePath fp;
  FilePath[] fps;
  TablePath tp;
}
public SerializableWhitelist addTypesToWhiteList(
    SerializableWhitelist swl){
  return swl;
}
```

Listing 65: pseudo example showing how to add types to GWT serializable whitelist, so that they can get used in ArrayList

GWT also allows defining own (de)serialization methods. [Ker11, p. 80-84] gives more information about this topic.

The end of this subsection shows that it is safer to do the login mechanism with Tomcat instead of creating an own one, which uses GWT RPC calls. The main aim is to illustrate that Google's RPC mechanism cannot guarantee any security. Listing 66 shows the source code of this example. Manipulating the result is not even difficult: Firstly, a JavaScript debugger will be opened, e.g. pressing F12 in the Internet Explorer. Secondly, the generated script file will be selected and a search for the word "access" will find the location which causes the error message. Thirdly, a breakpoint will be set at this location and the debugging process will start. After

```
public class LoginServiceImpl extends RemoteServiceServlet
    implements LoginService {
  @Override
  public boolean login(String username) {
    return username.equals("admin") || username.equals("user");
  }
}
```

```
private Button btnLogin = new Button("Login");
private TextBox txtUser = new TextBox();
public void onModuleLoad() {
  VerticalPanel vpanel = new VerticalPanel();
  vpanel.add(new HTML(
    "insert 'admin' to get administrative rights<br>" +
    "insert 'user' to get only reading rights"));
  vpanel.add(txtUser);
  btnLogin.addClickHandler(this);
  vpanel.add(btnLogin);
  RootPanel.get().add(vpanel);
}
@Override
public void onClick(ClickEvent event) {
  LoginService.Util.getInstance().login(txtUser.getValue(), new
      AsyncCallback<Boolean>() {
    @Override
    public void onSuccess(Boolean result) {
      if(!result) {
        Window.alert("Access denied!");
        return;
      }
      RootPanel.get().clear();
      RootPanel.get().add(new HTML("You are logged in as " +
          txtUser.getValue()));
    }
    @Override
    public void onFailure(Throwable caught) {
      caught.printStackTrace();
      Window.alert("error occured");
    }
  });
}
```

Listing 66: source code showing login mechanism using GWT RPC
Top: server side implementation of RPC, bottom: code for creating the view and calling the server to authenticate the user

entering any user name into the text box, the web applications calls the server to check the login name. Fifthly, the access variable will be changed from `false` to `true` in order to grant access, even if the server denied it. Figure 22 illustrates the hacked results after the debugger's continue button was pressed. Browser plugins like HackBar for Firefox even allow manipulating POST calls to the server. This way the user name can be changed, which grants the attacker more access and allows more damage to be done. For more information about hacking GWT RPC calls see [Gut]. Section 6.5 explains how more secure logins can be done by using Tomcat's login mechanism and asking for every server operation tomcat whether the user has rights to

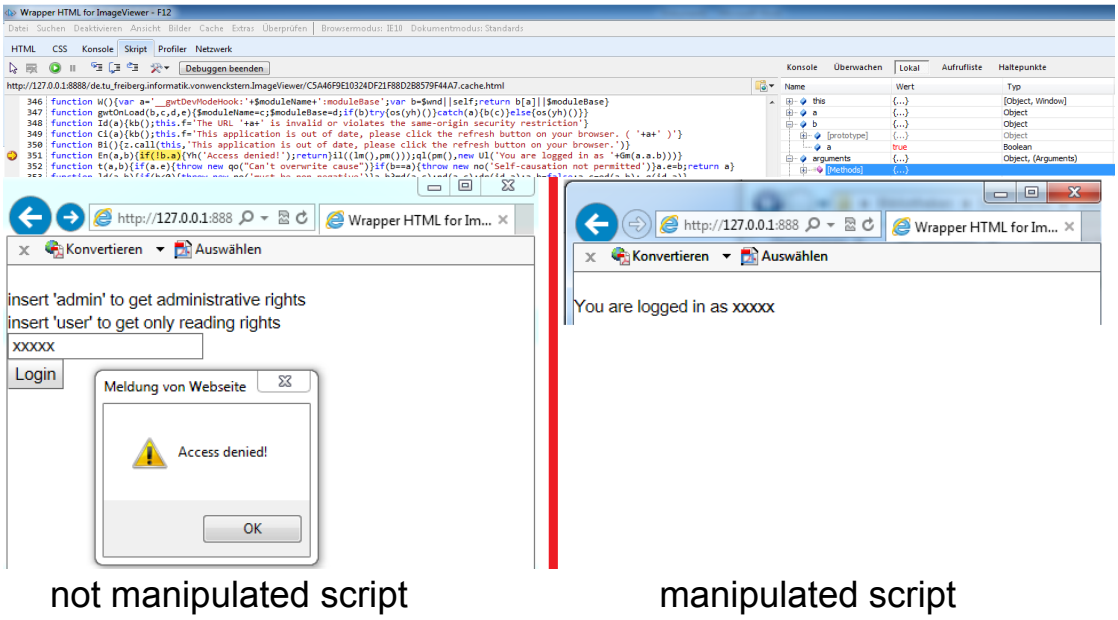


Figure 22: HTTP post result manipulated in IE debugger.

Table 9: GWT application's URL of different program states

state	URL
page1:name=von Wenckstern;firstName =Michael;sex=male;	http://.../survey.html#page1%3Aname%3Dvon %20Wenckstern%3BfirstName%3DMichael %3Bsex%3Dmale%3B
page2:hobbies=soccer,tennis,volleyball; friends=Maren,Stefan,Oliver	http://.../survey.html#page2%3Ahobbies%3D soccer%2Ctennis%2Cvolleyball%3B friends%3DMaren%2CStefan%2COliver

execute remote calls. This mechanism prevents granting more rights by manipulating POST calls.

3.7 History Management

The main difference between dynamic and static websites is that dynamic ones will not load a new web page when the user clicks on a hyperlink or triggers any action command. The advantage of dynamic pages is that the browser will not load an entire new website to display new information. This results in a shorter page rendering time, and so the user sees the new information earlier. But dynamic web applications also have a big disadvantage: Because they do not load a new HTML page, the browser does not know that the content has changed. This is the reason why the browser's history management will not work. The result is that the user cannot navigate to previous application states in most AJAX homepages by pressing the back button. The next example is a web application doing a survey, which contains different questionnaire pages. In this case the user expects to return to the previous page by pressing the browser's back button. Fortunately, GWT supports manipulating the browser's history to give the user the expected feeling. GWT allows setting different markers in the URL, which represents different states of web applications. Table 9 shows a mapping of different URLs to different states in the survey example. If the application sets the actual state to the browser history every time the user clicks on the next button, then the browser URL will change. This makes the browser believe

that a new HTML page has been loaded. The browser keeps these URLs which represent the different web application states. This allows the user to jump to a defined state, which is the URL part after the pound sign, after the back button has been pressed. Since the URL before the hash sign did not change, the web browser will not reload the page. The invocation of GWT's History.newItem(String state) function sets a new history state to the URL.

The method History.addValueChangeHandler adds a URL changed handler to the rich internet application. This function receives events when the URL history has been manipulated. Pressing the forward or backward button in the browser invokes such a notification to the GWT program.

Figures A55, A56, A58, A59 and listing A36 show an example of how to manipulate the browser's history and how to react to history changes caused by the user.

This paragraph explains the most important code parts. Listing 67 illustrates the source code,

```
129     } else if(title.equals("next1")) {
130       String n = name.getValue().trim();
131       String fn = firstName.getValue().trim();
132       if(n.isEmpty() || fn.isEmpty()  || sex.getSelectedIndex() ==
              -1) {
133         Window.alert("You did not insert all necessary data!");
134       } else {
135         actualState.put("page", "page2");
136         actualState.put("name", n);
137         actualState.put("firstname", fn);
138         actualState.put("sex", sex.getValue(sex.getSelectedIndex()
              ));
139         setHistoryURL();
140       }
141     } else if(title.equals("next2") || title.equals("prev2")) {
```

Listing 67: Source code of Survey.java (excerption, see listing A36 for complete source code)

which is executed after the user presses the next button on page 1. Line 132 checks if the user inserted all necessary data. Lines 135 to 139 save the actual state into a HashMap. After this has been done, the call setHistoryURL in the next line creates a new browser URL out of the content of the HashMap.

Listing 68 displays the code of this function. Line 177 iterates over all HashMap keys. The

```
175   public void setHistoryURL() {
176     String state = "";
177     for(String key : actualState.keySet()) {
178       state += key + "=" + actualState.get(key) + ";";
179     }
180     History.newItem(URL.encode(state));
181   }
```

Listing 68: Source code of Survey.java (excerption, see listing A36 for complete source code)

next line generates a key-value-pair like firstname=Michael. The string state contains all of the key-value-pairs separated by a semicolon. Line 180 encodes the state to a valid browser URL, which will be added to the web browser. This causes a notification of the created history

token. That means the onValueChange function is invoked straight after the newItem method of the History class has been called. If the history changed event should not be fired, then History.newItem(URL.encode(state), false) has to be invoked. The second parameter false prevents the browser from firing the history changed event. Listing 69 displays the most interesting part of the public void onValueChange(ValueChangeEvent<String> event) function.

```
189     String keyValues[] = state.split(";");
190     for(String keyValue : keyValues) {
191       String key = keyValue.split("=")[0];
192       String value = keyValue.split("=")[1];
193       actualState.put(key, value);
194     }
195     RootPanel.get().clear();
196     if(actualState.get("page").equals("page1")) {
197       firstName.setValue(actualState.get("firstname"));
198       name.setValue(actualState.get("name"));
199       if(actualState.containsKey("sex")) {
200         sex.setSelectedIndex(actualState.get("sex").equals("female
              ") ? 0 : 1);
201       } else {
202         sex.setSelectedIndex(-1);
203       }
204       RootPanel.get().add(page1);
205     } else if(actualState.get("page").equals("page2")) {
```

Listing 69: Source code of Survey.java (excerption, see listing A36 for complete source code)

Lines 189 to 194 are more or less the opposite of the setHistoryURL method shown in listing 68. Line number 195 removes all widgets of the website's body element. This results in deleting the previously displayed survey page. Lines 197 and 198 set the values of the TextBoxes to the ones stored in the HashMap. The next line checks whether the HashMap contains any information about the user's sex. This test avoids a null pointer exception when the code actualState.get("sex").equals(...) is executed. This check is needed because the user can manipulate the URL directly and so the sex=male or sex=female parameter can be removed. Line number 204 adds the first survey page to the browser's body element. This results in showing up the first page with all the information the user inserted earlier. This example demonstrated how any GWT application can support the browser's forward and back button.

This subsection provides an answer to the question as to how a web application stores its states in GWT. The survey model data has been saved to the URL. This gives the user the opportunity to bookmark the actual website together with the actual data. This way the data is written to the hard disk. The user can share the program state easily by sending anybody the generated URL, e.g. per e-mail. The only limitation is that the URL length is limited 2 MB[26]. This means that the web program cannot store a gigabyte sized state as a desktop program could do. So the answer whether it is possible to save a web application state in GWT is a partial yes; because small amount of data can be stored and shared, but it does not work for larger data.

[26] minimum except of IE 10: Firefox 16: 20MB, Chrome 21: 2MB, IE 10: 2 KB; measured with the program shown in listing A37

3.8 Client Bundle

3.8.1 Using ImageResources in the ClientBundle interface

A detailed introduction to ClientBundles can be found at [Goo10e]. This subsection explains how to use image icons in GWT applications with the ClientBundle interface to speed up the application. It also shows how to define browser specific CSS, which are recognized in the GWT deferred binding process.

The speed test program displays 123 push buttons with a specific LaTeX icon. The time needed to show the 123 buttons will be measured with and without the usage of a GWT ClientBundle. The IconsURL class sets the image URL to the push buttons and is displayed in listing 70. The IconsClientBundle class sets the image resource to the push buttons and is shown in listing 71. Line 2 stores the reference containing all images into the im variable.

```
1  public class IconsURL extends Grid {
2    private final String[] latexIconsURL = new String[] {"((a.png",
       "(a.png",... };
3    public IconsURL() {
4      ...
5      for(String url: latexIconsURL) {
6        Image im = new Image("latexImg/" + url);
7        PushButton btn = new PushButton(im);
8        ...
9        this.setWidget(row, col, btn);
10     } /* end for */ } /* end IconsURL() */ } /* end class */
```

Listing 70: Java code of IconsURL class

```
1  public class IconsClientBundle extends Grid {
2    private final Images im = Images.Util.getImages();
3    private final String[] latexIconsClientBundle = new String[] {im
       .img0().getSafeUri().asString(), im.img1().getSafeUri().
       asString(),...}
4    public IconsClientBundle() {
5      ...
6      for(String url: latexIconsClientBundle) {
7        Image im = new Image(url);
8        PushButton btn = new PushButton(im);
9        ...
10       this.setWidget(row, col, btn);
11     } /* end for */ } /* end IconsURL() */ } /* end class */
```

Listing 71: Java code of IconsClientBundle class

The next line creates a String array, which contains the link to ImageResources. For any picture smaller than 40 KB, the link of the ImageResource object is equal to the picture's content. Google uses data URLs to the store the content as String. The String value of im.img0().getSafeUri().asString() is "...". The source code of the Images interface, which extends the ClientBundle interface is shown in listing 72.

Line 2 creates a utility class returning the Images interface. This class is the same for every image client bundle. Line 13 defines the img0 function, which returns the ImageResource of the picture "pics/((a.png". Each method in the Images interface needs to define the relative path of

```
1  public interface Images extends ClientBundle {
2    public static class Util {
3      private static Images images = null;
4      public static Images getImages() {
5        if(images == null) {
6          images = GWT.create(Images.class);
7        }
8        return images;
9      }
10   }
11
12   @Source("pics/((a.png")
13   ImageResource img0();
14   @Source("pics/(a.png")
15   ImageResource img1();
16   ...
17 }
```

Listing 72: Java code of Images interface

the image it returns with the Source annotation. In addition to the required Source annotation, the ImageOptions one can be added to define the width, height, or other image options by writing e.g. @ImageOptions(width = 118, height = 156).

In order to inspect the GWT generated classes, which implement the Images interface and return the ImageResources of the pictures, the project has to be compiled with the additional parameter -gen C:\...\DA_ClientBundleImage\gen. Now the files Images_default_InlineClientBundleGenerator.java and Images_default_StaticClientBundleGenerator.java in the directory C:\...\DA_ClientBundleImage\gen\de\tu_freiberg\informatik\vonwenckstern\client\resource can be opened with Eclipse.

The Images_default_InlineClientBundleGenerator class contains private static and final variables externalImage, externalImage0, and so on, which contain the data URL of the images. The declaration of externalImage is private static final java.lang.String externalImage= "..." in these examples. Each function in the Images interface has its own static initializer class, which creates in the constructor the ImageResourcePrototype object with the declared annotation parameter such as width, height, and the data string externalImageX (the @Source parameter). The static get method of this class returns the ImageResourcePrototype object. The implementation of the Images interface's img0 method returns only the ImageResourcePrototype object of the img0Initializer class. The Images_default_StaticClientBundleGenerator class contains the location of one big picture, which contains all the small images. The static initializer classes extract in their constructor the small image out of the big one. If the web browser supports data URLs, then the GWT compiler will use the Images_default_InlineClientBundleGenerator class as Images interface implementation by default. Adding the code line <set property name="ClientBundle.enableInlining" value="false" /> into the *.gwt.xml file and recompiling the project, forces the compiler to use the Images_default_StaticClientBundleGenerator class to reduce the size of the *.cache.html files.

If the web project sets ClientBundle.enableInlining to false, then the code of the IconsClientBundle needs to be changed as shown in listing 73. The main difference is that the Image object has been created with the ImageResource imRes, instead of creating it with a String containing the link of the ImageResource. If the Image has been created by the ImageResource object, GWT sets the picture as CSS background image. This is needed to cut the desired image

```
1  public class IconsClientBundle extends Grid {
2    private final Images im = Images.Util.getImages();
3    private final ImageResource [] latexIconsClientBundle = new
       ImageResource [] {im.img0(), im.img1() ,...}
4    public IconsClientBundle() {
5      ...
6      for(ImageResource imRes: latexIconsClientBundle) {
7        Image im = new Image(imRes);
8        PushButton btn = new PushButton(im);
9        ...
10       this.setWidget(row, col, btn);
11     } /* end for */ } /* end IconsURL() */ } /* end class */
```

Listing 73: Java code of IconsClientBundle class when ClientBundle.enableInlining property is disabled.

out of the big image bundle. If the Image object is created with the code shown in listing 71, then GWT would set the picture as HTML src parameter. The result would be that the large image bundle would be shown every time, instead of the desired small image. The problem with setting a picture as background image is that it can no longer be easily scaled. Changing the size of the Image im object by calling im.setPixel(width, height) will not change the size of the sub picture.

Listing 74 shows the ScalableImage class, which accepts an ImageResource object in the constructor and allows adjusting the size of the sub picture to the wanted pixel size. This code updates the CSS properties background-position, and background-size additional to the CSS properties height and width to achieve the wanted behavior. Listing 75 displays the first three lines in the for each loop. These lines show how to use the ScalableImage to resize the sub images of the large picture bundle. If the ImageResource imRes object has not been added to the root panel, then the onLoad method in the ScalableImage class will not be called, and the image is not shown.

When the browser shows the IconsURL panel, it has to load 123 small pictures from the server. Since the browser only opens a certain number of GET requests at the same time, it needs several GET rounds to load all the images. Figure A60 illustrates the network traffic. The result is that the user has to wait about 3 seconds[27] until all images are loaded. When the user clicks on the "show LaTeX Icons loaded from ClientBundle" button, then all the images are visible immediately. Figure A61 shows the network traffic. The reason is that GWT compiles all the images into the HTML as data URLs, and so the browser does not have to fetch any picture at all. After disabling the ClientBundle.enableInlining property, the browser only loads the big image bundle, which needs about 200ms. Figure A62 displays the network traffic; there were two images loaded, because GWT sets the img src parameter to a transparent one pixel large gif icon. The user does not recognize the 200ms; and because the big picture ends with *.cache.png, it will be downloaded only the first time and the next time it will be loaded from the web browser's cache and is directly available.

Most large web application projects contain several hundred images. There are, for example the standard icons: load, save, print, insert, copy, cut, then there are the arrow and folder icons for any trees like expand, not expand, expand and selected, closed folder, opened folder, closed folder no access, and so on, and there are mostly icons for windows like the close, close-hover, minimize, minimize-hover, maximize, restore, and many border icons with nice color gradients.

[27] In this test the compiled GWT project was uploaded to kilo.de, and the page was opened in Firefox at a house with a 6 Mbit/sec download rate.

```
1  import com.google.gwt.resources.client.ImageResource;
2  import com.google.gwt.user.client.DOM;
3  import com.google.gwt.user.client.ui.Image;
4
5  public class ScalableImage extends Image {
6    private int width;
7    private int height;
8    private ImageResource res;
9
10   public ScalableImage(ImageResource res, int width, int height) {
11     this.setUrl(res.getSafeUri());
12     this.res = res;
13     this.width = width;
14     this.height = height;
15   }
16
17   @Override
18   public void onLoad() {
19     int widthOfBigImage = this.getOffsetWidth();
20     int heightOfBigImage = this.getOffsetHeight();
21     double scaleX = width / res.getWidth();
22     double scaleY = height / res.getHeight();
23     this.setResource(res);
24     DOM.setStyleAttribute(getElement(), "backgroundPosition",
           Integer.toString((int) (res.getLeft() * -1 * scaleX))+"px "
           +
25          Integer.toString((int) (res.getTop() * -1 * scaleY))+"px "
               );
26     DOM.setStyleAttribute(getElement(), "backgroundSize", Integer.
           toString((int) (widthOfBigImage * scaleX))+"px "+
27          Integer.toString((int) (heightOfBigImage * scaleY))+"px ")
               ;
28     this.setPixelSize((int) (res.getWidth()* scaleX), (int) (res.
           getHeight() * scaleY));
29   }
30 }
```

Listing 74: Java code of ScalableImage class allowing to resize the subfigures.

```
1 ScalableImage im = new ScalableImage(imRes, 60, 60);
2 RootPanel.get().add(im); // PushButton class does not fire
      onAttach event, so we need to attach the image to the RootPanel
3 PushButton btn = new PushButton(im);
```

Listing 75: Example code how to use the ScalableImage class.

Because the user should not have to wait several seconds when it opens a new dialog for the first time, it is advisable to save the application images into ClientBundles.

3.8.2 Using CssResources in the ClientBundle interface

This subsection part explains the use of ClientBundle's CssResource types. The example will be the creation of a CSS transition navigation menu as shown in figure 23. The CSS effect code is copied from [Chi, pp. 72-73]. First of all, the transitionNavigation.css file has to be

Figure 23: Screenshot of CSS transition navigation menu

created as shown in listing 76. This code contains the special CSS3 transition effects. It will be optimized later. Listing 77 displays the TransitionNavigation interface, which extends the CssResource one. This class has its own interface method for every CSS class definition. There is only one CSS class definition .ownStyle in the CSS file. The name of the interface function can be arbitrary as it will be linked to the CSS class. This is done with the Java annotation @ClassName (without the point).

The navigation.html (see listing 78) file contains the menu's HTML code. The menu could also be created with the UiBinder interface by writing this HTML code into the XML file, but this example wants to illustrate a different way.

Listing 79 shows the Resources interface extending the ClientBundle one. This interface has the two functions navigation returning the content of the HTML file, and navigationCSS returning the TransitionNavigation interface, which represents the CSS file. The fact that the CSS content will be injected into the browsers DOM before it is used is important. The best way of doing this is to call the ensureInjected function (line 7 in listing 79) directly after the ClientBundle object has been created with GWT.create.

The entry point class CSS is displayed in listing 80. Line 3 stores the Resource interface reference into the variable res. The next line creates an HTML widget with the text specified in navigation.html. Line 6 adds the CSS class ownStyle to the body element in order to override the standard background and text color. That was all the work needed to compile the CSS file into the GWT generated HTML one. The advantage is that the web browser does not have to load the extra transitionNavigation.css file and saves the time for one GET round trip.

The CSS in listing 76 will be optimized and simplified for changes. If the menu entries become larger, then the width properties in li a and ul have to be updated. A person who looks at the CSS code for the first time does not know where it has to change the width. There are two possibilities which make it easier for other persons to enlarge the menu.

- At the top of the CSS file the menuWidth and menuWidthInner parameter can be defined with the following lines:

```
1  @CHARSET "ISO-8859-1";
2  <!-- /*
3  CSS copied from Chip magazine Web Design 2013, pp. 72-73
4  */ -->
5
6
7  .ownStyle {
8          background-color: black;
9          color: #6A8916;
10 }
11
12 ul {
13         width: 230px;
14         background-color: #303728;
15         padding: 0px;
16 }
17
18 li {
19         list-style-type: none;
20 }
21
22 li a {
23         display: block;
24         width: 196px;
25         padding: 3px 4px;
26         margin: 5px 13px;
27         color: #969E8D;
28         border-bottom: 1px dotted #96BF1F;
29         text-decoration: none;
30         -moz-transition: background-color 0.3s ease-in;
31         -webkit-transition: background-color 0.3s ease-in;
32         -o-transition: background-color 0.3s ease-in;
33         transition: background-color 0.3s ease-in;
34 }
35
36 li a:hover {
37         background-color: #96C11F;
38         color: #fff;
39         -moz-transition: background-color 0.01s;
40         -webkit-transition: background-color 0.01s;
41         -o-transition: background-color 0.01s;
42         transition: background-color 0.01s;
43 }
```

Listing 76: CSS code of transitionNavigation.css file

@def menuWidth 230px;
@def menuWidthInner 196px;
And the width definitions in ul and li a has to be changed as shown in listing 81.

- Alternatively, the outer width will be only defined as a static field in an extra class and two Java functions, which return the menuWidth and the innerMenuWidth calculated out of the width field, will be added. Listing 82 shows an example code. It is important that the menuWidth and innerMenuWidth function return a String, because if the result is e.g. an int value, then the GWT compiler will not evaluate them. The CSS parameters

```
1 import com.google.gwt.resources.client.CssResource;
2
3 public interface TransitionNavigation extends CssResource {
4   @ClassName("ownStyle")
5   String ownStyle();
6 }
```

Listing 77: Java code of TransitionNavigation interface.

```
1 Resource types:
2 <ul>
3   <li><a href="...#ClientBundle">Overview</a></li>
4   <li><a href="...#DataResource">Data resources</a></li>
5   <li><a href="...#TextResource">Text and external text resources<
        /a></li>
6   <li><a href="...#ImageResource">Image resources</a></li>
7   <li><a href="...#GwtCreateResource">GWT-create resources</a></li
        >
8   <li><a href="...#CssResource">CSS resources</a></li>
9 </ul>
10 }
```

Listing 78: HTML code of navigation.html text file.
The ellipses in the URL stands for https://developers.google.com/web-toolkit/doc/latest/DevGuideClientBundle

```
1 public interface Resources extends ClientBundle {
2   public static class Util {
3     private static Resources res = null;
4     public static Resources getResources() {
5       if(res == null) {
6         res = GWT.create(Resources.class);
7         res.navigationCSS().ensureInjected();
8       }
9       return res;
10    }
11  }
12
13  @Source("navigation.html")
14  TextResource navigation();
15
16  @Source("transitionNavigation.css")
17  TransitionNavigation navigationCSS();
18 }
```

Listing 79: Java code of Resources interface.

menuWidth and menuWidthInner have to be defined in followed way:
@eval menuWidth TransitionNavigation.Constants.menuWidth();
@eval menuWidthInner TransitionNavigation.Constants.innerMenuWidth();
The ul and li a width definitions are equal to the ones in the first possibility.

I personally prefer the second version better, because only the _menuWidth value has to be changed and the CSS file does not have to be touched at all.

```
1 public class CSS implements EntryPoint {
2   public void onModuleLoad() {
3     Resources res = Resources.Util.getResources();
4     HTML html = new HTML(res.navigation().getText());
5     RootPanel.get().add(html);
6     RootPanel.getBodyElement().addClassName(res.navigationCSS().
          ownStyle());
7   }
8 }
```

Listing 80: Java code of CSS entry point class.

```
1 ul {
2   width: menuWidth;
3   ... }
4 li a {
5   width: menuWidthInner;
6   ... }
```

Listing 81: CSS parameterized with menuWidth and menuWidthInner.

```
1 public interface TransitionNavigation extends CssResource {
2   public static class Constants {
3   /** outer menu width in pixel */
4     private static int _menuWidth = 230;
5     public static String menuWidth() { return _menuWidth + "px"; }
          // function must return String, otherwise it does not
          work
6     public static String innerMenuWidth() { return Integer.
          toString(_menuWidth - 34) + "px"; }
7   }
8   @ClassName("ownStyle")
9   String ownStyle();
10 }
```

Listing 82: Saving CSS parameter in Java file.

The next paragraphs optimize the CSS code by eliminating unused vendor specific CSS commands like -moz-transition, -webkit-transition, -o-transition and transition. The prefixes -moz-, -webkit- and -o- stand for the vendor specific CSS commands of Mozilla Firefox, Webkit renderer in Safari or Google Chrome, and Opera. Internet Explorer 10 only needs the transition CSS command and will ignore the vendor specific commands anyway, and Chrome only pays attention to the -webkit-transition instruction. This is why conditional CSS should be used to deliver for each web browser only the CSS command it needs and not more. The li a and li a:hover rules in the transitionNavigation.css file are displayed in listing 83.

The available user agent deferred binding values are in gwt-user.jar's com.google.gwt.user-agent.UserAgent.gwt.xml file located and differ from various GWT versions. CSS rules, e.g. for li a, can be defined at different places; the GWT compiler merges them together into one rule. The first occurrence of li a and li a:hover contains the CSS code which is the same for all web browsers; the second one has only the code for the specific browser. In GWT the keywords @if, @elif, or @else define conditional CSS. The condition in @if or @elif always looks like <deferred-binding-property> <space-separated list of values>. Figure A63 illustrates the

compiled result without the usage of conditional CSS, and figures A64, A65, and A66 show it with the usage of conditional CSS. Conditional CSS reduces the size of the *.cache.html files so that they can be downloaded faster.

```
li a {
  display: block;

  width: menuWidthInner;
  padding: 3px 4px;
  margin: 5px 13px;
  color: #969E8D;
  border-bottom: 1px dotted #96BF1F;
  text-decoration: none;
}

li a:hover {
  background-color: #96C11F;
  color: #fff;
}

@if user.agent gecko1_8 {
  li a { -moz-transition: background-color 0.3s ease-in; }
  li a:hover { -moz-transition: background-color 0.01s; }
} @elif user.agent safari {
  li a { -webkit-transition: background-color 0.3s ease-in; }
  li a:hover { -webkit-transition: background-color 0.01s; }
} @elif user.agent opera {
  li a { -o-transition: background-color 0.3s ease-in; }
  li a:hover { -o-transition: background-color 0.01s; }
} @else {
  li a { transition: background-color 0.3s ease-in; }
  li a:hover { transition: background-color 0.01s; }
}
```

Listing 83: Using conditional CSS to deliver only browser-specific CSS.

This subsection will finish by showing how the access to ImageResources in CSS files can be established. The above example will be extended with a background picture. Figure 24 illustrates the beautiful final result. The following three lines shown in listing 84 should be added to the Resource interface.

```
@ImageOptions(width=1900)   // picture downloaded from
@Source("light_waves.jpg")  // http://www.freeimageslive.co.uk/
ImageResource bgImage();    // free_stock_image/lightwavesjpg-0
```

Listing 84: Adding ImageResource function bgImage() to Resource interface.

Finally, the CSS class .ownStyle has to be changed as shown in listing 85.

```
@sprite .ownStyle {
  gwt-image: "bgImage";
  color: #6A8916;
}
```

Listing 85: Adding ResourceImage object into CSS file.

Figure 24: ImageResource as background picture

The code gwt-image: "bgImage" tells the GWT compiler that it should use the ImageResource bgImage() as background picture. In order to use the GWT specific CSS command gwt-image, the @sprite annotation has to be added before the CSS class definition.

The GWT Client Bundle technology can be compared to the Java mechanism which allows storing resources into the JAR file. If the image resources are small enough, then both store them into one file. In GWT the images are included in the *.cached.html file. The aim of both mechanisms is to speed up the resource loading time by loading them into the browser's DOM or Random-Access Memory (RAM) before they are actually needed.

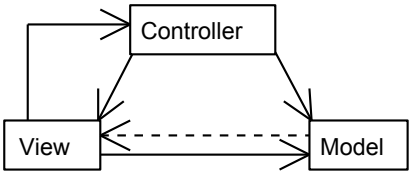


Figure 25: Model View Controller pattern, source [Der]

4 Model-View-Presenter Architecture

4.1 Comparison of MVP and MVC

This entire subsection is a summary of Interactive Application Architecture Patterns [Der].

Introduction

Both Model-View-Controller and Model-View-Presenter patterns have the goal of separating the concerns of interactive applications: Model-View-Controller (MVC)'s primary design intention was to separate the presentation layer from domain concerns. The existence of the Controller component was a byproduct of dividing the input and output of a program due to the complexities inherited from the host operation system. Today the separation between device input and output at application level has not to be carried out, because most runtime environments like JRE or .NET already do this. This means that the formalize work of the Controller for intercepting user input is no longer necessary in platforms which natively provide this function. Many elemental design patterns as described in Gamma's book "Design Patterns- Elements of Reusable Object-Oriented Software" are well structured patterns giving implementation independent solutions to the most common problems. In order to understand the creation of composite patterns like interaction ones, it is very helpful to understand the history of their further development as no general solution for all platforms exists.

The choice of a design pattern should be made by taking into account the following rule: Firstly a very similar problem, for which a design pattern already exists, should be searched for and then used. However, it is not recommended to use a pattern in which the real life problems were synthetically created after it was first invented.

Model-View-Controller pattern

The MVC pattern was invented to separate the application's model, presentation and user input into different expert elements. Trygve Reenskaug invented the MVC pattern in 1979 to have an interface for manipulating multiple views of data at once. As the name assumes, the pattern consists of three components: A Model holding the data and business functionality of the program. Generally, real world objects, processes and their behavior will be mapped into a Domain Model. In nearly all applications the View displays the Model data using widgets and other user interface components. The Controller takes care of user input actions like mouse clicks or key hits and thus allows the user to interact with the View and to change the data Model. Figure 25 displays the relationship between these three components.

Every object which can be manipulated by the user, contains of a Model-View-Controller triad. It is important that the Model has neither access to the View nor to the Controller. But both of them can change the Model's state. The Model can only communicate to the Controller or to the View by sending messages to them. The View and Controller have a link to the Model and work together, allowing the user to manipulate the Model. Since every View is associated with one Controller and vice versa, the Controller class is often a subclass of the View class.

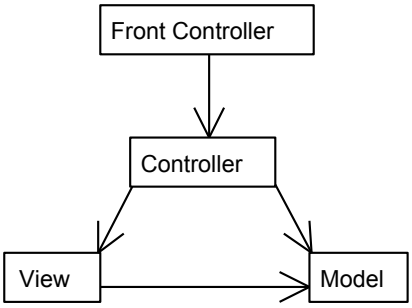


Figure 26: MVC pattern for server intensive web applications, source [Der]

View-Controller represents the presentation layer and the Model the application concerns. The View handles the output, e.g. drawing elements to the screen, and the Controller is responsible for the input, e.g. reacting to user events. This is why both of them have to interact with the Model: The Controller has to manipulate the Model according to the user input. The View has to read the different model data in order to display them.

After the user has triggered some events, the Controller intercepts the input and reacts accordingly. As mentioned before, this can be the interaction with the Model in order to modify some data. Another response can be the change of some visual widgets in the View component, e.g. highlighting scrollbars. In MVC the Controller's task is not to separate the View from the Model. This division is done by using the Observer Pattern and not the Controller. The Controller was introduced as a link between the end user and the application, and not between the View and the Model.

MVC pattern for server intensive web applications

This pattern works very similarly to the previous one, except that the server side input is now processed by specialized components. In JavaServer Pages the MVC separation is done in the following way: HTTP requests are given to the Java Servlet which interacts with the Model and later the Servlet gives the control to a JSP component for generating the HTML code which is then rendered in the browser. The Apache Struts web framework is responsible for the success of this pattern. The aim of the Struts framework was to use the MVC pattern as the following quote shows "The Model represents the business or database code, the View represents the page design code, and the Controller represents the navigational code. The Struts framework is designed to help developers create web applications that utilize MVC architecture." [The08a]

In web applications, a Front Controller administered the common infrastructure concerns (like security, session and login management) and redirects the incoming HTTP requests to individual Controllers. JSP uses a Servlet and ASP.NET an IHTTPHandler interface as Front Controller.

In this pattern the Model does the same as in the previous one. The View is the generated HTML content, which can be a rendered text-based template or a compiled object from a template. Similar to the classical MVC pattern is the Controller being responsible for the user input. The only difference is that it does not receive direct hardware signals like mouse down, key pressed, and so on; but it will get this information about the events from the delegated HTTP request. Figure 26 shows how these four components work together.

The Observer Pattern cannot be used to update the View. The Controller generates a class representing all the data and the behavior of the user interface and sends it to the view, which uses this information to generate the HTML code. This means the view only renders the Controller's output stream.

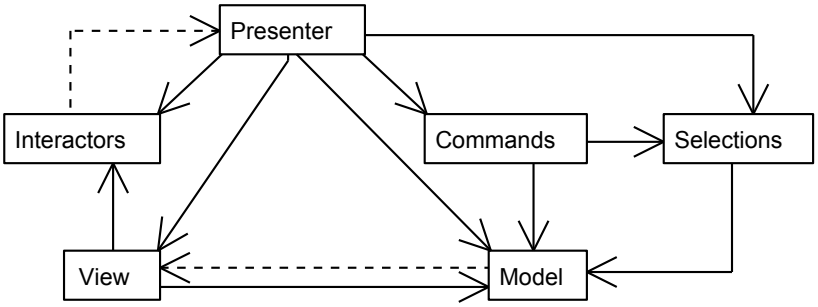


Figure 27: Taligent Model-View-Presenter pattern, source [Der]

Model-View-Presenter pattern

All Model-View-Presenter (MVP) patterns are a variation of the classical MVC pattern. This subsection introduces two MVP variations: Taligent and Dolphin Smalltalk MVP pattern.

Taligent Model-View-Presenter pattern

The Taligent MVP pattern divides the application into data, data manipulation, data specification, user interaction, application coordination, and presentation.

As always, the Model represents the data and business functionality of the program. Selections define parts of the Model's data on which should be operated, e.g. they can define columns or rows. Commands specify the actions which can be carried out with the data. Similar to the traditional MVC pattern, the View displays the Model data to the user by rendering widgets on the screen. Interactors map user events - like mouse clicks or keyboard input - onto operations which will be executed on the Model. The Presenter creates the proper Models, Commands, Selections, Views and Interactors and manages the interactions between these components. Figure 27 illustrates the relationship of these six components.

The most important differences between this pattern and the classical MVC one are the Presenters and Interactors. The Presenter is like a boss for a particular subsystem within the program. It handles the lifecycle and links between the Model, Views, Commands, Interactors, and Selections. Since Interactors intercept the user events, there is no need for Presenters for every widget. Generally a Presenter exists for each view, but sometimes there are multiple Views for one Presenter. This is the case if there exists separate desktop, tablet and phone views, which are all managed by one Presenter. Interactors can be compared with Controllers in the MVC pattern, because they react to user inputs and call the correct Commands and Selections of the Model.

Dolphin Smalltalk Model-View-Presenter pattern

The Dolphin Smalltalk MVP divides the program logic into the three components Model, View and Presenter. By eliminating Selections, Interactors and Commands, the role of the Presenter has been simplified, because now it has only to redirect updates from the View to the Model. The history of the creation of this pattern is quite interesting. Since widgets of the Microsoft Windows operating system already handle most of the controller work, such as user events, the Dolphin Smalltalk team found that the MVC concept of a Controller, whose main task was to react on user events, is no longer needed. This was the reason why they decided to use the Taligent MVP pattern instead of the classical MVC one. But the Dolphin team misinterpreted the

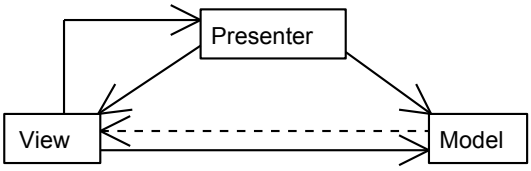


Figure 28: Dolphin Smalltalk Model View Presenter pattern, source [Der]

Taligent MVP pattern and so they made the use of an Application Model redundant by inserting the presentation logic from the Model to the Presenter.

As in the MVC pattern, the Model is responsible for the data and business functionality and the View displays the Model data using widgets. The Presenter has the presentation logic interacting with the Model. Figure 28 shows how these components work together.

Now the view manages the user events fired by the operation system and updates it or forwards the events to the Presenter to update the Model. Similar to the Taligent MVP pattern, a single Presenter is responsible for one View most of the time. The Model uses the Observer Pattern to let the View update itself.

Classical MVC vs. Dolphin Smalltalk MVP

Because GWT uses the Dolphin Smalltalk MVP pattern and Java Swing the classical MVC one, this paragraph compares these two. Both patterns use a triad-architecture and both of them have a Model and View component with the same functionality. The Controller of the MVC pattern as well as the Presenter of the MVP pattern is used to manipulate the model and the Model uses in both the Observer Patterns to update the View.

In the MVC pattern, the main task of the Controller was to intercept the user inputs. The Controller's job of changing the Model was only a byproduct which had to be done after receiving the inputs. On the other hand, the Presenter of the MVP pattern was planned to update the Model. In order to change the Model's data the Presenter has to handle events given by the View.

In GWT it is recommended to use both techniques. The MVP pattern should be used in GWT if the view is a panel which contains many predefined widgets. In this case the View should do the event handling, because all the GWT standard widgets already handle most of the native events and forward these. The View should delegate these events to the Presenter in order to update the Model. The MVC approach should be used in GWT for own created widgets as these have to react to many native events directly, in order to do update its generated HTML. There it is a good idea to let an extra Controller class handle the events.

4.2 GWT Model-View-Presenter pattern example: Agricola board game

While the previous subsection gave an introduction about the development history of the Model-View-Presenter pattern, this one explains the pattern on a game example. This subsection shows how to program the Agricola board game in the GWT specific MVP pattern. Google introduced this pattern at the contacts example, which can be found under the title "Large scale application development and MVP" at [Goo10b]. Generally this section follows the Google design, but sometimes it uses different approaches to have more control or to simplify things.

The next paragraphs present the Agricola board game in order to understand the example which will be programmed. The game is a turn-based strategy game, where the player has to build up a farm.

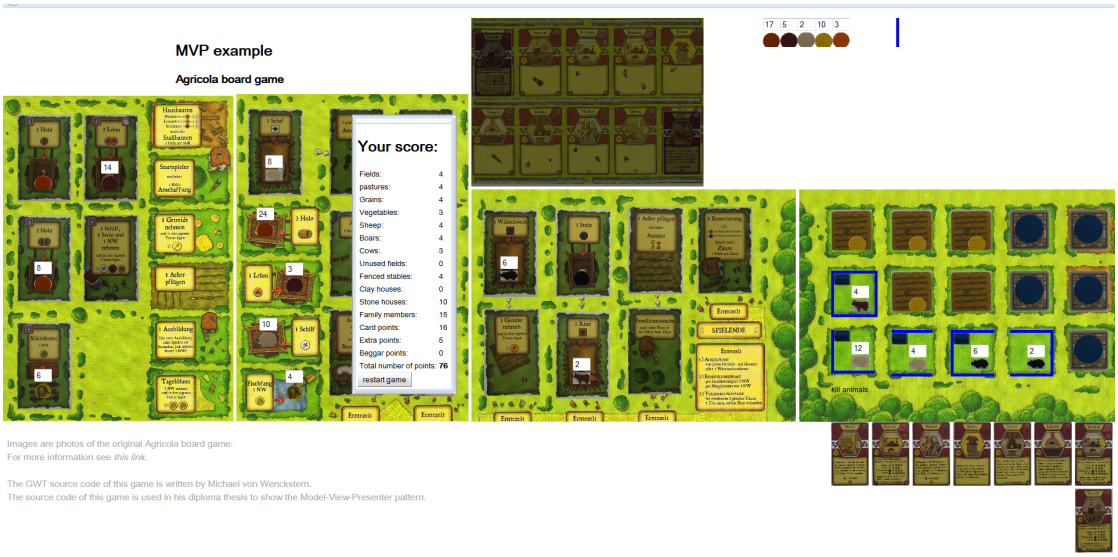


Figure 29: Screenshot of Agricola board game

The board contains 14 rounds belonging to six phases of the game. The phases become shorter and shorter the longer the game goes on, the first one consists of four rounds and the last one has only one round. Each phase finishes with a harvest season; where the gamer can crop its fields, has to feed its family, and the animals can re breed. The aim of the game is to be a rich farmer having many different animals, a large family supporting the agrarian, lots of fields to crop grain or vegetables, and to own many acquisitions like pottery, stone stove, or cookery. Before animals can be held, wood has to be collected in order to build a fence. The farmer must plow a field, get seeds, and finally sow its seeds in order to crop grain or vegetables. Before the farmer's family can have a baby, the house needs to be increased by one room for each new child. Different resources like reed, clay, wood or stone have to be collected to extend the house or to buy new acquisitions. The advantage of such purchases is that they allow the player to do special things like baking bread or killing its animals to make food. The food is needed to feed the family; more family members need more food. I chose this game as MVP example, because:

- It has five playing fields, and so there are at least five views and five presenters.
- The user can interact between these playing fields. This is why an application controller which manages the interaction is needed. The model also requires different events to tell the controller, what happened at one of the playing fields.
- The game is complicated enough to have a model representing the game's logic and functionality.
- Most of the user inputs can be done by using clicking events.
- The game does not need a server component, and so it can be published at every free web server. This way it can be easily demonstrated to any friends.
- The board game won the "complex board game of the year 2008" award, is very popular and easy to understand. This way it is not difficult to verify whether the program works correctly.

Figure 29 shows a screenshot of the Agricola board game. The package de.tu_freiberg.informatik.vonwenckstern is the root path in the game project hierarchy in Eclipse.

The client package contains:

- Agricola.java,
- AppController.java,
- AppView.java,
- AppView.ui.xml and
- EventBus.java.

The client.event package has the files:

- AddResourceEvent.java,
- BuildFenceEvent.java,
- BuildHouseEvent.java,
- ChildStartsWorkingEvent.java,
- EnableBigAcquisitionEvent.java,
- FamilyAdditionEvent.java,
- FamilyAdditionWithout-HouseEvent.java,
- GetBigAcquisitionEvent.java,
- GetBoarEvent.java,
- GetCowEvent.java,
- GetSheepEvent.java,
- NextRoundEvent.java,
- PlayerFieldDoneEvent.java,
- PlowFieldEvent.java,
- PlowFieldSeedEvent.java,
- ResourceModelChangedEvent.java,
- RestaurateAndFenceEvent.java,
- RestaurateEvent.java,
- SeedEvent.java, and
- ShowingDialogEvent.java.

To the client.model package belong the following files:

- AcquisitionCardModel.java,
- BackgroundCard.java,
- BaseFieldModel.java,
- BigAcquisitions.java,
- BigFieldModel.java,
- Child.java,
- FieldCard.java,
- HasAcquisitionCardModel.java,
- HasBaseFieldModel.java,
- Player.java,
- PlayerFieldModel.java,
- PlayerResourceModel.java,
- Resource.java, and
- SmallFieldModel.java.

The client.presenter package contains the files:

- BigAcquisitionsPresenter.java,
- CardFieldPresenter.java,
- InfoViewPresenter.java,
- PlayerFieldPresenter.java,
- Presenter.java,
- ResourcePresenter.java,
- Rounds1To7Presenter.java, and
- Rounds8To14Presenter.java.

The client.resources package has the following files:

- Images.java, and
- many jpg and png images.

To the client.view package belong these files:

- AcquisitionCardRenderer.java,
- BigAcquisitionsFieldView.java,
- BigAcquisitionsFieldView.ui.xml,
- BigFieldRenderer.java,
- CardFieldView.java,
- CardFieldView.ui.xml,
- ChildRenderer.java,
- HasPosition.java
- InfoView.java,
- InfoView.ui.xml,
- PlayerFieldView.java,
- PlayerFieldView.ui.xml,
- Renderer.java,
- ResChildRenderer.java,
- ResourceRenderer.java,
- Rounds1To7View.java,
- Rounds1To7View.ui.xml,
- Rounds8To14View.java,
- Rounds8To14View.ui.xml,
- SmallFieldRenderer.java,
- Tooltip.java,
- TooltipImage.java,
- TooltipImageAcquisitionRender-er.java, and
- TooltipImageChildRenderer.java

The project's package structure above reflects the usage of the Model-View-Presenter pattern. Firstly the model of the game will be explained. The BackgroundCard.java file is an enumeration of the 14 round cards and the five action cards of the game. BigAcquisitions.java file contains the constants for the ten acquisition cards. The Child.java file is a listing of the four col-

ors blue, green, pink, and red of the child stones. The FieldCard.java file is an enumeration of the objects like field, wooden house, and stable, which can be put on the farmyard. The Player.java file is a listing of the four colors a player can choose. The Resource.java file is an enumeration of the possible resources like wood, grain, and boar, which can be collected during the game. The AcquisitionCardModel represents the big acquisition the player bought. The interface HasAcquisitionCardModel is used by view classes having an AcquisitionCardModel. The BaseFieldModel saves the occupation state, the actual number and kind of resources for each round and action card. BigFieldModel extends the BaseFieldModel with the BackgroundCard. HasBaseFieldModel is an interface representing views which have a BaseFieldModel. PlayerResourceModel represents the wooden playing pieces available for the player. SmallFieldModel represents the state of one of the 15 farmyard units. The state stores information such as whether the field is fenced, and if there is a stable; or if the field is a wooden room and how many people are living in it and whether they are at work. PlayerFieldModel represents the player's property, it contains the acquisitions and all 15 farmyard units. The source code of this model displayed in listing A62 shows that this model contains several functions like getPastureInfo, isPasture, breedAnimals and sendPersonToWork controlling the course of the game. This shows that a model class can contain more than only the getter and setter methods of its field variables, which impression could be created if just a look at the simple GWT contacts tutorial has been taken. This is the advantage of this little bit more complex example.

The resources package contains all the images of the board game and the Images interface, which methods represent the pictures. The usage of such an interface has two advantages: After replacing one picture with another, e.g. if it has a better quality, the source code has only to be changed in this interface class, and there is no need to update all the other classes. The second benefit is that the GWT compiler inlines the images when they are small enough, and this will result in a faster loading time of the web application. Section 3.8.1 explains the usage of ClientBundles. The connection between the interface method name and the image file is done by using the Java Source annotation, e.g. @Source("DSC06140.JPG") ImageResource pottery(). The following code Images::pottery().getSafeUri().asString() returns the picture URL.

AcquisitionCardRenderer, BigFieldRenderer, ChildRenderer, ResourceRenderer and ResChildRenderer extend the GWT widget AbsolutePanel and display the model information of AcquisitionCardModel, BigFieldModel, Child information of BaseFieldModel, selected Resource of BaseFieldModel, and both Child and Resource information graphically. The Tooltip class opens a popup panel with more information about a widget when the user moves its mouse over it. Normally this behavior can be achieved by using the HTML title attribute. But this attribute does not allow line breaks or images. The TooltipImage extends the GWT Image widget by adding user information about this picture in form of the Tooltip class. The most interesting code of this class is shown in listing 86. The Java annotation UiChild al-

```
1 @UiChild(limit=1)
2 public void addTooltip(Widget tooltip) {
3   new Tooltip(this, tooltip);
4 }
```

Listing 86: UiChild annotation in TooltipImage
Listing A102 shows the complete source code of this class.

lows adding the widget, which contains the additional user information, to the UiBinder. Since every Image can only have one tooltip widget, the limit is set to one. Listing 87 displays how the addTooltip method in any *.ui.xml file can be called. The tooltip widget, which is an HTML widget in this example, can be added by using the UiBinder property tooltip. It is no coincidence

```
<a:TooltipImage ...>
  <a:tooltip>
    <g:HTML>
      <b>fountain</b> <br/>
      ...
    </g:HTML>
  </a:tooltip>
</a:TooltipImage>
```

Listing 87: Call the addTooltip method in any UiBinder file.
Listing A85 shows a complete source code example.

that the property name is the method name without the add prefix. Both TooltipImageAcquisitionRenderer and TooltipImageChildRenderer extend the TooltipImage class and provide a render method setting the URL of the image depending on the AcquisitionCardModel or BaseFieldModel.

The Rounds8To14.ui.xml file shows, how own widgets like BigFieldRenderer can be used in the UiBinder. Firstly, the client.view package containing the custom widget, has to be imported by adding the attribute xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view' to the ui:UiBinder tag at the beginning of the xml file. The a stands for Agricola, and every time it is used like <a:?></a:?> in this project, then the widget or the widget's property as shown in listing 87 belongs to the client.view or client.view.* package. Since the BigFieldRenderer needs a BigFieldModel in order to create an HTML output, it is necessary to create for each BigFieldRenderer widget its own model by writing the code as shown in listing 88. Because the BigFieldModel needs the constants of the enumeration classes Back-

```
<ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.
    model.BigFieldModel" field="modelBoar">
        <ui:attributes bgCard="{BOAR}" ressource="{R_BOAR}"
            ressourceCount="1" ressourceRoundAddition="1"
            description="get boars"/>
</ui:with>
```

Listing 88: Code example how to create a model in the UiBinder.
In the final Agricola version, we do not create the model in the UiBinder view anymore. There we use an extra model class as displayed in listing A66.

groundCard and Resource, they have to be imported with the following code lines <ui:import field="de.tu_freiberg. informatik.vonwenckstern.client.model.BackgroundCard.*" /> and <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.Resource.*"/>. As explained in section 3.5.4, the ui:field attribute is the variable name of the model and has to be unique in the xml file. The code <a:BigFieldRenderer model="{modelBoar}" /> adds the model variable to the widget. This XML code snippet is equivalent to the following line of Java code (new BigFieldRenderer()).setModel(modelBoar). There are two ways of creating an image with the picture defined in the Images interface. In both ways a GWT instance of the Images interface has to be created with the following code line <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client.resources.Images'/> first. Either <g:HTML> </g:HTML> or <g:Image url="{im.rounds8To14.getSafeUri.asString}" /> can be used to create the image. The official Google documentation at [Goo02b] uses the resource attribute and not the url one. This means the following line <g:Image resource="{im.rounds8To14}"/> should be used. But

this XML code has one big drawback that it is not easily possible to resize the picture by adding height or width attributes, because the resource version will embed the image as CSS background instead of setting the url attribute. As explained in section 3.8.1 the browser cuts parts away instead of rescaling the entire picture to the required size.

The <g:at left="20" top="35"> </g:at> attribute of the AbsolutePanel is the panel position, which is relative of its own one, of the inserted child widget. This technique has been extensively used to put the round cards at the right position above the background images. Figure A67 illustrates the view widgets and their connection to the models. Figure A68 shows the relationship between the views and the presenters in this project. Every view class, which is not just a simple renderer widget, has its own presenter. Every presenter has its own Display interface, which will be implemented by the corresponding view class. The interface is used for the communication between the presenter and its view. This project uses an interface and not a view reference in the presenter to have the opportunity to switch the views without modifying (or with as few as possible modifications in) the presenter class. The project will be extended with a mobile version in the next subsection. This means new mobile views, which implement the same presenter displays as the desktop version, have to be created. In this case there are two views for each presenter. Every presenter class implements the Presenter interface containing a getView function which returns the view widget of the presenter. A comparison of the Presenter interface of this project with the one used in the GWT contacts tutorial shows that the GWT interface has the function go, which takes as parameter a HasWidgets variable. The go version has two disadvantages. The first one is that many panels of older GWT extension libraries like GXT 2.0 do not implement the HasWidgets interface, and so GXT panels cannot be passed as a parameter to the go function. The other disadvantage is that we could not figure work out how to divide the web application into several view parts, where every view has its own presenter and the AppController switches out just one view part and not the entire page, when the go version has been used. One of the following passages about the AppView explains how to switch just one view, when the getView method is used. It may be possible that the go version has advantages in testing the application with JUnit. Apart from the Display interface in InfoViewPresenter, all Display interfaces have at least the two methods asWidget returning the view as widget and the registerHandlers having a ClickHandler object as parameter. The ClickHandler parameter is used to register the click events to the presenter in the view class. These are the only assumptions on the views. This means they consist of widgets, which are able to handle click events as user inputs. Google uses a slightly different way to register several click handlers with its presenter. It prefers the usage of several methods returning the HasClickHandlers interface in their Display interface instead of one registerHandlers method. The disadvantage of this version is that for every button or other widget, which handles click events, one method has to be created. This results in an unnecessary blow up of the Display interface and of the View class. The class code increases, because for every widget, which will be returned in the view implementation of the Display interface method, a UiBinder variable has to be created. And as a result the view class contains many references which may slow down the compiled JavaScript code, just to register events in the presenter class.

The cooperation of presenters and views will be explained at the BigAcquisitionsPresenter and BigAcquisitionsFieldView classes. The presenter class is shown in listing 89. This class defines the Display interface with the two previously mentioned methods and one function with the name hideAcquisition. The constructor needs the view as parameter. After the presenter object has been completely created, the constructor calls in the last line display.registerHandlers(this) method. It tells the view class to register the widget's click handlers to presenter's onClick method. The onClick method is called when the user has selected a big acquisition card. This function does nothing other than redirecting the event to the AppController by using the application's EventBus. More information about the EventBus will be

```
1  public class BigAcquisitionsPresenter implements Presenter,
       ClickHandler {
2    public interface Display {
3      public void hideAcquisition(BigAcquisitions aquisition);
4      public void registerHandlers(ClickHandler p);
5      public Widget asWidget();
6    }
7
8    private Display display = null;
9
10   public BigAcquisitionsPresenter(Display display) {
11     this.display = display;
12     display.registerHandlers(this);
13   }
14
15   @Override
16   public void onClick(ClickEvent event) {
17     if(event.getSource() instanceof HasAcquisitionCardModel) {
18       EventBus.fire(new GetBigAcquisitionEvent(((
             HasAcquisitionCardModel) event.getSource()).getModel()));
19     }
20   }
21
22   @Override
23   public Widget getView() {
24     return display.asWidget();
25   }
26
27   public void hideAcquisition(BigAcquisitions aquisition) {
28     display.hideAcquisition(aquisition);
29   }
30 }
```

Listing 89: BigAcquisitionsPresenter class

given later. The getView method returns the view as widget by calling the Display's asWidget method. The hideAcquisition function tells the view to dismiss the acquisition card which the player had bought before. Listing 90 shows the registerHandlers and hidesAcquisition implementations of the BigAcquisitionsFieldView class.

As displayed in the registerHandlers method, there is no need for any UiBinder variable. The panel variable has the type AbsolutePanel. The UiBinder binds the panel in view's constructor with the code line panel =binder.createAndBindUi(this). All handlers have been registered with the given ClickHandler interface by iterating over all panel's child widgets, which have the type TooltipImageAcquisitionRenderer. This iteration technique is also used to hide the image representing the given acquisition. Another advantage of using a loop to register the handlers and hide the widget is that the game can be extended more easily with more acquisitions by simply adding them in the BigAcquisitionsFieldView.ui.xml file; there is no need to manipulate the BigAcquisitionsFieldView.java file at all. Since the web application has six presenters BigAcquisitionsPresenter, CardFieldPresenter, InfoViewPresenter, PlayerFieldPresenter, Rounds1To7Presenter, and Rounds8To14Presenter, they have to send information to the AppController to share the user inputs with each other. Listing 89 displays the use of the EventBus to fire events to notify the AppController.

```
1  @Override
2  public void registerHandlers(ClickHandler p) {
3    for(int i=0; i<panel.getWidgetCount(); i++) {
4      Widget w = panel.getWidget(i);
5      if(w instanceof TooltipImageAcquisitionRenderer) {
6        ((TooltipImageAcquisitionRenderer) w).addClickHandler(p);
7      }
8    }
9  }
10
11 @Override
12 public void hideAcquisition(BigAcquisitions aquisition) {
13   for(int i=0; i<panel.getWidgetCount(); i++) {
14     Widget w = panel.getWidget(i);
15     if(w instanceof TooltipImageAcquisitionRenderer &&
16         (((TooltipImageAcquisitionRenderer) w).getModel().
              getAcquisition() == aquisition) ) {
17       w.setVisible(false);
18       break;
19     }
20   }
21 }
```

Listing 90: Source code of registerHandlers and hideAcquisition implementations of the BigAcquisitionsFieldView class.

All possible events are located in the package client.event. The java files in this package look very similar. In order to minimize the amount of java files in the client.event package, every file contains the event class, the event handler interface, and the "has event" interface. The idea of having all three classes or interfaces in one file is copied from the GXT 3 library [Sen04]. Listing 91 shows the GetBigAcquisitionEvent.java file.

```
1  /** Fires after the user selected a big acquisition */
2  public class GetBigAcquisitionEvent extends GwtEvent<
       GetBigAcquisitionHandler> {
3
4    /** Handler type */
5    private static Type<GetBigAcquisitionHandler> TYPE;
6
7    /** Gets the type associated with this event.
8     * @return returns the handler type
9     */
10   public static Type<GetBigAcquisitionHandler> getType() {
11     if (TYPE == null) {
12       TYPE = new Type<GetBigAcquisitionHandler>();
13     }
14     return TYPE;
15   }
16
17   private AcquisitionCardModel acquisition;
18
19   public GetBigAcquisitionEvent(AcquisitionCardModel acquisition)
       {
20         this.acquisition = acquisition;
21   }
```

```
22
23    public  AcquisitionCardModel getAcquisition() {
24            return acquisition;
25    }
26
27    @SuppressWarnings({"unchecked", "rawtypes"})
28    @Override
29    public Type<GetBigAcquisitionHandler> getAssociatedType() {
30      return (Type) TYPE;
31    }
32
33    @Override
34    protected void dispatch(GetBigAcquisitionHandler handler) {
35      handler.onGetBigAcquisition(this);
36    }
37
38    /** Handler class for {@link GetBigAcquisitionEvent} events. */
39    public interface GetBigAcquisitionHandler extends EventHandler {
40
41      /** Called when a player selected a big acquisition */
42      void onGetBigAcquisition(GetBigAcquisitionEvent event);
43    }
44
45    /** A widget that implements this interface is a public source
          of
46     * {@link GetBigAcquisitionEvent} events. */
47    public interface HasGetBigAcquisitionHandler {
48
49      /** Adds a {@link GetBigAcquisitionHandler} handler for
50       *  {@link GetBigAcquisitionEvent} events.
51       * @param handler the handler
52       * @return the registration for the event */
53      HandlerRegistration addGetBigAcquisitionHandler(
            GetBigAcquisitionHandler handler);
54    }
55  }
```

Listing 91: GetBigAcquisitionEvent.java file has the three classes GetBigAcquisitionEvent, GetBigAcquisitionHandler, and HasGetBigAcquisitionHandler

The easiest way to create a new event java file is to copy an existing event java file as displayed in listing 91. Then the old event name, e.g. GetBigAcquisitionEvent, has to be replaced with the new event name as well as the old handler name like GetBigAcquisitionHandler with the new handler name. If the replace dialog in Eclipse, which is available after pressing control plus F, is used, then only the two above mentioned replacements have to be carried out. The "has handler" interface name like HasGetBigAcquisitionHandler will be replaced automatically. The handler function name at line 35 and 42 has to be changed to the desired one. Now all the copied boiler plate codes have been changed and the focus will be on the actual event class displayed between lines 17 and 25. In the example the event class has one private variable containing the information, which should be sent from the BigAcquisitionsPresenter class to the AppController one. This data is passed to the event class as constructor parameter and cannot be changed, because the event class has only a getter method of this variable. Listing 92 shows the source code of the EventBus class in the project. Since there is only one EventBus in the project, the singleton pattern can be used. Calling EventBus.getEventBus() gives access to the EventBus. In order to shorten the code when firing events from EventBus.getEventBus().fireEvent(...)

```
1  public class EventBus extends HandlerManager implements
2    HasGetBigAcquisitionHandler,... {
3
4    private EventBus() {
5      super(null);
6    }
7    private static EventBus eventBus = new EventBus();
8    public static EventBus getEventBus() {
9      return eventBus;
10   }
11
12   public static void fire(GwtEvent<?> event) {
13     eventBus.fireEvent(event);
14   }
15 @Override
16   public HandlerRegistration addGetBigAcquisitionHandler(
       GetBigAcquisitionHandler handler) {
17     return addHandler(GetBigAcquisitionEvent.getType(), handler);
18   }
19   ...
20 }
```

Listing 92: Source code of EventBus class.

to EventBus.fire(...), the static function fire has been added to this class. The EventBus implements all "has handler" interfaces together with the specific handler registration functions. These methods indicate that the EventBus fires the corresponding events. It is important to fire only those events over the EventBus which are needed to communicate between different presenters. If the presenter received an event from its view and can handle it completely alone, then it should not fire this event over the EventBus, as the other presenters are not interested in this event anyway.

The AppView class is only a container view. This gives it the possibility to switch out different parts of the view. The AppView contains an AbsolutePanel having six SimplePanel widgets as children. One of the six views can be simply changed by setting a new widget to one of the SimplePanel objects. The communication between the AppController and the AppView works the same way as the communication between any presenter and its view. Figure A69 shows that the AppController creates and has links to all other presenters. This figure illustrates that the AppController implements many handler interfaces to receive events which must be shared between different presenters. Listing 93 illustrates the onGetBigAcquisition function, which handles the GetBigAcquisitionEvent described above. After the AppController received the event that a user wants to buy a big acquisition, it tests whether this option is allowed. In Agricola, acquisitions can be only bought if one of the family stones is on the buying card. Later the AppController checks whether the player has enough resources to pay for this purchase. If the AppController grants permission to buy the acquisition, it takes the resources from the player, takes this acquisition out of the store, and adds it to the player field. There are three presenters involved in this action (InfoViewPresenter managing the player's resources, PlayerFieldPresenter adding the new purchase to the player's farm, and BigAcquisitionsPresenter removing the bought object).

The Agricola class is the application's starting point. This class just starts the AppController with AppController app = new AppController(new AppView(), Player.BLUE) and attaches the AppView to the root panel with RootPanel.get().add(app.getView()). The start-

```
1  @Override
2  public void onGetBigAcquisition(GetBigAcquisitionEvent event) {
3    if(!display.isBigAcquisitationFieldEnabled()) {
4      Window.alert("The␣big␣acquisitation␣field␣is␣disabled.\nEnable
           ␣this␣field␣by␣using␣the␣field␣card␣1␣big␣acquisitation␣-␣
           it␣is␣coming␣in␣round␣1,2,3␣or␣4.");
5      return;
6    }
7    BigAcquisitions ba = event.getAcquisition().getAcquisition();
8    if(
9        (ba == BigAcquisitions.BA_FIRE_PLACE) && (resourceModel.
             getClayCount() < 2) ||
10       (ba == BigAcquisitions.BA_FIRE_PLACE2) && (resourceModel.
             getClayCount() < 3) ||
11       ...
12       (ba == BigAcquisitions.BA_BASKET_MAKER) && (resourceModel.
             getReedCount() < 2 || resourceModel.getStoneCount() < 2)
13   ) {
14     Window.alert("You␣have␣not␣enough␣resources␣to␣buy␣the␣big␣
           acquisitation.\nYour␣turn␣is␣over␣now.");
15   } else {
16     switch(ba) {
17     case BA_FIRE_PLACE: resourceModel.addRessource(Resource.R_CLAY
           , -2); break;
18     case BA_FIRE_PLACE2: resourceModel.addRessource(Resource.
           R_CLAY, -3); break;
19     ...
20     case BA_BASKET_MAKER: resourceModel.addRessource(Resource.
           R_REED, -2); resourceModel.addRessource(Resource.R_STONE,
           -2); break;
21     case BA_NONE: break;
22     }
23     ((PlayerFieldPresenter) rightPresenter).addBigAcquisitation(
           event.getAcquisition());
24     ((BigAcquisitionsPresenter) topPresenter).hideAcquisition(ba);
25   }
26   display.setForceBigAcquisitationField(false);
27   ((PlayerFieldPresenter) rightPresenter).
         setGettingBigAcquisitation(false);
28 }
```

Listing 93: Source code of onGetBigAcquisition method in the AppController class.

ing class mostly contains things which must be done before starting the application (launching the AppController). In this example a dialog box, which lets the user choose its favorite color, can be opened. Afterwards the user's selection can be passed as parameter to the AppController.

4.3 Extending the Agricola web application with mobile views

As previously mentioned, the advantage of the Model-View-Presenter pattern is that it gives the opportunity to add a new specific view without changing the model, event, and presenter packages. A new mobile view, which does not load the large images and fits perfectly on iPad 2 screens, should be added. Figure 30 shows the new mobile layout of the Agricola board game.

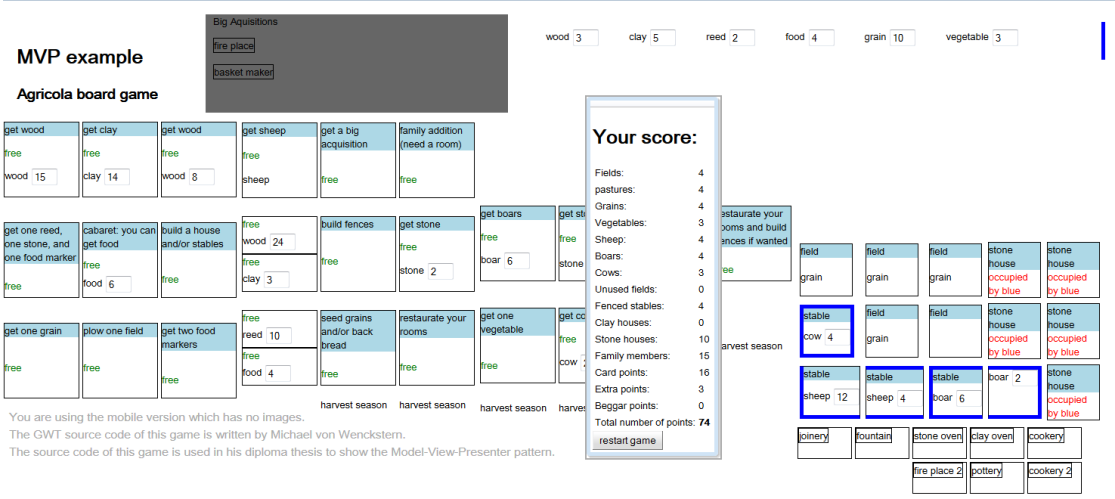


Figure 30: Mobile view of Agricola board game
Take a look at figure 29 to compare it with the desktop version.

Firstly all java files with exception of HasPosition.java, Tooltip.java, and Renderer.java are are copied into the new package client.view.desktop. The client.view.mobile package contains the layout and rendering files specific for the mobile view. Then a new AppViewMobile class has to be created in the client package, which has the general layout of the mobile website. The new structure of the project looks like this:

The client.event, client.model and client.presenter packages are not changed at all. The client package has the files:

- Agricola.java,
- AppController.java,
- AppView.java,
- AppView.ui.xml,
- AppViewMobile.java,
- AppViewMobile.ui.xml, and
- EventBus.java.

The client.view package contains of these following files:

- DesktopViewFactory.java,
- HasPosition.java,
- MobileViewFactory.java,
- Renderer.java,
- Tooltip.java, and
- ViewFactory.java.

To the client.view.desktop package belong these files:

- AcquisitionCardRenderer.java,
- BigAcquisitionsFieldView.java,
- BigAcquisitionsFieldView.ui.xml,
- BigFieldRenderer.java,
- CardFieldView.java,
- CardFieldView.ui.xml,
- ChildRenderer.java,
- InfoView.java,
- InfoView.ui.xml,
- PlayerFieldView.java,
- PlayerFieldView.ui.xml,
- ResChildRenderer.java,
- ResourceRenderer.java,
- Rounds1To7View.java,
- Rounds1To7View.ui.xml,
- Rounds8To14View.java,
- Rounds8To14View.ui.xml,
- SmallFieldRenderer.java,
- TooltipImage.java,
- TooltipImageAcquisitionRenderer.java, and
- TooltipImageChildRenderer.java.

The below listed files belong to the mobile view package client.view.mobile:

- AcquisitionCardRenderer.java,
- BigAcquisitionsFieldView.java,
- BigAcquisitionsFieldView.ui.xml,
- BigFieldRenderer.java,
- CardFieldView.java,
- CardFieldView.ui.xml,
- ChildRenderer.java,
- InfoView.java,
- InfoView.ui.xml,
- LabelAcquisitionRenderer.java
- PlayerFieldView.java,
- PlayerFieldView.ui.xml,
- ResChildRenderer.java,
- ResourceRenderer.java,
- Rounds1To7View.java,
- Rounds1To7View.ui.xml,
- Rounds8To14View.java,
- Rounds8To14View.ui.xml,
- SmallFieldRenderer.java,
- TooltipPanel.java, and
- TooltipPanelChildRenderer.java.

The layout of the mobile view is completely different from the layout of the desktop version. As shown in the *.ui.xml files in the view.desktop package nearly all desktop views make use of the AbsolutePanel. However, most *.ui.xml files in the view.mobile package use the Grid object to layout their widgets. The Rounds1To7View.ui.xml mobile file uses the Grid and the VerticalPanel widget to position its renderer widgets. The registerHandlers method of the mobile version of Rounds1To7View.java is displayed in listing 94. In contrast to the desktop

```
1  public void registerHandlers(ClickHandler p) {
2    for(int r=0; r<grid.getRowCount(); r++) {
3      for(int c=0; c<grid.getCellCount(r); c++) {
4        Widget w = grid.getWidget(r,c);
5        if(w instanceof Renderer) {
6          w.addDomHandler(p, ClickEvent.getType());
7        } else if(w instanceof VerticalPanel) {
8          VerticalPanel vp = (VerticalPanel)w;
9          vp.getWidget(0).addDomHandler(p, ClickEvent.getType());
10         vp.getWidget(1).addDomHandler(p, ClickEvent.getType());
11       }
12     }
13   }
14 }
```

Listing 94: Source code of registerHandlers method in Rounds1To7View.java file.

version, the view cannot just iterate over the children by using one for loop to register the handlers, it now has to iterate over the rows and columns of the grid. The mobile PlayerField.view.ui file uses VerticalPanels, Grid cells, and HorizontalPanels to create the required layout. In this case the iteration has to be done recursively over the main panel. Listing 95 shows the code of the registerHandlers method. Since this code is universal to register handlers of the wanted widgets of the panel, it will be explained in more detail. The panel variable references to the VerticalPanel having the child widgets Grid, HorizontalPanel, and Grid again. This means that _registerHandlers(..., VerticalPanel) starts with the first Grid widget. Since the Grid widget is a container having more child widgets, the function _registerHandlers(..., Grid1) is invoked again. Grid1 contains many SmallFieldRenderer widgets implementing the Renderer interface. Now the click handler can be added to the SmallFieldRenderer objects. Since Grid1 has no container object as child widget, it will not invoke the _registerHandlers method again. After _registerHandlers(...,Grid1) finished, _registerHandlers(..., VerticalPanel) will continue with VerticalPanel's next child widget: the HorizontalPanel. The function _registerHandlers(...,HorizontalPanel) is called, because HorizontalPanel implements the HasWidgets interface. The child widgets of the HorizontalPanel are one Label and two PushButtons,

```
1  public void registerHandlers(ClickHandler p) {
2    _registerHandlers(p, panel);
3    btnEnclosure.addClickHandler(p);
4    btnFeedingFamily.addClickHandler(p);
5  }
6
7  private void _registerHandlers(ClickHandler p, HasWidgets cont) {
8    Iterator<Widget> it = cont.iterator();
9    while(it.hasNext()) {
10     Widget w = it.next();
11     if(w instanceof Renderer) {
12       w.addDomHandler(p, ClickEvent.getType());
13     } else if(w instanceof HasWidgets) {
14       _registerHandlers(p, (HasWidgets) w);
15     }
16   }
17 }
```

Listing 95: Source code of registerHandlers method in PlayerField.java file.

which means no handler is added to them. After _registerHandlers(..., HorizontalPanel) has returned, the last child widget of the VerticalPanel is inspected: the second Grid widget. This means that _registerHandlers(..., VerticalPanel) invokes the function _registerHandlers(..., Grid2), to iterate over the Grid2's child widgets having the type AcquisitionCardRenderer. Since this type also implements the Renderer interface, the click handler will be added to all Grid2's child widgets. The operative word is that the mobile views implement the same presenter Display interface as the desktop ones. This gives the opportunity to change the views with each other. This is done by defining a ViewFactory. The code is displayed in listing 96. The interface has methods returning a suitable view for each presenter. The utility class of the

```
1  public interface ViewFactory {
2    public AppController.Display getAppView();
3    public BigAcquisitionsPresenter.Display getAcquisitionsView();
4    public ResourcePresenter.Display getCardFieldView();
5    public InfoViewPresenter.Display getInfoView();
6    public PlayerFieldPresenter.Display getPlayerFieldView();
7    public ResourcePresenter.Display getRounds1To7View();
8    public ResourcePresenter.Display getRounds8To14View();
9
10   public static class Util {
11     private static ViewFactory viewfactory = GWT.create(
           ViewFactory.class);
12     public static ViewFactory getViewFactory() {
13       return viewfactory;
14     }
15   }
16 }
```

Listing 96: Source code of ViewFactory class.

interface allows access to the presenters' views at every program place. The following line of code ViewFactory.Util.getViewFactory().getAppView() can be used to access the application's main view. The GWT.create function, which allows defining specific deferred bindings,

creates the ViewFactory object. One binding replaces this interface with the mobile view implementation of this interface, and the other one uses the desktop view implementation instead of this pure interface. Listing 97 shows a part of the DesktopViewFactory and listing 98 shows fragments of the MobileViewFactory. The only difference between these two listings is that

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view;
2
3  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       InfoViewPresenter;
4  import de.tu_freiberg.informatik.vonwenckstern.client.view.desktop
       .InfoView;
5  ...
6
7  public class DesktopViewFactory implements ViewFactory {
8    ...
9    private static final InfoViewPresenter.Display infoView = new
         InfoView();
10
11   @Override
12   public InfoViewPresenter.Display getInfoView() {
13     return infoView;
14   }
15   ...
16 }
```

Listing 97: Source code parts of DesktopViewFactory class.

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view;
2
3  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       InfoViewPresenter;
4  import de.tu_freiberg.informatik.vonwenckstern.client.view.mobile.
       InfoView;
5  ...
6
7  public class MobileViewFactory implements ViewFactory {
8    ...
9    private static final InfoViewPresenter.Display infoView = new
         InfoView();
10
11   @Override
12   public InfoViewPresenter.Display getInfoView() {
13     return infoView;
14   }
15   ...
16 }
```

Listing 98: Source code fragments of MobileViewFactory class.

the DesktopViewFactory imports the InfoView class from the view.**desktop** package and that the MobileViewFactory imports it from the view.**mobile** package. Please bear in mind that the views should not be created outside the ViewFactory in order to get view deferred binding working properly. This means the presenters in the AppController class have to be created with the code displayed in listing 99.

Finally, the rule defining which ViewFactory should be used has to be added in the Agricola.gwt.xml file. Listing 100 shows the code that should be inserted before the </module> tag. Lines 1 to 3 replace the ViewFactory interface with the DesktopViewFactory class. This

```
1 ViewFactory vf = ViewFactory.Util.getViewFactory();
2 leftPresenter = new CardFieldPresenter(vf.getCardFieldView(),
      player);
3 middlePresenter = new Rounds1To7Presenter(vf.getRounds1To7View(),
      player);
4 infoPresenter = new InfoViewPresenter(vf.getInfoView(),
      resourceModel);
5 rightPresenter = new PlayerFieldPresenter(vf.getPlayerFieldView(),
       playerModel, resourceModel);
6 bottomPresenter = new Rounds8To14Presenter(vf.getRounds8To14View()
      , player);
7 topPresenter = new BigAcquisitionsPresenter(vf.getAcquisitionsView
      ());
```

Listing 99: Source code showing how to create presenters in AppController class.

```
1 <replace-with class="de.tu_freiberg.informatik.vonwenckstern.
      client.view.DesktopViewFactory">
2     <when-type-is class="de.tu_freiberg.informatik.vonwenckstern.
          client.view.ViewFactory" />
3 </replace-with>
4 <!-- Smart phone detection -->
5 <define-property name="ismobile" values="yes,no" />
6 <property-provider name="ismobile"><![CDATA[
7 return (navigator.userAgent.indexOf('iPhone') > -1
8 || navigator.userAgent.indexOf('Android') > -1
9 || navigator.userAgent.indexOf('iPad') > -1) ? 'yes' : 'no';
10 ]]>
11 </property-provider>
12 <replace-with class="de.tu_freiberg.informatik.vonwenckstern.
      client.view.MobileViewFactory">
13     <when-type-is class="de.tu_freiberg.informatik.vonwenckstern.
          client.view.ViewFactory" />
14     <when-property-is name="ismobile" value="yes" />
15 </replace-with>
```

Listing 100: Source code defining the deferred binding for the different ViewFactories. Smart phone detection code is copied from the book [Dan10] at page 213

means the compiler changes the following line of code private static ViewFactory viewfactory = GWT.create(ViewFactory.class) to private static ViewFactory viewfactory = new DesktopViewFactory(). Line 5 defines the newly deferred binding property ismobile, which can have the two values yes and no. Lines 6 to 11 contain the JavaScript code, which uses the bootstrapper to decide whether it should deliver the desktop or mobile version of the web page. Compiling this GWT application now needs 12 permutations. Section 3.3.2 describes the different permutations and the entire application loading process in more detail.

4.4 Introducing activities in the Agricola Model-View-Presenter pattern enabling browser history

At this point, the game is still unable to redo a turn, which has been accidently set. This subsection describes how to implement this new feature. It shows how the MVP application supports history management in such a way that the user can use the browser's forward and backward button to navigate through the game.

The following Java files will be added to the project hierarchy: In the client package these files will be inserted:

- HistoryController.java, and
- Utils.java.

The following files will be added to the client.event package:

- HistoryChangedEvent.java,
- RequestHistoryEvent.java, and
- SaveHistoryToURLEvent.java.

Normally there is no need to add files to the client.model package. But since firstly some models in the UiBinder xml files were created, they have to be bundled into separate files. The following files should be inserted:

- BigAcquisitationsModel.java,
- CardFieldModel.java,
- HistoryMap.java,
- HistoryMap_CustomFieldSerializer.java,
- Rounds1To7Model.java, and
- Rounds8To14Model.java.

In the client.presenter package only an interface file has been added:

- Activity.java

Not one file has been created in the client.view, client.view.desktop, and client.view.mobile packages.

The Utils.java file contains only a static equals function which checks the equality of objects even if one or both of them are null. The HistoryController class manages the entire history of the application. Since the game needs only one HistoryController, the class uses the singleton pattern. This class has a HistoryMap, which is a normal serializable HashMap with the key and value types Integer and Serializable. The HistoryController uses this map to store the current history state of an Activity class. The Activity interface extends the Presenter interface and has three additional methods:

1. public Type<?> getActivityKey();
2. public T getActualHistory();
3. public void setActualHistory(T state);

In this example the serializable template parameter T represents the model which uses the presenter class to save its state and to restore its view out of a given state. The getActivityKey returns a unique key for every presenter class. The same code is used to generate individual keys as Google does to generate unique keys for their GWTEvent types. In addition to the HistoryMap, the HistoryController has an extra HashMap<Integer, Activity>, the activityPresenterMap, containing all registered Activity presenters, which are interested in changes of their model. The ArrayList<Integer> activityHistoryChanged saves the ids of the presenters, whose state has been changed since the last backup to the browser's URL. The HistoryController reacts to the following events:

1. The ValueChangeEvent<String> is fired when the browser's URL has been changed, e.g. when the user presses the back or forward button.
2. The HistoryChangedEvent is fired when the current state of a presenter has been changed.

In this case the activity id of the presenter is added to the activityHistoryChanged ArrayList.

3. The RequestHistoryEvent is fired when a presenter wants to know its current URL state. The presenters fire this event at the end of their constructors to receive the actual state. This event is needed because if the browser opens the web application with the URL address#state, then the GWT History class invokes the ValueChangeEvent<String> directly, with the state token as value. But at this time the AppController is unable to create all of the presenters; and so the presenters need to ask for their actual state in their constructors to show the wanted state information.
4. The SaveHistoryToURLEvent is fired when the HistoryController should save all the states of the registered presenters in the activityHistoryChanged list to the browser's URL. This event will be fired every time after the user has done an entire turn with one person. This means the browser chronic contains the model data of all presenters for each step; and this allows the user to redo and undo all of its playing moves.

Listing 101 has the code of the onHistoryChanged and onRequestHistory functions. These

```
1  @Override
2  public void onHistoryChanged(HistoryChangedEvent event) {
3    if(!activityHistoryChanged.contains(event.getActivity().
         getActivityKey().hashCode())) {
4      activityHistoryChanged.add(event.getActivity().getActivityKey
           ().hashCode());
5    }
6  }
7
8  @SuppressWarnings("unchecked")
9  @Override
10 public void onRequestHistory(RequestHistoryEvent event) {
11   Serializable s = historyMap.get(event.getActivity().
         getActivityKey().hashCode());
12   event.getActivity().setActualHistory(s);
13 }
```

Listing 101: Source code showing the onHistoryChanged and onRequestHistory methods of the HistoryController class.

two functions work exactly as described in points two and three in the enumeration above. Listing 102 takes a closer look at the onSaveHistoryToURL method. Lines 4 to 12 actualize the historyMap with the current model states of the activity presenters, which had previously announced their state changes . The changed variable is true, if the new history really differs from the old history one. This check is done as it is possible that the state of a presenter changed twice, so that it does not differ from the last URL backup. Line 16 serializes the historyMap containing the new presenter states to a string. The gwt-versatile serialization library, which is available at [S.], can be used. Since this library can only be downloaded for GWT 2.4, little changes have to be made in the downloaded source code to make it compatible with GWT 2.5 (The 2.5 version is available on the DVD). Listing 103 shows the beginning of the serialized string of the historyMap. Since this string always starts with the serialization signature of the HistoryMap class, "de.tu_freiberg. . . . .HistoryMap/2831883331" will be removed from the beginning of the serialized string in line 18. The loop in line 19 until 21 replaces the serialization signature of the model classes with shorter names. Lines 22 and 23 abbreviate true with T and false with F. The versatile library uses the \! sign as object separator. The \! sign will be replaced with *yy*, because

```
1  @Override
2  public void onSaveHistoryToURL(SaveHistoryToURLEvent event) {
3    boolean changed = false;
4    for(int historyKey: activityHistoryChanged) {
5      Activity a = activityPresenterMap.get(historyKey);
6      if(a != null) {
7        Serializable oldHistory = historyMap.get(historyKey);
8        Serializable history = a.getActualHistory();
9        changed |= (oldHistory == null || !oldHistory.equals(history
             ));
10       historyMap.put(historyKey, history);
11     }
12   }
13   activityHistoryChanged.clear();
14   if(changed) {
15     // the history changed
16     String serialized = Serializer.serialize(historyMap);
17     // removing useless values, because it starts everytime with
           the same
18     serialized = serialized.substring((Serializer.
           getSerializationSignature(HistoryMap.class) + "\\!").length
           ());
19     for(int i=0; i<modelNames.length; i++) {
20       serialized = serialized.replace(modelNames[i], shortNames[i
             ]);
21     }
22     serialized = serialized.replace("\\!true\\!", "\\!T\\!");
23     serialized = serialized.replace("\\!false\\!", "\\!F\\!");
24     oldHistoryToken = URL.encode(serialized.replace("\\!", "yy"));
25     History.newItem(oldHistoryToken, false);
26   }
27 }
```

Listing 102: Source code showing the onSaveHistoryToURL method of the HistoryController class.

```
de.tu_freiberg.informatik.vonwenckstern.client.model.HistoryMap
    /2831883331\!3\!1\!de.tu_freiberg.informatik.vonwenckstern.
    client.model.CardFieldModel/890100682\!de.tu_freiberg.
    informatik.vonwenckstern.client.model.BigFieldModel
    /3442264851\!true\!de.tu_freiberg.informatik.vonwenckstern.
    client.model.Child/3400640457\!0\!de.tu_freiberg.informatik.
    vonwenckstern.client.model.Resource/953721184\!9\!1\!
```

Listing 103: Beginning of the serialized historyMap string.

```
3yy1yyS6MyyS5MyyTyyS7Myy0yyS12Myy9yy1
```

Listing 104: Shortened and encoded URL string of Listing 103

the model data does not contain two ys in a row and the \! sign must be URL encoded and yy not. Listing 104 demonstrates the shortened and encoded URL string of listing 103. Listing 105 displays the most important parts of the onValueChange(ValueChangeEvent<String> event)

function. Line 2 gets the actual URL token, which are all characters after the # sign in the

```
1  public void onValueChange(ValueChangeEvent<String> event) { /* 1 */
2      String token = event.getValue();
3    if (token != null && !token.equals(oldHistoryToken)) { /* 2 */
4      oldHistoryToken = token;
5      if (token.equals("start")) {
6        History.newItem(null, false);
7        Window.Location.reload(); // reload the app
8      } else { /* 3 */
9        String deserialized = URL.decode(token).replace("yy", "\\!");
10       ... // replace other shortcuts to original names
11       HistoryMap newHistoryMap = Serializer.deserialize(deserialized);
12       for(int key : newHistoryMap.keySet()) { /* 4 */
13       if(historyMap.containsKey(key)) { /* 5 */
14           Serializable newHistory = newHistoryMap.get(key);
15           if(!historyMap.get(key).equals(newHistory)) { /* 6 */
16               // history of this activity presenter changed
17               Activity activity = activityPresenterMap.get(key);
18               if(activity != null) { /* 7 */
19               activity.setActualHistory(newHistory);
20    /* 7 */ } /* 6 */ }  /* 5 */ } else if(activityPresenterMap.containsKey
          (key)) { /* 8 */
21           // history is new, because the key is not part of historyMap
22           ... /* 8 */ } /* 4 */ }
23       ...
24       historyMap = newHistoryMap; /* 3 */ }  /* 2 */ }  /* 1 */ }
```

Listing 105: Source code of onValueChange method of the HistoryController class.

browsers URL. Line 3 checks whether the new token is difference from the actual URL token, because if the new token is the same as the old one then the history does not change. If the new token is equals *start*, then the entire web application will be reloaded in line 7. Reloading the entire page is done, because it is easier than restoring all the start values. It is important to reset the URL state as shown in line 6 to avoid infinite loop. Line 9 decodes the URL back to get a normal Unicode string. In line 10 the three ellipses represent the code doing the inverse work to the code displayed in lines 18 to 24 in listing 102. Line 11 deserializes the URL token to the HistoryMap containing the presenters' model data. The loop in lines 12 to 22 iterates over the presenters (the hashmap contains the unique ids of the presenters) and set the models extracted out of the URL to the presenters to let them update their views in line 19.

All of the presenter models have the three methods public void update(T model), public boolean equals(Object o) and public T clone(). Listing 106 gives an example implementation of these functions for the PlayerResourceModel. An update and a clone method has been used to return the model data instead of just returning or changing the model data's reference, because returning the reference will cause the equals method to reply with true for every comparison made in the HistoryController class. If the model's reference has been overwritten instead of using an update function, then all HandlerRegistrations, which have been added to the model to inform the view or presenter after the model data changed, will be lost.

Listing 107 shows the InfoViewPresenter class in place of all other activity presenters (BigAcquisitionsPresenter, CardFieldPresenter, PlayerFieldPresenter, Rounds1To7Presenter, and Rounds8To14Presenter). The presenter registers itself to the model resource change handler to be informed after the model data has changed. Later, the activity presenter adds itself to the HistoryController's activityPresenterMap to get the new model data, which has been extracted out of the URL. In the last line of its constructor, the presenter requests its actual URL model data, which was passed to the web application as loading parameter. The onRe-

```
1  public void update(PlayerResourceModel model) {
2    if(model == null) return;
3    this.woodCount = model.woodCount;
4    ...
5    this.stableCount = model.stableCount;
6  }
7
8  public boolean equals(Object o) {
9    if( !(o instanceof PlayerResourceModel))
10     return false;
11   PlayerResourceModel pm = (PlayerResourceModel)o;
12   return woodCount == pm.woodCount && ... && stableCount ==pm.stableCount;
13 }
14
15 public PlayerResourceModel clone() {
16   PlayerResourceModel pm = new PlayerResourceModel();
17   pm.woodCount = woodCount;
18   ...
19   pm.stableCount = stableCount;
20   return pm;
21 }
```

Listing 106: Source code of public void update(T model), public boolean equals(Object o) and public T clone() methods of the PlayerResourceModel class.

sourceChanged method informs the view to update its content and the HistoryController that the model data has changed. As already mentioned above it is important not to return the actual model reference in the getActualHistory function. In the setActualHistory method, the presenter updates its model with the new data entered by the URL and lets the view display the new model data. Another advantage of using the update method instead of assigning the new reference is that the update method can control the model parts which should be changed. This is useful as only the values which may be changed, should be serialized to the URL. The variables, whose value will always be the same, get the transient keyword, which tells the serializer to ignore them. This means these values are not stored into the URL, and so the transient variables have no assigned value after the URL deserialization process. If the reference had been assigned instead of using the update method, then the transient values like the acquisition card descriptions would have been overridden by the no assigned values and this would result in losing data. The advantage of the Model-View-Presenter pattern is that neither any view classes in the client.view, client.view.desktop, and client.view.mobile packages, nor the AppController with its AppView and AppViewMobile classes have to be changed. Now the Agricola board game web application is well structured, making it easier to integrate further extensions. It is also more user-friendly, because it allows the user to change its playing turn by pressing the back button. The complete source code of the Agricola example is available in section A 3.1 and the game can be tested online at [Micc].

```
1  public class InfoViewPresenter implements Activity<PlayerResourceModel>,
       ResourceModelChangedHandler {
2    public interface Display {
3      public void updateView(PlayerResourceModel model);
4      public Widget asWidget();
5    }
6
7    private Display display = null;
8    private PlayerResourceModel model = null;
9
10   public InfoViewPresenter(Display display, PlayerResourceModel model) {
11     this.display = display;
12     this.model = model;
13     model.addResourceModelChangedHandler(this);
14     HistoryController.getInstance().addActivityPresenter(this);
15     EventBus.fire(new RequestHistoryEvent(this));
16   }
17
18   @Override
19   public Widget getView() {
20     return display.asWidget();
21   }
22
23   @Override
24   public void onResourceChanged(ResourceModelChangedEvent event) {
25     display.updateView(model);
26     EventBus.fire(new HistoryChangedEvent(this));
27   }
28
29   private static Type<InfoViewPresenter> TYPE = new Type<InfoViewPresenter
         >("InfoViewPresenter");
30   @Override
31   public Activity.Type<?> getActivityKey() {
32     return TYPE;
33   }
34
35   @Override
36   public PlayerResourceModel getActualHistory() {
37     return model.clone();
38   }
39
40   @Override
41   public void setActualHistory(PlayerResourceModel state) {
42     model.update(state);
43     display.updateView(model);
44   }
45 }
```

Listing 107: Source code of InfoViewPresenter class.

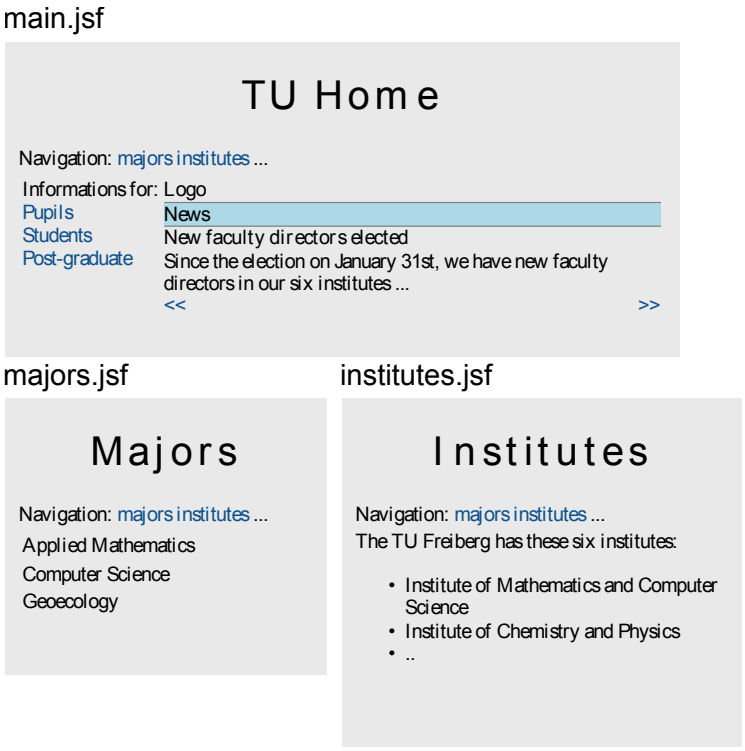


Figure 31: Screenshot of JSF version in category 1.

5 Comparison of the two web frameworks: GWT and JSF

5.1 Definitions of comparison fields

According to point 14 of the Java framework guide from OIO at [Ori02], most Java developers have chosen either JavaServer Faces or Google Web Toolkit in the last two years to create web applications.

That is why this section compares these two technologies with each other. As a first step, the comparison fields have to be defined. Both technologies will be tested in the most common website types:

1. Almost completely static sites with a little bit of dynamic content, e.g. news update.
2. Doing a survey in both technologies.
3. Creating a forum to show data.
4. Writing a chat application.
5. Developing a speed game: Snake.

5.2 Comparison in category 1: Nearly completely static sites with a little bit of dynamic content, e.g. news update.

This subsection begins with the first point and with the JSF version. The website has a similar structure to the homepage of the TU Freiberg at http://tu-freiberg.de/. Figure 31 illustrates the layout of the JSF variant. The main.jsf page has a header acting as navigation bar, and on the left side links for special groups. The middle part of the website displays the latest news from the university. The arrows below the news text can be used to navigate to the next or previous message. Majors.jsf has a list of all possible university courses, which can be attended. The list is created dynamically, so that only a new major has to be added into the database and the jsf file

does not have to be changed. The institutes.jsf page is completely static, because the university will not get a new institute soon.

It is actually really easy to create such a web page with JSF[28]. At first the managed bean class Info, which contains the functions public String[] getMajors(), public int getMaxUniversityNews(), public String decUniversityNews(), public String incUniversityNews(), public String getUniversityNews(), and public int getUniNewsIndex(), has to be created. The function getMajors returns a list of all courses which are available at the university. The method getMaxUniversityNews returns the amount of available news to decide whether the actual news is the last one to hide the next button at the web page. The functions decUniversityNews, and incUniversityNews decrement or increment the index counter to show the previous or next piece of news. The getUniversityNews function returns the actual news depending on the index counter. The method getUniNewsIndex returns the news index counter. The complete source code of this class is displayed in listing A124.

The main.xhtml file contains normal static HTML except of the news part, which is shown in listing 108. The code needs the following html namespaces xmlns:h="http://java.sun.com/

```
1 <h:panelGroup id="uniNews">
2   <h:outputText value="#{info.universityNews}" escape="false"/><br
      />
3   <f:ajax render="uniNews">
4     <div style="float:␣left;"><h:commandLink value="&lt;&lt;"
        action="#{info.decUniversityNews}" rendered="#{info.
        uniNewsIndex␣!=␣0}"/></div>
5     <div style="float:␣right;"><h:commandLink value="&gt;&gt;"
        action="#{info.incUniversityNews}" rendered="#{info.
        uniNewsIndex␣!=␣info.maxUniversityNews-1}"/></div>
6   </f:ajax>
7 </h:panelGroup>
```

Listing 108: XML code of news part in the main.xhtml file.
Complete source code is available in listing A125.

jsf/html", and xmlns:f="http://java.sun.com/jsf/core". Line 2 displays the actual news, the escape attribute has been set to false in order to interpret the value as HTML string and not as text. Since it is not allowed to use < or > letter in an xhtml file, the characters < for the < sign and > for the > symbol have to be used as values in the commandLink object. The action attribute in commandLink or commandButton refers to a method which is called after the user has clicked at the link or button. The method has to return a String describing the page to which the user now navigates. For example: if the String is "main", the user will be redirected to "/main.jsf". The rendered attribute is true, if the server should draw this object. If it should be hidden, then the value must set to false. In the example, the previous arrows are only visible when the news index is not zero. This means the user cannot navigate to the previous news when the first page is displayed. If the commandLink tags were not included in an ajax tag, the browser would reload the entire website in order to update the news part. This behavior would be inconvenient, because only a small part of the homepage should be updated, without reloading the entire site with many images (the TU Freiberg page has many pictures, this example has only a place holder). The render attribute in f:ajax in line 3 tells the server what part of the page should be updated after the user clicked at any link or button inside the f:ajax tag. In this case the browser DOM element with the id uniNews will be updated. Inside the h:panelGroup

[28]Download http://www.coreservlets.com/JSF-Tutorial/jsf2/code/jsf-blank.zip, import it into the Eclipse environment, change only the *.java and *.xhtml files, and run the project on the Tomcat 7 server

tag, which has the id uniNews, are the components which will be refreshed. In this example the news text and the arrows should be brought up to date. This example demonstrates that it is relatively easy to integrate AJAX page updates with JSF.

The JSF code generating a table with all the university courses is displayed in listing 109.

```
1 <h:panelGrid columns="1">
2   <c:forEach items="#{info.majors}" var="item">
3     <h:panelGroup>
4       <h:outputText value="#{item}" />
5     </h:panelGroup>
6   </c:forEach>
7 </h:panelGrid>
```

Listing 109: XML code generating the majors table in the majors.xhtml file.
Complete source code is available in listing A126.

For this code the JSF namespaces xmlns:h="http://java.sun.com/jsf/html", and xmlns:c="http://java.sun.com/jsp/jstl/core" is needed. Line 1 creates a table with one column. Line 2 iterates over the string array containing the majors of the college. Lines 3 and 5 are the beginning and ending of the panelGroup tag defining one table entry. Line 4 writes one string item of the array into the table entry. JSF makes it very easy to display data from a database.

The next paragraphs implement the same behavior in GWT. Since neither the JSF server has to be restarted nor any Java classes need to be compiled to add or change any xhtml file, the static HTML pages cannot be created with the UiBinder or any Java code in GWT as this would result in recompiling the entire GWT application in order to change the content of any static page. The GWT application has to request, parse and modify the HTML sites to achieve the same behavior for static sites in GWT as in JSF. One modification is the change of any internal href string like majors to majors. Without that modification, the browser would leave the GWT application. The parsing process looks for any special tags like [{[universityNews]}] or [{[majors]}], because these tags require data from the server.

Listing 110 shows the source code of the onValueChange function, which loads the HTML page defined as URL history parameter (it is the string after the # sign in the URL). If the URL contains no specific parameter, whose side should be loaded, then the URL will be set to "main.html" in line 4. This is the equivalent to the redirect <% response.sendRedirect ("main.jsf"); %> in the index.jsp file in the JSF project. Line 6 decodes the URL parameter to find out which HTML page should be loaded. Line 10 sends a GET request to the server to receive the content of the HTML website. Line 13 extracts the HTML text from the request and saves it into the variable t. In the next line the hyperlink references will be replaced as mentioned above. Line 18 checks whether the HTML string contains the special code for the news widget. If it contains the special code, then it will be replaced with a normal HTML div tag having a unique id in line 20. The unique id is needed to insert the news widget later. Lines 23 to 27 parses the HTML code for the majors widget displaying all the courses. Line 28 creates an HTMLPanel with the HTML code t, which contains only regular HTML and no special code anymore. Line 30 inserts the news widget into the HTML panel at the tagged place. The same replacement will be done with the Majors widget in line 33. The source code for the UniversityNews (see listing A128) and Majors widgets is a little bit more complex than the JSF code shown in listing 108 and 109. Listing 111 shows the source code of the Majors widget. The constructor loads the courses using the GWT RPC mechanism, and the update method creates the HTML table with the loaded content using the GWT Grid widget.

This example has shown that the GWT implementation is much more complex than the JSF one. The main reason is that a separate parser had to be written and the JSF one could not be

```
1  public void onValueChange(final ValueChangeEvent<String> event) {
2    String url;
3    if(event.getValue() == null || event.getValue().length() == 0) {
4      url = "main.html";
5    } else {
6      url = URL.decode(event.getValue());
7    }
8    RequestBuilder builder = new RequestBuilder(RequestBuilder.POST,
         url);
9    try {
10     builder.sendRequest("GET " + url, new RequestCallback() {
11       @Override
12       public void onResponseReceived(Request request, Response
             response) {
13         String t = response.getText();
14         t = replaceHyperlinks(t);
15         int id = 0;
16         int uniNews = -1;
17         int majors = -1;
18         if(t.contains("[{[universityNews]}]")) {
19           uniNews = id;
20           t = t.replace("[{[universityNews]}]", "<div id=\"gwt-
                 substitute"+id+"\"></div>");
21           id++;
22         }
23         if(t.contains("[{[majors]}]")) {
24           majors = id;
25           t = t.replace("[{[majors]}]", "<div id=\"gwt-substitute"
                 +id+"\"></div>");
26           id++;
27         }
28         HTMLPanel f = new HTMLPanel(t);
29         if(uniNews > -1) {
30           f.add(new UniversityNews(), "gwt-substitute"+uniNews);
31         }
32         if(majors > -1) {
33           f.add(new Majors(), "gwt-substitute"+majors);
34         }
35         RootPanel.get().clear();
36         RootPanel.get().add(f);
37       }
38       @Override
39       public void onError(Request request, Throwable exception) {
40         Window.alert("Failure: Could not load the page:" + URL.
               decode(event.getValue()) + " from the server.");
41       }
42     });
43   } catch (RequestException e) {
44     e.printStackTrace();
45   }
46 }
```

Listing 110: Java code of onValueChange function in InfoSite.java file.
Complete source code is available in listing A127.

```
1  public class Majors extends Grid {
2    private String[] majors = null;
3    public Majors() {
4      InfoService.Util.getInstance().getMajors(new AsyncCallback<
           String[]>() {
5        @Override
6        public void onSuccess(String[] result) {
7          majors = result;
8          update();
9        }
10       @Override
11       public void onFailure(Throwable caught) {
12         Window.alert("Could not load majors.");
13       }
14     });
15   }
16   private void update() {
17     this.resize(majors.length, 1);
18     int i=0;
19     for(String s:majors) {
20       this.setText(i++, 0, s);
21     }
22   }
23 }
```

Listing 111: Java code Majors widget.

used. It would even be possible to parse the JSF xhtml file in GWT so that both technologies can use the same files. It was not done in this example in order to keep the source code as small as possible.

The first category wins JSF clearly with 10:2. GWT gets two points, because it is possible (but with a lot of work) to achieve the same behavior as the JSF application.

JSF 10:
GWT 2:

5.3 Comparison in category 2: Doing a survey in both technologies.

This subsection generates a survey website in both technologies. Since section 3.7 has already implemented a survey homepage in GWT, only the JSF version will be described here. First of all, listing A129 creates the managed bean class Survey with the following fields: String name, String firstName, boolean male, boolean hobbySoccer, boolean hobbyTennis, boolean hobbyBasketball, boolean hobbyVolleyball, boolean hobbyFootball, boolean hobbyBaseball, ArrayList<SelectItem> friends, String newFriend, and int selectedItem. This class has the following command methods: add, getAllFriends, delete, goToPage1, goToPage2, and goToPage3. Listing 112 shows the add method. The SelectItem represents one item in the list

```
1 public String add() {
2   friends.add(new SelectItem(friends.size(), newFriend));
3   newFriend = null;
4   return "page2";
5 }
```

Listing 112: Java code of add method in the Survey.java file.

that represents all the friends of the user. The constructor takes two arguments: the first one is the value, which is connected with the item; the second one is the text, which is displayed to the user. The first argument can be any java Object; the last one must be a java String.

The source code of page1.xhtml, page2.xhtml, and page3.xhtml is very easy to understand. Listing 113 displays the body part of the source code of page1. A normal HTML table is used

```
1 <h:body><h:form>
2 <table><tr><td><h1>Page 1</h1></td><td></td></tr>
3 <tr><td><h2>general information</h2></td><td></td></tr>
4 <tr><td>Name:</td><td><h:inputText value="#{survey.name}"/></td></
      tr>
5 <tr><td>First name:</td><td><h:inputText value="#{survey.firstName
      }"/></td></tr>
6 <tr><td>Sex:</td><td><h:selectOneListbox size="2" value="#{survey.
      male}"><f:selectItem itemValue="true" itemLabel="male"/><
      f:selectItem itemValue="false" itemLabel="female"/></
      h:selectOneListbox></td></tr>
7 <tr><td></td><td> <h:commandButton value="next &gt;&gt;" action="
      #{survey.goToPage2}"/> </td></tr>
8 </table>
9 </h:form></h:body>
```

Listing 113: XML code of body part in page1.xhtml file.

to layout the first survey page. The values of the input elements are automatically loaded from or stored into the managed bean class Survey. This paragraph explains the h:selectOneListbox tag in more detail. A list box has been created which displays two elements at the same time by setting the size attribute to 2. The value of the list box is saved into the boolean variable male. The list box has two selectable items. The first one displays the text "male" to the user and saves the value true into the boolean variable male. The second item illustrates the text "female" and represents the value false to the variable male.

Listing 114 shows the body part of the source code of page2. The user can choose its hobbies by selecting the appropriate checkbox. For example, if the user selects the checkbox with the text soccer, then the boolean variable hobbySoccer has the value true; if the user does not select this checkbox, then variable has the value false. There is a textbox where the user can input the name of its new friend. The friend can be added to the list when the user presses the add button. Listing 112 shows the source code of the add method. All friends are displayed in a list box. Since the list has only a size of ten elements, the user has to scroll if it has more than ten friends. The JSF tag <f:selectItems value="#{survey.friends}"/> says that the ArrayList<SelectItem> friends represents the elements of the list box.

Page3 (listing A130 displays the XML code of page3.xhtml) summarizes the results of the survey. The JSF code <td>Name:</td><td>#{survey.name}</td> displays the user's name. In JSF 2.0 there is no need to use the h:outputText tag to display data. The sex of the person can be shown by using the question mark operator like <td>Sex: </td><td>#{survey.male ? 'male' : 'female'}</td>. The query operator has also been used to list all the user's hobbies by writing <td>Hobbies:</td><td>#{survey.hobbySoccer ? 'soccer, ' : ''} #{survey.hobbyTennis ? 'tennis, ' : ''} ... </td>.

It is very easy to write a survey website in JSF and the developer does not have to bother with history management. Since the survey web page consists of the different pages (page1.jsf, page2.jsf and page3.jsf), the user can navigate through the survey by using the browser's forward and back button. The survey data is available to the user until the browser is closed, because the managed bean class has been annotated with @SessionScoped. Another advantage of using JSF

```
1 <h:body><h:form>
2 <table><tr><td><h1>Page 2</h1></td><td></td></tr>
3 <tr><td><h2>hobbies and friends</h2></td><td></td></tr>
4 <tr><td>Hobbies:</td><td><h:selectBooleanCheckbox value="#{survey.
      hobbySoccer}"/>soccer <h:selectBooleanCheckbox value="#{survey.
      hobbyTennis}"/>tennis <h:selectBooleanCheckbox value="#{survey.
      hobbyBasketball}"/>basketball <h:selectBooleanCheckbox value="
      #{survey.hobbyBaseball}"/>baseball <h:selectBooleanCheckbox
      value="#{survey.hobbyVolleyball}"/>volleyball <
      h:selectBooleanCheckbox value="#{survey.hobbyFootball}"/>
      football</td></tr>
5 <tr><td></td><td><h:inputText value="#{survey.newFriend}"/><
      h:commandButton value="add" action="#{survey.add}"/></td></tr>
6 <tr><td>Friends:</td><td><h:selectOneListbox size="10" value="#{
      survey.selectedItem}"><f:selectItems value="#{survey.friends}"/
      ></h:selectOneListbox></td></tr>
7 <tr><td></td><td><h:commandButton value="delete" action="#{survey.
      delete}"/></td></tr>
8 <tr><td><h:commandButton value="&lt;&lt; previous" action="#{
      survey.goToPage1}"/></td><td> <h:commandButton value="next &gt
      ;&gt;" action="#{survey.goToPage3}"/> </td></tr>
9 </table>
10 </h:form></h:body>
```

Listing 114: XML code of body part in page2.xhtml file.

and not the earlier shown GWT version is that the user cannot accidentally pass personal data to a third person by copying the URL and sending it to anybody else. This is because JSF does not save the user data in the URL like GWT.

This means the second category wins JSF with 10:6. GWT receives 6 points, because it is possible to do a survey and there must not be written as much code as in JSF. But in GWT it can happen that the developer forgets the history support and the user has to re-enter all the data when it presses the back button to change any of its earlier inputs. In GWT the user data can also be stored by using the HTML5 sessionStorage (for more information see [Refc]).The data is only saved as long as the browser tab is open and the data is not saved into the URL. If this technique is used to support the browser history, then GWT has the same behavior and security as the JSF application.

JSF 10: [redacted]
GWT 6: [redacted]

5.4 Comparison in category 3: Creating a forum to show data.

In this subsection JSF runs against GWT in the third category: Writing a forum to show data from a file or a database. In this comparison it does not matter whether the data is loaded from a file or from a database. In both cases the source code is the same, because the server sided part runs in JSF and GWT on Tomcat and the data is returned either by a get function in JSF or by a RPC function in GWT. This means the data reading part is the same in both cases. In order to save the explanation on how to set up a database, the forum data will be loaded from a local file in both cases.

The evaluation starts with the JSF version again. Figure 32 shows the topics.jsf page, which displays the overview of all available topics in the forum and allows the user to create a new

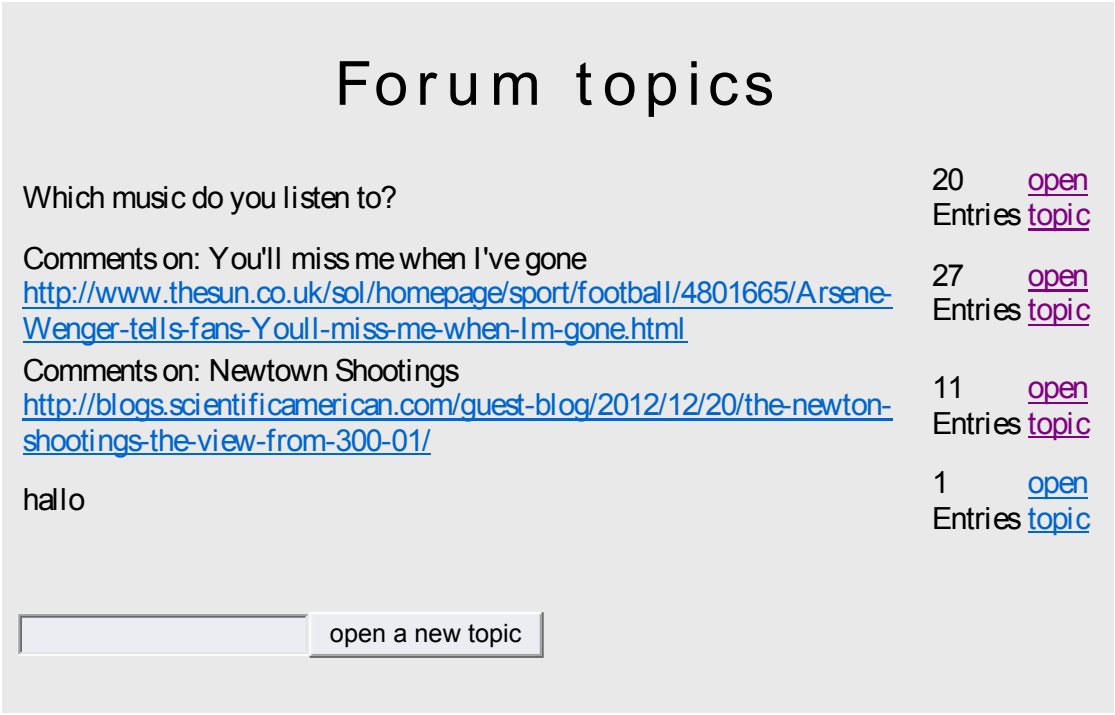


Figure 32: Screenshot of Topic.jsf page.

topic. Figure A70 illustrates the topic entries of "Comments on: You'll miss me when I've gone" [The]. This page also allows adding a new entry to this topic. The managed bean class Forum (see listing A131) is request scoped this time and has the two static sub classes:

- Entry representing a forum topic entry. This class has the field variables String userName, Date time, and String content. The userName represents the nick name the user entered to post its message. The variable time stores the local server time when the post was added to the topic and the variable content saves the posted message.
- Topic represents an entire forum topic. It contains the three field variables String name, int id, and Entry[] entries. The variable name is the topic name, the variable id is needed to access the topic (e.g. when the topic should be loaded then the browser opens a URL like: showtopic.jsf?id=1), the array entries contains all the topic entries.

The Forum class has the following field variables String newTopicName, String userName, String entryText, int topicId, which is connected to the id GET parameter by using the managed property annotation @ManagedProperty(value="#{param.id}"). The Forum constructor loads the topics array out of the topics.data file by using the ObjectInputStream. If a new topic is created, or if a user adds a new entry to an existing topic, then the save method is called, which writes the topics array into the topics.data file with the ObjectOutputStream. The openNewTopic action method is displayed in listing 115 and the addNewEntry method is shown in listing 116. Since line breaks are allowed in the entryText in listings 115 and 116, the HTML key characters<,> and & have to be encoded with the following symbols <, >, and &. If this is not done, the user could enter malicious HTML code, which would be executed; but this way the HTML code will only be displayed and not executed.

The body part source code of topics.xhtml is shown in listing 117. As explained in listing 109, an iteration over the topics array will be done with the c:forEach tag. Since the item.name contains valid HTML code, e.g. hyperlinks, and the hyperlinks should be inter-

```
1  public String openNewTopic() {
2    if(newTopicName != null && newTopicName.length() > 0) {
3      ArrayList<Topic> atopic = new ArrayList<Topic>(Arrays.asList(
           topics));
4      newTopicName = newTopicName.replace("&", "&amp;").replace("<",
           "&lt;").replace(">", "&gt;").replace("\n", "<br>");
5      atopic.add(new Topic(newTopicName, atopic.size(), new Entry[]
           {}));
6      topics = atopic.toArray(topics);
7      save();
8      newTopicName = null;
9    }
10   return "topics";
11 }
```

Listing 115: Java code of the openNewTopic function in the Forum class.

```
1  public String addNewEntry() {
2    if(userName != null && userName.length() > 0 && entryText !=
       null && entryText.length() > 0) {
3      ArrayList<Entry> aentry = new ArrayList<Entry>(Arrays.asList(
         topics[topicId].entries));
4      entryText = entryText.replace("&", "&amp;").replace("<", "&lt;
         ").replace(">", "&gt;").replace("\n", "<br>");
5      aentry.add(new Entry(userName, new Date(), entryText));
6      topics[topicId].entries = aentry.toArray(topics[topicId].
         entries);
7      save();
8      userName = null;
9      entryText = null;
10   }
11   return "showtopic";
12 }
```

Listing 116: Java code of the addNewEntry function in the Forum class.

preted as HTML code and not displayed as text source code, the escape attribute has to be set to false. The other JSF tags have already been explained.

The code of showtopic.xhtml (see listing A132) is very similar to the topics.xhtml one: The c:forEach tag iterates over #{forum.topic.entries} to display the topic entries with a h:panelGrid tag, which has the columns attribute 2.

This example showed that is not difficult at all to create a dynamic forum in JSF. The next passages implement the forum in GWT. The GWT project has three packages: client, shared and server. The client package has the Forum (see listing A133) class, and the two RPC interfaces ForumService (see listing A134) and ForumServiceAsync. The shared package has the classes, which are sent from the server to the client or vice versa. These are the Entry class, Topic class, and TopicInformation class. The Entry and Topic classes are similar to the Forum.Entry and Forum.Topic classes in JSF. The TopicInformation (see listing A135) class contains the name of the topic, the id of the topic and the number of topic entries. This class is used to give a topic overview, meaning there is no need to transfer the entire entries array just to count the length, thus saving network traffic and causing the overview page to be loaded faster. The server package contains only the ForumServiceImpl (see listing A136) class implementing the

```
<h:body><h:form>
<h1>Forum topics</h1>
<h:panelGrid columns="3">
  <c:forEach items="#{forum.topics}" var="item">
    <h:panelGroup>
      <h:outputText value="#{item.name}" escape="false" />
    </h:panelGroup>
    <h:panelGroup>
      #{fn:length(item.entries)} Entries
    </h:panelGroup>
    <h:panelGroup>
      <a href="showtopic.jsf?id=#{item.id}">open topic</a>
    </h:panelGroup>
  </c:forEach>
</h:panelGrid>
<br/><br/>
<h:inputText value="#{forum.newTopicName}"/>
<h:commandButton value="open a new topic" action="#{forum.
    openNewTopic}" />
</h:form></h:body>
```

Listing 117: XML code of body part in topics.xhtml file.

ForumService interface. The constructor, the save, addNewTopic, and addNewEntry methods of this class are very similar to or the same as in the JSF Forum class.

Listing 118 shows the equivalent Java GWT code of the JSF xhtml code, which is displayed in listing 117. The GWT version has to load the topic information array from the server first. After that the data can be illustrated to the user. This code looks very similar to the code in listing 117, because it also iterates over the array and creates a table to show the data with three columns. The only difference is that a topic has to be loaded with the history parameter like "#1" and not with a GET parameter like "?id=1". As the onValueChange method has been used to load the forum topic, the GWT history support has been implemented automatically.

The third category finishes with a tie, because it is very easy in both technologies to write a forum. The reason for this is that the forum is site based, when a new website is loaded for every topic; but the forum is also application based, if the table data of the topic entries table are updated with the new entries of the newly selected topic. GWT is surprisingly good in this category, because the layout of the forum is always the same and so it can be compiled into the static HTML file. This is the main difference between this category and category 1, where the layout of the pages differs a lot (the layout of main.jsf and institutes.jsf was completely different). This is why both technologies earned ten points:

JSF 10:
GWT 10:

5.5 Comparison in category 4: Writing a chat application.

Now category 4 will be compared. In this subsection, a web based chat application should be written. Figure 33 shows the screenshot of the chat application. In order to receive the chat answers directly, the long polling technology is used in JSF and in GWT. This means when the chat client is loaded (after the user entered its login name), the browser sends an update request to the server and the server holds this request until any user sends a new message. After the server has received the new message, it returns the update request. Now the web browser has received the update result, which is empty in this case, but the receive event indicates that new

```
1  RootPanel.get().clear();
2  RootPanel.get().add(new HTML("<h1>Forum topics</h1>"));
3  ForumService.Util.getInstance().getTopics(new AsyncCallback<
       TopicInformation[]>() {
4    @Override
5    public void onSuccess(TopicInformation[] s) {
6      Grid g = new Grid(s.length, 3);
7      for(int i=0; i<s.length; i++) {
8        g.setHTML(i, 0, s[i].getName());
9        g.setText(i, 1, s[i].getNumberOfEntries() + " Entries");
10       g.setHTML(i, 2, "<a href=\"#"+ s[i].getId() + "\">open topic
             </a>");
11     }
12     RootPanel.get().add(g);
13     textbox.setValue(null);
14     btn.setText("open a new topic");
15     HorizontalPanel hp = new HorizontalPanel();
16     hp.add(textbox);
17     hp.add(btn);
18     RootPanel.get().add(hp);
19
20   }
21   @Override
22   public void onFailure(Throwable arg0) {
23     Window.alert("Failure: could not load forum topics");
24   }
25 });
```

Listing 118: Java code excerption of the onValueChange function in the Forum class.

chat messages are available. This means the browser can fetch all the available messages and can send a new update request to be informed when somebody has written a new message.

The JSF version works with IE 10, Chrome 21, and Eclipse Indigo's integrated web browser; but it does not work in Firefox 16. This will not result in a minus point for JSF, it is probably just a programming error on my part. But nevertheless the example shows how the chat application could work in both technologies.

The comparison starts with the JSF version again. The project has three managed bean classes: Chat (see listing A137) which is application scoped, User (see listing A138) which is session scoped, and Message (see listing A139) which is request scoped. The Chat class is started with the server and is available as long as the server is online, which means that there is only one Chat class for all user requests. This class has a synchronized LinkedList<Message> list field variable storing all the message of the different users, and a synchronized HashMap<Integer, User> users to receive user name and gender from the message list's user id. The User class has a static integer variable userIds, which allows giving each user a unique user id stored in the integer variable userId. It also contains a string variable nickName, and a boolean variable isMale. The Message class has a string variable message, an integer variable userId, which is connected to the session's User.userId variable with the managed property @ManagedProperty(value="#{user.userId}"), and a Date time variable.

This paragraph takes a look at the String send and String getUpdate action methods of the Chat class (see listing 119) to give an understanding of how the long polling technique is implemented. The update request calls the getUpdate function. The code synchronize(this) can be used in this class, because it is known that this class exists only once at

Figure 33: Screenshot of JSF chat application

this server. Then the server calls this.wait() to wait until any user sends a new message in line 20. That was all, the method does nothing else except waiting for new messages. The send method is called when any user sends a new message. Since the Message class is request scoped, the actual request's message object can be accessed by calling FacesContext.getCurrentInstance().getExternalContext().getRequestMap().get("message"). In the JSF bean notation the first letter of a class, which should always be a capital letter, has to be changed to a small letter. This is why getRequestMap().get("message") has to be written and not getRequestMap().get("Message"). After the message object has been extracted from the actual request, the addMessage method will be called. This function adds the message to the list and notifies every waiting thread in the getUpdate method that it has received a new message. This causes the threads not to wait anymore, and so the update request returns to the web browser.

The login.xhtml (see listing A140) file contains a <h:inputText value="#{user.nickName}"/> tag where the user can enter its nickname, a <h:selectOneRadio value="#{user.male}"><f:selectItem itemValue="true" itemLabel="male"/><f:selectItem itemValue="false" itemLabel="female"/></h:selectOneRadio> tag where the user selects its gender, and a <h:commandButton action="#{chat.login}" value="login" /> tag where the user finishes the login process. Listing 120 shows the source code of the login function in the Chat class. This function uses .getSessionMap().get("user") to get the user object as the User class is session scoped. After the user object is added to the users map, the web browser will be redirected to chatroom.jsf.

Listing 121 demonstrates the JSF code of chatroom.xhtml. Inside <h:outputScript target="head"> the following JavaScript code is defined: The function f simulates a user click at

```
1  public String send() {
2    Message msg = (Message) FacesContext.getCurrentInstance().
         getExternalContext().getRequestMap().get("message");
3    addMessage(msg.clone());
4    msg.setMessage(null);
5    return "chatroom";
6  }
7
8  public void addMessage(Message msg) {
9    list.add(msg);
10   synchronized (this) {
11     this.notifyAll();
12   }
13 }
14
15 public String getUpdate() {
16   synchronized(this) {
17     boolean cont = true;
18     while(cont) {
19       try {
20         this.wait(); // waits until a new message comes in
21         cont = false;
22       } catch (InterruptedException e) {
23         e.printStackTrace();
24         cont = true;
25       }
26     }
27   }
28   return "chatroom";
29 }
```

Listing 119: Java code of the String send and String getUpdate action methods of the Chat class.

```
1  public String  login() {
2    User user = (User) FacesContext.getCurrentInstance().
         getExternalContext().getSessionMap().get("user");
3    addUser(user);
4    return "chatroom";
5  }
```

Listing 120: Java source code of the login function in the Chat class.

the update button, which results in sending an AJAX update request. This AJAX request calls the getUpdate method of the Chat class. The code window.onload= f means that the function f is invoked after the website is completely loaded. The h:panelGroup tag has the id content and contains the elements which should be updated when the getUpdate request returns. This is the case when any user sends a new message. The <f:ajax render="content" event="change" onevent="f()"> says that the panelGroup should be updated after the update request returned and the event="change" onevent="f()" tells JSF that the function f should be invoked after a successful return of the update event. Normally the <h:commandButton value="send" action="#{chat.send}" /> tag should be inside the f:ajax tag, too. But if so, then nothing happens when the user presses the send button. I wondered why and started the JavaScript de-

```
1 <html xmlns="http://www.w3.org/1999/xhtml"
2   xmlns:h="http://java.sun.com/jsf/html"
3   xmlns:f="http://java.sun.com/jsf/core">
4 <h:head>
5 <h:outputScript target="head">
6 function f() {
7   document.getElementById("formId:update").click();
8   document.getElementById("formId:mSend").value = 0;
9 }
10 function f2() {
11   if(document.getElementById("formId:mSend").value == 1) {
12     f();
13   }
14 }
15
16 window.onload= f;
17 </h:outputScript>
18 <title>Chat room</title>
19 <link href="./css/styles.css"
20   rel="stylesheet" type="text/css"/>
21 </h:head>
22 <h:body>
23 <h1>Chat room</h1>
24 <h:form id="formId">
25 <h:panelGroup id="content">
26 <h:outputText value="#{chat.messages}" escape="false"/><br/>
27 <h:inputTextarea value="#{message.message}" />
28 </h:panelGroup><br/>
29 <f:ajax render="content" event="change" onevent="f()">
30   <h:commandButton value="update" action="#{chat.getUpdate}" id="
       update" />
31 </f:ajax>
32 <h:commandButton value="send" action="#{chat.send}" />
33 </h:form>
34 </h:body></html>
```

Listing 121: XML code chatroom.xhtml file.

bugger. Lines 1324-1329, which is shown in listing 122, in jsf.js display the reason. This means that JSF can, under some circumstances, handle only one AJAX call. But if a new message has been sent to the server using an AJAX call, then there are two active calls in the web browser: The still active update AJAX call, and the newly sent AJAX call. This is the reason why the chat.send call of the send command button is a normal page-reload call and not an AJAX one.

The next passages describe the GWT version of the chat example. There are three packages: client, server, and shared. The client package contains the entry point class Chat (see listing A141) and the two RPC interfaces ChatService (see listing A142) and ChatServiceAsync. The shared package has the classes Message and User. These two classes are very similar to corresponding JSF classes. The server package has only the ChatServiceImpl (see listing A143) class. This class has a static Object lock, which is used in the synchronized block and for the wait and notifyAll methods. This is necessary since nobody knows how many ChatServiceImpl classes Tomcat or Jetty creates.

Listing 123 shows the equivalent GWT source code of listing 121. This code is longer than the xhtml code in the JSF project, but this code is easier to read and write. The sendBtnHandler

```
1322 // if there is already a request on the queue waiting to be
       processed..
1323 // just queue this request
1324 if (!req.que.isEmpty()) {
1325     if (!req.fromQueue) {
1326         req.que.enqueue(req);
1327         return;
1328     }
1329 }
```

Listing 122: JavaScript code in jsf.js.

is passed as ClickHandler to the send button. This means: If the user presses the send button, then the user message will be sent to the server. After receiving the finished event from the server, which means the new message is added to the list, the current history changed event is fired. This causes the invocation of the onValueChange function, which reloads the chat message table. In onValueChange method the message list will be loaded from the server. After receiving this list, the function loadUsers, which loads the user list from the server, will be called. Both lists are needed in order to map the user ids of the messages to a user name. After receiving both messages a user HashMap will be created and the messages can be rendered into a table. This will be appended to the root panel to show the messages to the user. Later on, the server's update function, which only returns after a new message was added to the list, will be called. This means the onSuccess method of the AsyncCallback cb object is invoked after any user sends a new message. In this onSuccess function, the chat message table will be reloaded and the update function will be invoked again. This way the application updates its messages every time when any user sends a new message. At the start of the program, the application registers the value change handler and calls the onValueChange function by invoking the History.fireCurrentHistoryState method.

In the fourth category, GWT narrowly beats JSF by 10:8. JSF gets 8 points because it does not support multiple AJAX calls under some circumstances. GWT does not win more clearly, because with 203 lines of source code (140[29] lines Chat.java + 6[30] ChatService.java + 57[29] ChatServiceImpl.java) it needs twice the amount of source code as the JSF program with 105 lines (65[29] Chat.java + 10[31] login.xhtml + 30[31] chatroom.xhtml).

5.6 Comparison in category 5: Writing the speed game Snake.

Finally the classical snake game will be written in both technologies. The aim of the game is to feed the snake so that it grows. In the game, the player has to collect the red pixels which represent the food with its snake. If the snake bites itself or hits the wall, then it will die and the game is over.

This subsection starts with the JSF version. It has one session scoped managed bean class Snake (see listing A144). This class has the fields ArrayList<Point> snakeList, int keyCode, boolean lost, int iContinue, Point food, and Random rand. The snakeList contains the x and y coordinates of the snake body parts. At the beginning the snake has the small size of three body parts and these are located at the points (2,0), (1,0), and (0,0). The first interesting action

[29] lines containg only the { sign are not counted
[30] the automatic created source code lines of the GWT Eclipse plugin are not counted
[31] DOCTYPE line and xmlns import lines are not counted

```
1  private final ClickHandler sendBtnHandler = new ClickHandler() {
2    public void onClick(ClickEvent event) {
3      Message m = new Message(ta.getValue(), user.getUserId());
4      ChatService.Util.getInstance().addMessage(m, new AsyncCallback
           <Void>() {
5        public void onSuccess(Void result) {
6          History.fireCurrentHistoryState(); // reload chat
7        } // onFailure skipped
8      });
9    }
10 };
11 public void onValueChange(ValueChangeEvent<String> event) {
12   ...
13   ChatService.Util.getInstance().getMessages(new AsyncCallback<
         Message[]>() {
14     public void onSuccess(final Message[] m) {
15       loadUsers(m);
16     } // onFailure skipped
17   });
18 }
19 protected void loadUsers(final Message[] m) {
20   ChatService.Util.getInstance().getUsers(new AsyncCallback<User
         []>() {
21     public void onSuccess(User[] u) {
22       HashMap<Integer, User> users = new HashMap<Integer, User>();
23       for(User us: u) {
24         users.put(us.getUserId(), us);
25       }
26       appendMessages(m, users);
27       AsyncCallback<Void> cb = new AsyncCallback<Void>() {
28         public void onSuccess(Void result) {
29           History.fireCurrentHistoryState(); // reload chat
30         } // onFailure skipped
31       };
32       ChatService.Util.getInstance().update(cb);
33     } // onFailure skipped
34   });
35 }
36 public void onModuleLoad() {
37   History.addValueChangeHandler(this);
38   History.fireCurrentHistoryState();
39 }
```

Listing 123: Java code excerption of GWT Chat class.

method is String getImage, which returns the image of the game. Since I did not figure out how to draw an SVG image with JSF, this function creates a 50x50 table and sets the table cells background color to:

- black, when there is a snake body part at this position,
- red, when the food is located at this position, and
- white, when there is no snake and no food.

The other interesting command function String timer moves the snake in the desired direction. The player can control the direction by pressing the arrow keys up, down, left, and right. This function also detects if the snake bites itself or if it can eat the food.

Up to this point the code was pretty straight forward and easy, but the snake.xhtml file contains more JavaScript code than the actual JSF one. Listing 124 displays the source code of snake.xhtml. Since JSF does not allow calling any managed bean function every 100 ms, this

```
1 <h:head><title>Snake</title>
2 <link href="./css/styles.css"
3       rel="stylesheet" type="text/css"/>
4 <h:outputScript target="head">
5 window.onload= function() {  // inline function 1
6   setInterval(function(){// inline function 2
7     if(document.getElementById("formId:continue").value == 1) {
8       document.getElementById("formId:timerButton").click();
9     }
10  },100);
11  document.getElementById("formId:keyinput").addEventListener("
        keydown", function(event){ // inline function 3
12    document.getElementById("formId:keyCode").value = event.
        keyCode;
13  }, true);
14  document.getElementById("formId:keyinput").focus();
15  document.getElementById("formId:reset").addEventListener("click"
        , function(event){
16    document.getElementById("formId:keyinput").focus();
17  }, true);
18 };
19 </h:outputScript></h:head>
20 <h:body><h:form id="formId">
21 <h1>Snake</h1>
22 <h:inputSecret value="#{snake.input}" style="opacity:␣0.2;␣
      position:␣absolute;␣left:␣100px;␣top:␣300px;␣z-index:␣20;␣
      width:␣250px;␣height:␣250px;" id="keyinput">
23 </h:inputSecret>
24 <h:inputHidden id="keyCode" value="#{snake.keyCode}" /><
      h:inputHidden id="continue" value="#{snake.iContinue}"/>
25 <h:commandButton value="Restart" action="#{snake.restart}" id="
      reset">
26   <f:ajax render="snakeImage␣keyCode" execute="keyCode␣continue"/>
27 </h:commandButton>
28 <h:outputLabel value="#{snake.image}" escape="false" style="
      position:␣absolute;␣left:␣100px;␣top:␣300px;" id="snakeImage">
29 </h:outputLabel>
30 <h:commandButton id="timerButton" action="#{snake.timer}">
31   <f:ajax render="snakeImage" execute="keyCode␣continue"/>
32 </h:commandButton>
33 </h:form></h:body>
```

Listing 124: XML code of snake.xhtml file.

code has to be written in JavaScript. In the window.onload function (it is the inline function 1), a timer will be created which calls every 100 ms inline function 2. The inline function 2 checks whether the game is still being played (this means .value == 1) and if so, it simulates a click on the timerButton. This JSF code looks very much like a hack and it would be nice if JSF had a tag which allows to specify the called function and the interval directly; e.g. <f:timer action="#{snake.timer}" interval="100" repeat="true" />. The keydown eventListener has also to be written in JavaScript, because the AjaxBehaviorEvent as parameter in the JSF lis-

tener function registered with <f:ajax event="keyup" listener="#{listener_function}" /> does not contain any information about the JavaScript event. This means the listener_function is called every time the user presses any key, but there is no way to find out what key the user pressed. This is why the inline function 3 stores the pressed key code into a hidden input field. Since JavaScript only fires key events on elements which have a focus, and the HTML table element cannot gain the focus; an inputSecret tag, which gets the focus and handles the key events, is used. This input element will be put directly over the h:outputLabel tag with CSS absolute position and the transparency will be set to 80% so that it will not hide the snake image completely. The h:inputSecret tag and not the h:inputText tag has to be used, because if the down arrow is pressed inside the h:inputText then the browser suggests the last entered words as popup panel. However the player should not see any popup every time the user wants the snake to go downwards. This JSF code gives the "anything wrong" feeling, because it contains only "hacks". If this code is executed at localhost on the server PC, then it runs fluently. But if it runs on another PC, which is connected with the server over 54 MBit wireless LAN, then the game does not run fluently at all. The game cannot even be played correctly; because if the gamer presses the down key and then the left key the snake does not go down first, it goes directly left and the game is over. The reason for this behavior is that the server is not fast enough in handling the draw event every 100ms. It therefor ignores the fact that the player pressed the down key, and continues right away with the left key.

The GWT implementation needs only the one Snake class, which is shown in listing A145. In the onModuleLoad function a timer has been created called the timer method, which is the same as in the JSF version as has the render function, that creates the table output every 100 ms. In GWT the onKeyDown function returns the key code of the pressed key. Since this GWT example only needs the server to deliver the HTML file once, then the game runs completely in JavaScript on the local PC, it is fast and does not judder at all.

The last category is won clearly by GWT with 10:0. JSF gets no points, because the game does not even work in the same WLAN, and so it will not work if the player is even further away. In the first category, GWT got 2 points, because it was possible to achieve the same behavior in GWT by writing a renderer and the website could display new static content without recompiling the GWT project. In this category the JSF file had to be extended with a lot of JavaScript code, and even with the additional JavaScript code, the game does not run fluently, and so it is not playable at all.

5.7 Summary

Figure 34 illustrates the points earned by Google Web Toolkit and JavaServer Faces in comparison categories 1 to 5.

JSF has ten points in the categories one to three. All these categories have in common:

- The website displays static or dynamic content to the user.
- The project contains many different static or dynamic web pages.
- The user can navigate through the different websites with the browsers back and/or forward button.
- Web pages use traditional or AJAX based form submissions to send data to the server.
- Most times the web application reacts to click events and not to any other JavaScript events like key pressed, mouse moved, and so on.

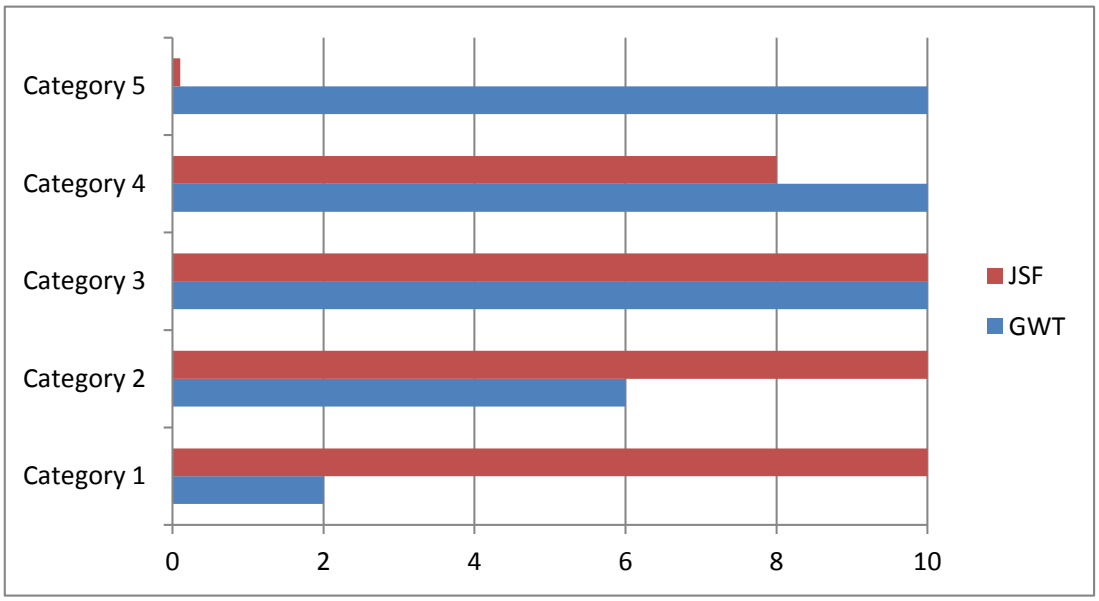


Figure 34: Points earned by Google Web Toolkit and JavaServer Faces in Comparison categories 1 to 5.

GWT won in the categories four and five. These two categories have in common:

- The entire web application has only one web page most of the time.
- The web page has almost no static content; and if it contains dynamic content, then the layout of the dynamic websites is very similar.
- The history management was mostly used to save application data; and not to navigate from one web page to another. If the user navigates with the back or forward button through the web application, then the GWT website will not load a new web page, it only refreshes the content of the same web application.
- The website does not use form based submissions at all to send data to the server. The communication between server and web browser is done by using GWT RPC mechanism, which invokes many asynchronous post calls to the server and calls other JavaScript functions to manipulate the website after receiving the server's answer.
- All the websites look and behave like traditional desktop application. The web application can easily handle all kinds of JavaScript events like key pressed, mouse moved, focus gained or lost, and so on.

It is not possible to state a clear winning technology, because JSF 2.0 and GWT 2.5 have been developed for different application fields. The technology has to be chosen according to the website's purpose: JSF if the website is form based and satisfies the characteristics of categories one to three; and GWT if the web page looks and acts like a common desktop application like MS Office.

This means the JSF framework cannot be used as a GWT substitute to create desktop-like web applications.

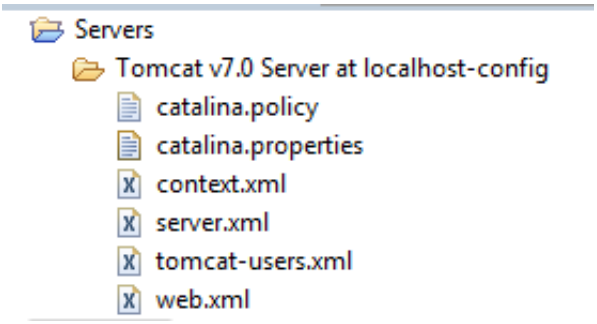


Figure 35: Servers directory in Eclipse

6 Security

6.1 Download Tomcat

Firstly the Tomcat 32-bit or 64-bit Windows zip version has to be downloaded from the official Apache Tomcat page at [Apa]. It is important not to download the Windows Service installer as it is more difficult to configure Eclipse using Tomcat as service. Afterwards the downloaded zip file has to be extracted, e.g. in the eclipse folder to have both Eclipse and Tomcat in the same folder. After which a new server[32] has to be created in Eclipse with the entries given in table 10. If the configuration has succeeded, then the files shown in figure 35 have been created in the

Table 10: Steps how to create a new Server in Eclipse

Server adapter:	Apache –> Tomcat v7.0 Server
Server's host name:	localhost
Server runtime environment:	Add . . . –> Name: Apache Tomcat v7.0 Tomcat installation directory: ...\eclipse\apache-tomcat-7.0.25 JRE: Workbench default JRE

Eclipse Package Explorer. The files context.xml, server.xml, and web.xml will be manipulated later to add user session configuration, database access, and manage the user logins.

6.2 Dynamic Web Application Project with GWT and Tomcat

Firstly, a Dynamic Web Project[33] has to be created in Eclipse with the entries given in tables 11 and 12.

Table 11: Dynamic Web Project wizard part 1.

Project name:	DA_TomcatGWT
Use default location	checked
Target runtime	Apache Tomcat v7.0
Configuration	Default Configuration for Apache Tomcat v7.0

[32] File -> New -> Other ... -> Server -> Server
[33] File -> New -> Project ... -> Web -> Dynamic Web Project

Table 12: Dynamic Web Project wizard part 2.

Context root:	DA_TomcatGWT
Content directory:	WebContent
Generate web.xml deployment descriptor	**checked**

Secondly, the project has to be converted into a GWT one[34]. Thirdly, a GWT Module[35] has to be added in order to compile the GWT project. The GWT module properties are listed in table 13. Since the GWT project should run on Tomcat and not on Jetty, the Run Configuration[36] has

Table 13: GWT module properties.

Source folder:	DA_TomcatGWT/src
Module name:	TomcatGWT
Package name:	de.tu_freiberg.informatik.vonwenckstern
Create EntryPoint and public resources	checked
Use MVP framework	not checked
Configure for third-party toolkit	Use standard GWT only

to be configured as shown in table 14.

In order to create the first static files, the project should be compiled before it runs on Tomcat as shown in figure 36.

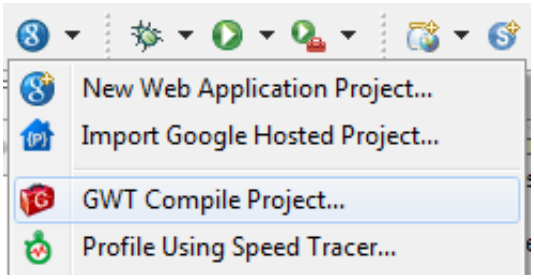


Figure 36: Screenshot how to compile the GWT project

If an error occurs, make sure that the GWT SDK library is above the Apache Tomcat library in the build path configuration[37] as shown in figure 37.

Before the GWT developer mode can be used, the Tomcat-Server[38] has to be started and the entries as shown in table 15 should be selected.

The GWT project should now be run in the development mode[39]. Because the GWT project has been started for the first time in development mode, the war directory, which is located at .../DA_TomcatGWT/WebContent, has to be selected.

[34] right click with mouse on DA_TomcatGWT project folder in Eclipse Project Explorer -> Google Web Toolkit -> Convert project into GWT project ...

[35] right click with mouse on DA_TomcatGWT project folder in Eclipse Project Explorer -> Google Web Toolkit -> GWT module

[36] Run -> Run Configurations -> double click at Web Application

[37] right click with mouse on DA_TomcatGWT project folder in Eclipse Project Explorer -> Properties -> Java Build Path -> Order and Export

[38] right click with mouse on DA_TomcatGWT project folder in Eclipse Project Explorer -> Run As -> Run on Server

[39] Run -> Run History -> TomcatGWT

Table 14: Wizard Run Configurations

Name:	TomcatGWT
Tab Main:	Project: DA_TomcatGWT Main class: **com.google.gwt.dev.DevMode** Include system libraries when searching for main class: Not checked Include inherited mains when searching for main class: Not checked Stop in main: Not checked
Tab Server:	Embedded Server: Run build-in server: **Not checked**
Tab GWT:	URL: **http://localhost:8080/DA_TomcatGWT/TomcatGWT.html** Log level: Info Code Server Port: 9997 Automatically select an unused port: Not checked Available Modules: TomcatGWT – de.tu_freiberg.informatik.vonwenckstern
Tab Common:	Save as: Local file Display in favorites menu: Debug: checked, Run: checked Allocate console: checked Launch in background: checked Encoding: Default – inherited **(UTF-8)**

Table 15: Wizard showing up when Tomcat is started the first time.

How do you want to select the server?	Choose an existing server
Server	Localhost -> Tomcat v7.0 Server at localhost
Always use this server when running this project	**checked**

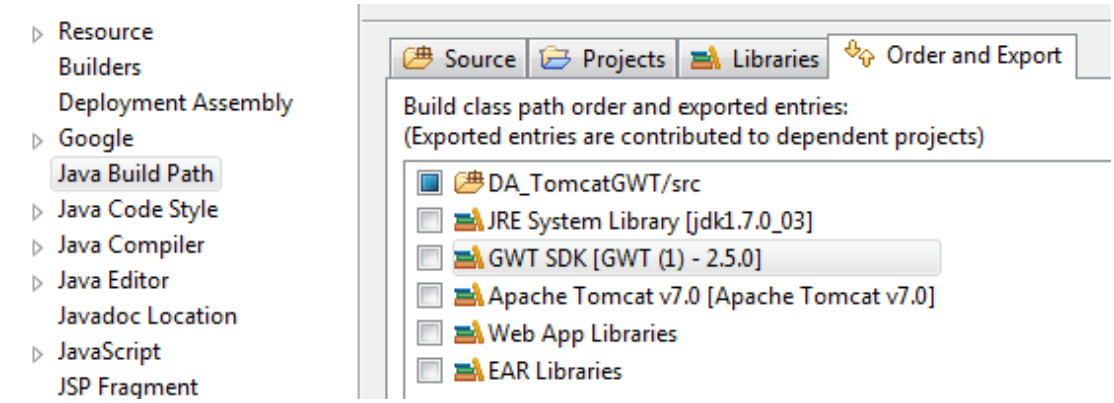


Figure 37: Screenshot of build path configuration

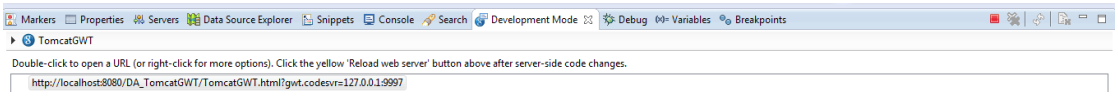


Figure 38: Screenshot showing how to terminate the Development Mode.

After the Development Mode is started, the URL can be copied by clicking with the right mouse button at the link and selecting the copy menu entry. If this URL has been inserted into the web browser and the page stays blank for a while, then the following steps have to be taken:

1. Open Google Web Application property[40] and select the entries shown in the table below:

This project has a WAR directory	**checked**
WAR directory	WebContent Run: checked
Launch and deploy from this directory	**checked**

2. Terminate the Development Mode by pressing the red square shown in figure 38.
3. Compile the GWT web application again as shown in figure 36.
4. Launch the development mode[39] again.

The client side of the GWT application can be debugged as usually by setting a breakpoint at the required line, e.g. at $\mathrm{clickMeButton} = \mathrm{new}\ \mathrm{Button}()$ in TomcatGWT.java, and launching the development mode in debug modus[41]. Finally, the shown URL should be inserted in the web browser and the debugger stops at the breakpoint. To debug the server side part of the GWT application, Tomcat has to be launched in debug mode[42].

After opening the Server tab, double clicking at Tomcat v7.0 Server at localhost and changing the Publishing settings as shown in figure 39, the browser website has only to be reloaded after changing any Java code in development mode and there is no need to relaunch the GWT Development mode every time.

[40] right click with mouse on DA_TomcatGWT project folder in Eclipse Project Explorer -> Properties -> Google -> Web Application

[41] Run -> Debug History -> TomcatGWT

[42] right click with mouse on DA_TomcatGWT project folder in Eclipse Project Explorer -> Debug As -> Debug on Server

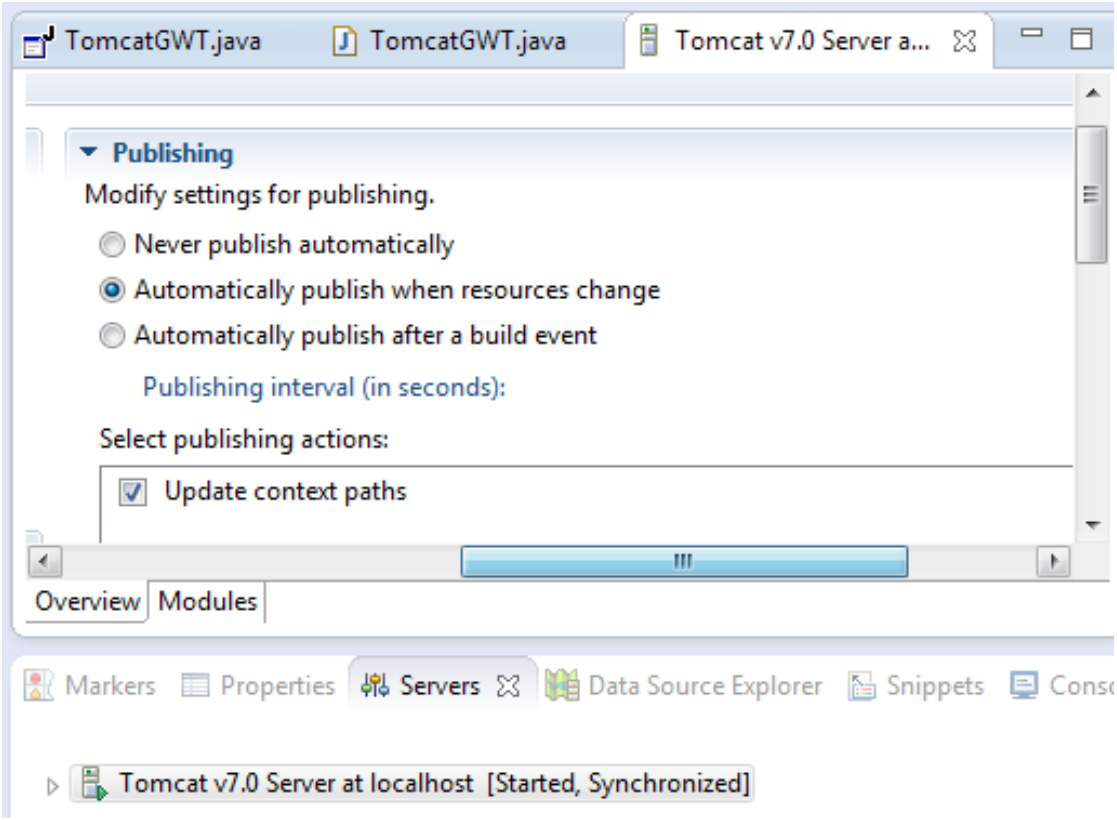


Figure 39: Screenshot of Tomcat's publishing settings

6.3 Establish HTTPS connections in Tomcat

6.3.1 Create a pem certificate

First a pem certificate is needed. StartCom [Sta] offers free and verified certificates. If a pem certificate is already available, e.g. the above mentioned one, then this part can be skipped and continue with section part 6.3.2.

This subsection part creates a Hypertext Transfer Protocol Secure (HTTPS) certificate, which is not verified and causes browser warnings. This is done to have a test certificate for this section. Firstly, OpenSSL has to be downladed at [Ope02]. Secondly, the Visual C++ 2008 Redistributables and afterwards the Win32 OpenSSL v1.0.1eLight have to be installed. If the OpenSSL-Win32\bin folder does not contain an openssl.cnf file, then it has to be downloaded at [Fed] and it has to be saved into the bin directory. Later the cmd.exe should be opened in Windows and the cd command can be used to go to the OpenSSL-Win32\bin directory folder. Listing 125 shows how to create a 2048 Bit RSA key.

```
1 openssl genrsa -out TomcatGWT.key 2048
```

Listing 125: DOS command to create a 2048 Bit RSA key

Listing 126 displays the command for the signing request. Finally, the certificate can be self-signed with the command illustrated in listing 127.

```
1 openssl x509 -req -days 3650 -in TomcatGWT.csr -signkey TomcatGWT.
    key -out TomcatGWT.pem
```

Listing 127: DOS command to sign the certificate.

```
bin>openssl req -new -key TomcatGWT.key -out TomcatGWT.csr
Loading 'screen' into random state - done
...
If you enter '.', the field will be left blank.
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Saxony
Locality Name (eg, city) []:Freiberg
Organization Name (eg, company) [Internet Widgits Pty Ltd]:tu-
    freiberg
Organizational Unit Name (eg, section) []:.
Common Name (e.g. server FQDN or YOUR name) []:tomcat-gwt.tu-
    freiberg.de
Email Address []:unknown@tu-freiberg.de
...
```

Listing 126: DOS command to create the signing request

6.3.2 Convert pem certificate into a key store object

Since Tomcat needs a key store object, the certificate has to be converted. Listing 128 shows the first step in how to convert it into a der certificate. The TomcatGWT_cert.der certificate can be

```
openssl pkcs8 -topk8 -nocrypt -in TomcatGWT.key -inform PEM -out
    TomcatGWT_key.der -outform DER
openssl x509 -in TomcatGWT.pem -inform PEM -out TomcatGWT_cert.der
     -outform DER
```

Listing 128: DOS command converting pem certificate.

opened in the Windows Explorer. Windows warns against the certificate as it is self-signed.

The program ImportKey should be downloaded from [Ema] and it can be extracted into the OpenSSL bin folder. Listing 129 shows how to execute this program. If the certificate needs a

```
bin>java ImportKey TomcatGWT_key.der TomcatGWT_cert.der
Using keystore-file : C:\Users\name\keystore.ImportKey
One certificate, no chain.
Key and certificate stored.
Alias:importkey  Password:importkey
```

Listing 129: DOS command how to execute ImportKey program.

validation chain, then it should be downloaded and renamed to chain.pem. Afterwards the following DOS commands have to be executed: openssl x509 -in chain.pem –inform PEM -out chain.der -outform DER and copy /b TomcatGWT_cert.der+chain.der chain_cert.der. The chain_cert.der certificate contains the TomcatGWT_cert.der certificate together with its chain certificates. This step is not needed in this example, because there is no certificate chain for the self-signed certificate.

The file keystore.ImportKey should then be copied into the keytool class file location at the JRE installations folder, e.g. C:\Program Files\Java\jre7\bin. The cmd.exe file should be opened next with administrative rights. After this, the cd command like cd C:\Program Files\Java\jre7\bin should be used to go to the keytool class location. The chain_cert.der certificate (if the certificate needs a certificate chain) can be created with the command shown in listing 130.

Otherwise the keystore password can be changed with the command displayed in listing 131. The password of the key store will not be replaced here, so it is still importkey. The reason for this decision is to avoid confusion about the key store password.

```
keytool -importcert -alias importkey -file chain_cert.der -
    keystore keystore.ImportKey
```

Listing 130: DOS command how to import the chain certificate into the key store.

```
keytool -storepasswd -keystore keyStore.ImportKey
```

Listing 131: DOS command how to change the password of the key store.

6.3.3 Configure Tomcat's XML files to enable HTPPS

The name of keystore.ImportKey has to be changed to TomcatGWT.ImportKey and the file has to be moved into the eclipse folder. The Servers/server.xml file should then be opened in Eclipse, and the <!-- <Connector port ... /> --> lines has to be changed to the one shown in listing 132. Listing 133 demonstrates the lines, which should be added in the Server/server.xml

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
  ciphers="TLS_RSA_WITH_AES_256_CBC_SHA,␣
     TLS_RSA_WITH_AES_128_CBC_SHA"
  maxThreads="150" scheme="https" secure="true"
  clientAuth="false" sslProtocol="TLS"
  keystoreFile="C:\GWT\eclipse\TomcatGWT.ImportKey"
  keystorePass="importkey" />
```

Listing 132: XML Connector code in server.xml.

file. These lines tell Tomcat to redirect any HTTP connection to HTTPS, if HTTPS is supported. Now the DA_TomcatGWT/WebContent/WEB-INF/web.xml file should be opened and the code

```
<Connector connectionTimeout="20000" port="8080" protocol="HTTP
    /1.1" redirectPort="8443"/>
```

Listing 133: Enabling HTTP:8080 redirect to HTTPS:8443.

displayed in listing 134 can be inserted before the </web-app> tag. After that the server has only to be restarted to secure the traffic between any web browser and the server with HTTPS. If a browser opens the website a warning appears, because the self-signed certificate is not valid.

6.4 Establish a database connection in Tomcat

6.4.1 Create TomcatGWT user and schema, and add the table countries

If MySQL has not been downloaded, then the MySQL Community Server and the MySQL Workbench from [Orac] must be downloaded and installed. If in MySQL Workbench another server instance exists, then open it, select Users and Privileges, and new user has to be added by pressing the button Add Account and insert the values shown in table 16. After the MySQL Workbench has been opened and a click at the New Connection link in the home screen has been made, the values as shown in table 17 should be inserted.

```
1  <security-constraint>
2    <web-resource-collection>
3      <web-resource-name>Wildcard means whole app requires
           authentication</web-resource-name>
4      <url-pattern>/*</url-pattern>
5      <http-method>GET</http-method>
6      <http-method>POST</http-method>
7    </web-resource-collection
8    <user-data-constraint>
9      <!-- transport-guarantee: CONFIDENTIAL, INTEGRAL, or NONE -->
10     <transport-guarantee>CONFIDENTIAL</transport-guarantee>
11   </user-data-constraint>
12 </security-constraint>
```

Listing 134: Forcing the web application to use only HTTPS connections

Table 16: Wizard Adding new user

Tab Login:	Login Name: TomcatGWT Limit Connectivity Hosts Matching: localhost Password: TomcatGWT
Tab Administrative Roles:	Select **DBA**, this cause to select everything else.
Tab Account Limits:	Max. Queries: 0 Max. Updates: 0 Max. Connections: 0 Concurrent Connections: 0

Figure 40 illustrates how to open the TomcatGWT connection with a double click. The following Structured Query Language (SQL) command `CREATE SCHEMA tomcatgwt` should be inserted into the open SQL File1 window; afterwards it should be executed[43].

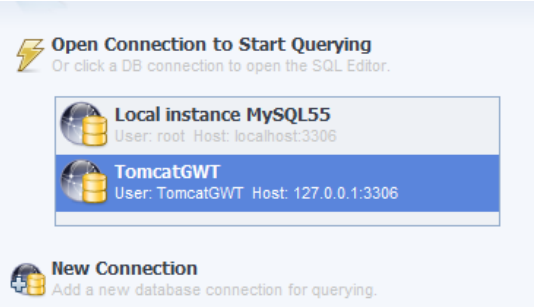


Figure 40: Screenshot showing how to open the TomcatGWT connection in the MySQL Workbench

Since all the administrative rights of the user TomcatGWT were only needed to create the tomcatgwt schema, they should be removed by selecting the TomcatGWT user and pressing the Revoke All Privileges button. Later on, Users and Privileges should be opened again to give TomcatGWT the privileges shown in figure 41. This means the user TomcatGWT can do everything in its own schema and the user has the right to do `SELECT` queries in the information_schema to find out what tables are available in its own schema.

[43] click at the icon with the lightning sign

Table 17: Wizard creating New Connection

Specifiy HostMachine:	localhost
Database Connection:	Connection Name: TomcatGWT Connection Method: Standard (TCP/IP) Tab Parameters: Hostname: localhost Port: 3306 Username: TomcatGWT Password: (nothing)
Testing DB Connection	Everything should be fine: Open Database Connection: OK Get Server Version: 5.5.23 OK Get Server OS: Windows OK Database connection tested successfully
Windows Management	Select the service to manage from the list below.: MySQL55 (Running, Start mode: Auto) Path to Configuration File: C:\ProgramData\MySQL\MySQL Server 5.5\my.ini
Test Settings	Check MySQL configuration file: OK Testing host machine setting is done
Complete Setup	Server Instance Name: TomcatGWT@localhost

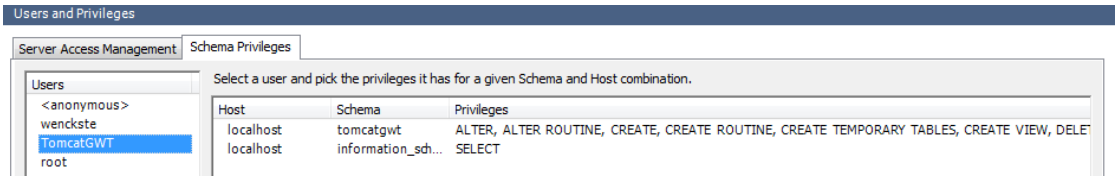


Figure 41: Schema Privileges of TomcatGWT user

Figure 40 illustrates how to open the TomcatGWT connection. After clicking with the right mouse button at the tomcatgwt schema, the menu entry *Set as Default Schema* should be selected. In order to execute queries in the GWT application, a table must be created. The SQL file of all countries in the world can be downloaded from [Gab08]. The countries.sql file can be opened with an arbitrary text editor; the MySQL commands beginning with the CREATE TABLE line and ending with the line ('Zimbabwe', 'zw', 'zwe') can be copied and executed into a new SQL file in the workbench.

6.4.2 Configure Tomcat's XML files to get access to the database connection

The MySQL database connector can be downloaded from [Orad]. The JAR file mysqlconnector-java-5.1.18-bin.jar can be copied into the folder ...\apache-tomcat-7.0.25\lib.

After the Servers/server.xml file has been opened, the lines displayed in listing 135 should be inserted below <GlobalNamingResources>. Listing 136 shows the code, which has to be

```
1 <Resource type="javax.sql.DataSource"
2   name="jdbc/global_TomcatGWT"
3   factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
4   driverClassName="com.mysql.jdbc.Driver"
5   url="jdbc:mysql://localhost/tomcatgwt"
6   username="TomcatGWT"
7   password="TomcatGWT"
8   initialSize="10"
9   maxActive="100"
10  maxIdle="50"
11  minIdle="10"
12  validationQuery="SELECT 1 `id` from `countries` LIMIT 1"
13  testOnBorrow="true"
14  testWhileIdle="true"
15  timeBetweenEvictionRunsMillis="30000"
16 />
```

Listing 135: XML code connecting Tomcat with the database.

```
1 <ResourceLink type="javax.sql.DataSource"
2   name="jdbc/TomcatGWT"
3   global="jdbc/global_TomcatGWT"
4 />
```

Listing 136: Context link to the database server resource.

inserted into Servers/context.xml file after the line <Context>. In order to test the database connection, a simple RPC call, which delivers all the country names in the database, will be created. The remote CountryService interface has the one method String[] getCountries(). At the beginning of the onModuleLoad function in the TomcatGWT.java file, the lines displayed in listing 137 can be added. Listing 138 displays the getConnection function in the CountryServiceImpl remote class. This code returns one MySQL connection from the connection pool, which has been added in listing 135 to Tomcat. The getCountries function implementation is displayed in listing 139. Line 6 tries to get a free database connection. This can fail if there is no free MySQL connection available. Line 7 and 8 creates the MySQL query to get all the country names and its abbreviations in the database. The next line sends this query to the MySQL database, and saves the result in the ResultSet variable rs. The command rs.next() returns true as long as there is still a new data record available which can be read. The Java code rs.getString(1) returns the first column of actual data record, and adds it to the list. It is very important that the MySQL connection is closed in line 16 to avoid all MySQL connections of the pool being occupied.

The TomcatGWT program can now be executed. If the RPC call fails, then the project must be recompiled to add the new getCountries remote function to the GWT whitelist. Figure 42 shows a screenshot of the remote database result.

```
CountryService.Util.getInstance().getCountries(new AsyncCallback<
    String[]>() {
  @Override
  public void onSuccess(String[] countries) {
    String alertStr = "";
    for(String country : countries) {
      alertStr += "\n" + country;
    }
    Window.alert(alertStr);
  }
  @Override
  public void onFailure(Throwable caught) {}
});
```

Listing 137: Java code requesting all countries in the database in the onModuleLoad function

```
public static Connection getConnection() throws Exception {
  // use the name of your database jdni
  String jdbcname = "jdbc/TomcatGWT";
  Context ctx = new InitialContext();
  Context envCtx = (Context)ctx.lookup("java:comp/env");
  DataSource ds = (DataSource)envCtx.lookup(jdbcname);
  Connection conn = ds.getConnection();
  return conn;
}
```

Listing 138: Java code of getConnection function

```
@Override
public String[] getCountries() {
  ArrayList<String> list = new ArrayList<String>();
  Connection con = null;
  try {
    con = getConnection();
    PreparedStatement ps =
      con.prepareStatement ("SELECT␣‘name‘,␣‘alpha_2‘␣from␣‘
          countries‘");
    ResultSet rs = ps.executeQuery();
    while (rs.next()) {
        list.add(rs.getString(1) + "(" + rs.getString(2) + ")");
    }
  } catch(Exception e) { e.printStackTrace();}
  finally {
    if(con != null) {
      try { con.close(); }
      catch (SQLException e) { e.printStackTrace(); }
    }
  }
  String[] s = new String[list.size()];
  list.toArray(s);
  return s;
}
```

Listing 139: Java code of getCountries remote implementation.

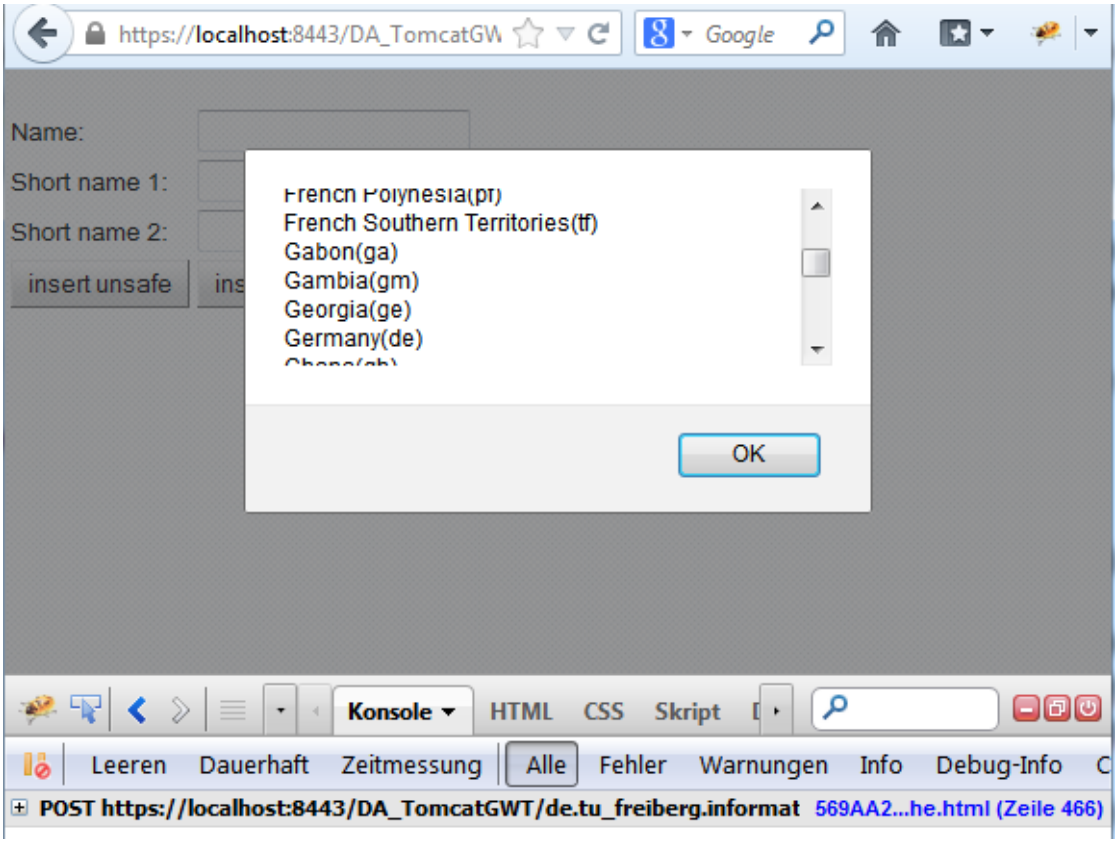


Figure 42: Screenshot of remote database result.

6.4.3 PreparedStatements avoid MySQL injections

This paragraph describes how the user can add a new country into the database. One time the country is added with the unsafe method con.createStatement().executeUpdate(), and the second time it is inserted into the table with the safe function con.prepareStatement().

The two functions have to be added to the CountryService remote interface: void insertCountyUnsafe(String name, String shortName1, String shortName2) and void insertCountySafe(String name, String shortName1, String shortName2). Listing 140 shows the code of the string connected MySQL Statement. Listing 141 displays the equivalent of list-

```
1 public void insertCountyUnsafe(String name, String shortName1,
    String shortName2) {
2   ...
3   con = getConnection();
4   String query = "INSERT␣INTO␣`countries`␣(`name`,␣`alpha_2`,␣`
      alpha_3`)␣VALUES␣('" +
5     name + "',␣'" + shortName1 + "',␣'" + shortName2 + "')";
6   con.createStatement().executeUpdate(query);
7   ...
8 }
```

Listing 140: Java code of unsafe function.

ing 140 with the usage of PreparedStatement. If the user enters in the name field only a text containing no special characters (e.g. my own country), then the behavior is the same in both

```
1  public void insertCountySafe(String name, String shortName1,
       String shortName2) {
2    ...
3    con = getConnection();
4    PreparedStatement ps =
5        con.prepareStatement ("INSERT␣INTO␣`countries`␣(`name`,␣`
             alpha_2`,␣`alpha_3`)␣VALUES␣(?,␣?,␣?);");
6    ps.setString(1, name);
7    ps.setString(2, shortName1);
8    ps.setString(3, shortName2);
9    ps.executeUpdate();
10   ...
11 }
```

Listing 141: Java code of safe function.

functions. But if the text contains any special character like ' (e.g. if the user enters Michael's own country), then the insertCountyUnsafe method will fail, because the String query contains no valid SQL anymore.

In the unsafe method, the user can even inject malicious MySQL, e.g. when the user inputs the code shown in table 18.

Table 18: Example how to inject MySQL code into the unsafe function.
Later on 1 can be replaced with 3, 5, 7, and 9, and 4 can be changed to 1, 2, 3, and so on.

Name:	', (SUBSTRING((select `User` from mysql.user LIMIT 4,1),1,2)),/*'
Short name 1:	(empty)
Short name 2:	*/'

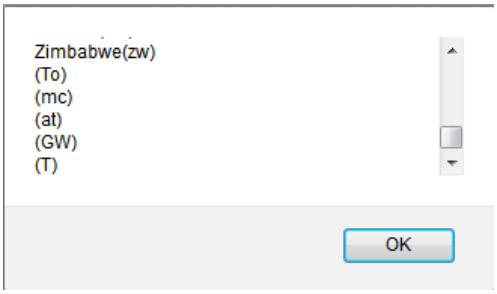


Figure 43: MySQL injection shows secret database login.

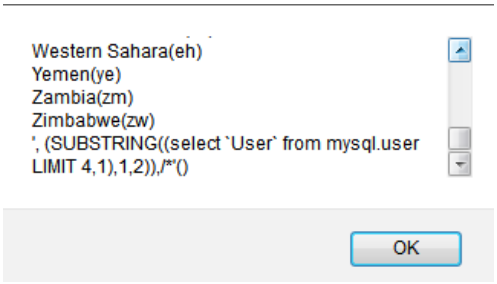


Figure 44: Safe function prevents MySQL injections.

Figure 43 illustrates a screenshot of how the hacker finds out the secret database login "TomcatGWT".If the `User` string is changed to `Password` in table 13, then the hacker knows the hash sum of the password. The attacker now has the administrative user names and the corresponding passwords. It is then possible to change everything in the database. If the attacker injects malicious MySQL in the safe mode, then the text will be inserted into the database as string (see figure 44) and will not be executed as SQL command.

This part finishes with some last pieces of advice. If dynamic tables need to be created, then the user should not choose the table name. The reason is that PreparedStatements cannot be used to select tables. In this case, the table name should be number-based, e.g. t1, t2, t3, and so on. The advantage is that the input parameter can be tested with the java expression "t" +

Integer.toString(tableId) as to whether it is an integer. If it is necessary for the user to choose a table name in the program, then the table id should be only mapped to a name in the MySQL tables. E.g. the entry in the MySQL tables: tableId = 3, name ='whatever', means that the name of user table is whatever.

6.5 Login mechanism in Tomcat

First of all, the tables shown in listing 142 should be created in the MySQL workbench. This

```
1 CREATE TABLE `tomcat_users` (
2   `login` varchar(20) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL
      ,
3   `password` varchar(32) CHARACTER SET utf8 COLLATE utf8_bin NOT
      NULL,
4   PRIMARY KEY (`login`)
5 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
6
7 CREATE TABLE `tomcat_roles` (
8   `role` varchar(20) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
9   PRIMARY KEY (`role`)
10 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
11
12 CREATE TABLE `tomcat_users_roles` (
13   `login` varchar(20) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL
      ,
14   `role` varchar(35) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
15   PRIMARY KEY (`login`,`role`),
16   KEY `tomcat_users_roles_foreign_key_2` (`role`),
17   CONSTRAINT `tomcat_users_roles_foreign_key_1` FOREIGN KEY (`
       login`) REFERENCES `tomcat_users` (`login`),
18   CONSTRAINT `tomcat_users_roles_foreign_key_2` FOREIGN KEY (`role
       `) REFERENCES `tomcat_roles` (`role`)
19 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
20
21 INSERT INTO `tomcatgwt`.`tomcat_users` (`login`, `password`)
     VALUES ('Michael', '3e06fa3927cbdf4e9d93ba4541acce86');
22 INSERT INTO `tomcatgwt`.`tomcat_roles` (`role`) VALUES ('user');
23 INSERT INTO `tomcatgwt`.`tomcat_users_roles` (`login`, `role`)
     VALUES ('Michael', 'user');
```

Listing 142: MySQL code to create the login tables.

means there is one valid user with the login Michael and the password Michael (a real program allows only passwords which have at least two small letters, two capital letters, two numbers, and two special signs like $ or §). Listing 143 displays the code which should be inserted after the line <Realm className="org.apache.catalina.realm.UserDatabaseRealm" resourceName="UserDatabase"/> in the Server/server.xml file. The command digest="md5" means that the user passwords are saved as MD5 hash in the database. Passwords should be never stored in plain text in the database.

After the DA_TomcatGWT/WebContent/WEB-INF/web.xml file has been opened, the code shown in listing 144 should be inserted above </webresource-collection>. This means that only logins with the connected role name "user" get access to the web application. If there is a need for a special administrative site, then the role admin has to be added in the tomcat_roles

```
<Realm  className="org.apache.catalina.realm.JDBCRealm" digest="
    md5"
    driverName="org.gjt.mm.mysql.Driver"
    connectionURL="jdbc:mysql://localhost/tomcatgwt"
    connectionName="TomcatGWT" connectionPassword="TomcatGWT"
    userTable="tomcat_users" userNameCol="login" userCredCol="
        password"
    userRoleTable="tomcat_users_roles" roleNameCol="role"
/>
```

Listing 143: XML code registering users in Tomcat.

```
<auth-constraint>
  <role-name>user</role-name>
</auth-constraint>
```

Listing 144: XML code which grants only users with the role "user" access to our web application.

table and the tomcat_users_roles table should give some user logins the role admin. This allows having sites, which grant only access to admin roles.

After the </security-constraint> line in the web.xml file, the code displayed in listing 145 should be entered. It is important to use a form based login and not the basic login mechanism

```
<login-config>
  <auth-method>FORM</auth-method>
   <form-login-config>
     <form-login-page>/login.jsp</form-login-page>
     <form-error-page>/login.jsp?fail=true</form-error-page>
  </form-login-config>
</login-config>
```

Listing 145: XML code setting up form based login mechanism.

as the basic one does not allow invalidating the user session and so no equivalent logout function exists. Finally, the login.jsp file where the user can enter its login name and password needs to be created. Listing 146 shows the JSP code. In the post form, the action value has to be set to j_security_check, in order to send the login to Tomcat. The login input field has to have the name j_username, and the password field the name j_password, so that Tomcat can extract the login and password values out of the post data. If the Tomcat server starts now and the URL https://localhost:8443/DA_TomcatGWT/TomcatGWT.html has been entered, then the user has to authenticate itself before it gets access to the GWT web application. Figure 45 shows the login screenshot. This means that attackers are not able to see any of the compiled JavaScript code, and so they have no information about the applications structure to find any vulnerability as shown in section 3.6.2 and figure 22. If the person does have the password, then it still has no rights to send or manipulate any GWT RPC calls in order to inject malicious SQL code, because Tomcat will block the RPC call. If Michael will be entered as login name and password, then access to the GWT website will be granted.

The GWT application will be extended so that it shows the login name of the logged-in user, and that it has a logout button. Firstly the users Maren, Oliver, and Stefan have to be added to

```
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO
    -8859-1">
<title>Login</title>
</head>
<body>
  <% if(request.getParameter("fail") != null && request.
      getParameter("fail").equals("true")) { %>
    <color="red"><h1>Login failed</h1></color>
  <% } %>
  <form method="POST" action="j_security_check">
  Login
  <table>
  <tr><td>Login name: </td><td><input type="text" name="j_username
      "></td></tr>
  <tr><td>Password: </td><td><input type="password" name="
      j_password"></td></tr>
  </table>
  <input type="submit" value="Login">
  <input type="reset" value="Reset">
  <br/> <br/>
  In order to use the GWT web application, make sure your browser'
      s JavaScript engine is enabled.
</form></body></html>
```

Listing 146: HTML and Java code of login.jsp file.

the tomcat_users table.[44] In this example the password for every user is the same as its login name, and every user has the same role name. The two functions String getLoginName and void logout have been added to the CountryService remote interface. The implementation of these methods is shown in listing 147.

```
@Override
public String getLoginName() {
  return getThreadLocalRequest().getUserPrincipal().getName();
}

@Override
public void logout() {
  getThreadLocalRequest().getSession().invalidate();
}
```

Listing 147: Java code of getLoginName and logout functions.

The user can be logged out by invalidating the session as the application uses the form based login mechanism. This means that if the user calls the page the first time, then Tomcat gives

[44] The MySQL code is:

```
INSERT INTO `tomcatgwt`.`tomcat_users` (`login`, `password`) VALUES ('Maren',
'144a85d15126fbd41ef4e701df6a8b3c');
INSERT INTO `tomcatgwt`.`tomcat_users_roles` (`login`, `role`) VALUES ('Maren', 'user');
INSERT INTO `tomcatgwt`.`tomcat_users` (`login`, `password`) VALUES ('Oliver',
'27090706d42a2525b9a07222f68dd3d4');
INSERT INTO `tomcatgwt`.`tomcat_users_roles` (`login`, `role`) VALUES ('Oliver', 'user');
INSERT INTO `tomcatgwt`.`tomcat_users` (`login`, `password`) VALUES ('Stefan',
'752b048c76633399b04f6e87a8b211ca');
INSERT INTO `tomcatgwt`.`tomcat_users_roles` (`login`, `role`) VALUES ('Stefan', 'user');
```

Figure 45: Screenshot of login formula

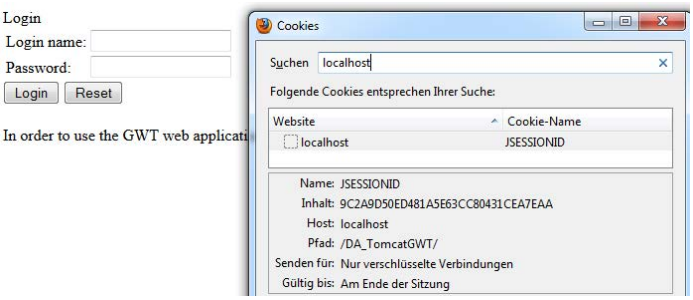


Figure 46: Screenshot of Tomcat's session id in Firefox

it a HTTP session id for the login request (see figure 46). After a successful login, Tomcat gives the browser a new session id, which it sends on every request to the server. The server checks the browser rights based on the HTTP session id. This means if the session id is in its session store, then the browser has access to the page; otherwise Tomcat denies the page access. Invalidating the actual session removes the session id from the store and this will cause an access deny (reloading the login form again) when the user sends any data to the server.

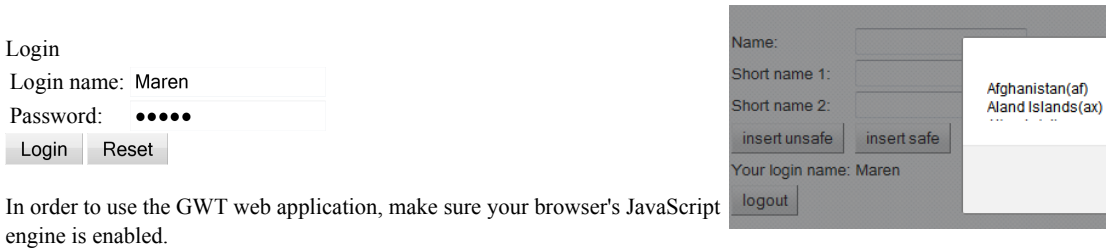


Figure 47: Screenshot of final result with logout button.

Finally, the TomcatGWT class has to be modified in such a way, that it calls the getLoginName remote method in the onModuleLoad function and that it invokes the logout remote method after the user has clicked at the logout button. Figure 47 shows a screenshot of the final result.

The user password should always be sent encrypted over a secure HTTPS connection to the server. If it is not done, then an attacker can read the plain password using a tool like [Wir]. The best way is to secure the entire application traffic with HTTPS avoiding anybody being able to spy into the user's session id: it requires only a little bit more computer resources in modern Intel Xeon PCs, but the security gain is large.

6.6 SafeHtml

This section has shown how the database can be protected against malicious SQL code before; this part demonstrates how to protect the web application against HTML injection now. A new GWT Java Project will be created to show some possible attacks and how they can be prevented. Figure 48 illustrates the layout and result of this page, if the user enters the text shown in listing 148 and presses the button, which will be wrongly shown in the unsafe method.

Listing 149 illustrates the most important part of the source code. The unsafe method simply connects two String objects together. The safe method uses the SafeHtmlBuilder to escape the user's text inputs with the appendEscapedLines function.

```
<input type="button" value="you won 200 dollor" onclick="document.
    getElementsByTagName('body')[0].onmousemove = function() {
  window.alert('you cannot move your mouse anymore without showing
    this message :)');
}">
```

Listing 148: HTML injection code

```
insertTextUnsafe.addClickHandler(new ClickHandler(){
  public void onClick(ClickEvent event) {
    html.setHTML(html.getText() + "<br><hr><br>" + tb.getValue());
  }
});
insertTextSafe.addClickHandler(new ClickHandler(){
  public void onClick(ClickEvent event) {
    html.setHTML(new SafeHtmlBuilder().appendHtmlConstant(html.
        getHTML()+"<br><hr><br>")
      .appendEscapedLines(tb.getValue()).toSafeHtml());
  }
});
```

Listing 149: Java code with and without the usage of SafeHtml.

Of course one may think the attacker can manipulate the browser's DOM with Firebug anyway, why should the user's inputs be escaped. The reason is quite easy: Imagine the user inputs have been saved in a database and the content will be loaded for every user on program start. If the HTML inputs are not escaped, then it can happen that every user can no longer use the GWT program, e.g. when the hacker injects an HTML redirect code.

A more detailed tutorial about SafeHtml can be found at [Goo10f].

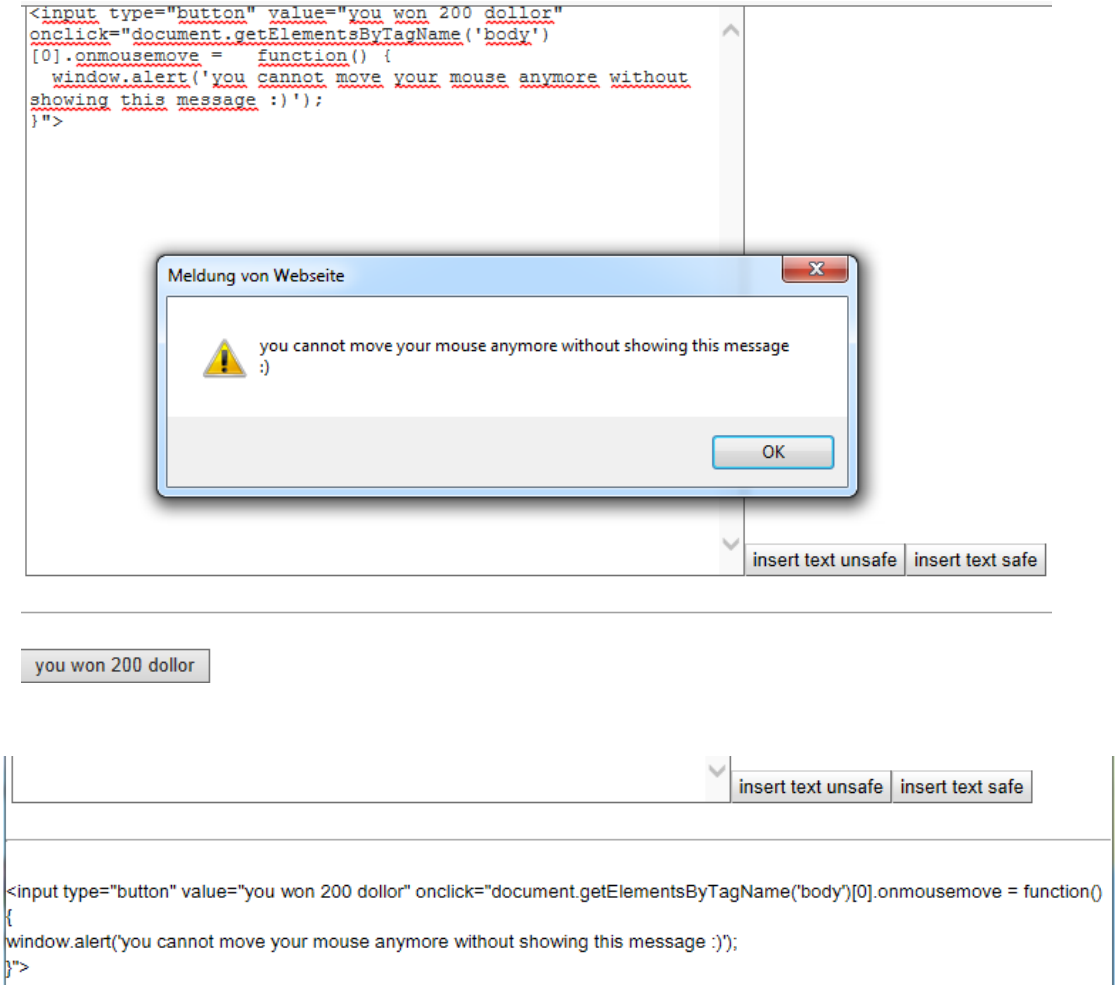


Figure 48: Top picture shows the usage of the unsafe text insert method, which allows HTML injections.
Bottom picture shows the result when the programmer uses the SafeHtml interface to update its widgets.

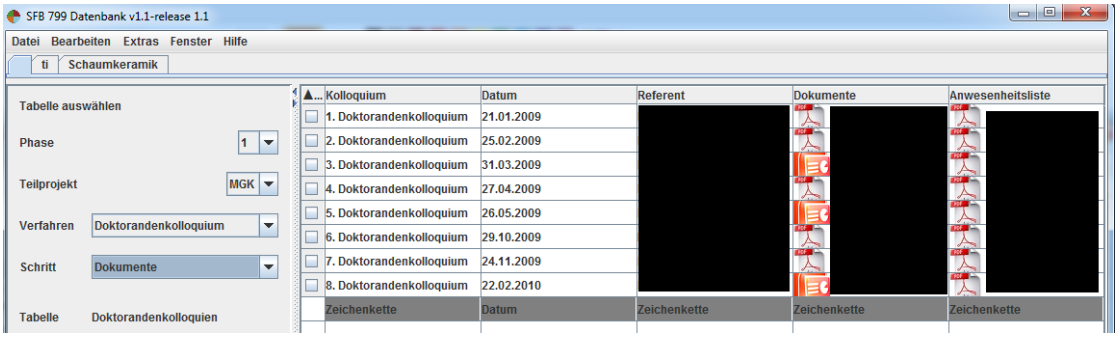


Figure 49: Screenshot of old Java version of the SFB799 database client

7 Presenting a complex software application written in GWT

As mentioned in the introduction, a user-friendly client program which should save the research results of "Trip-Matrix-Composite" project of the TU Freiberg should be developed. The Java program, which satisfied the functional specification document in 2009, was finished in the middle of 2010. Figure 49 shows the old Java Swing version. Since the project structure with phase, partial project, process, and step was not flexible enough, the user decided to replace it with a tree based one. Satisfying this wish means changing the entire database structure. Because version 0.5 of the Java program did not support file uploads, there was no need for an extra server, and so all MySQL requests were saved into the JAR archive. This was a large security hole as it is relatively easy to change the SQL strings in a published JAR file. The entire database can be destroyed by changing a simple value Update query to a Drop schema one. This would result in a big disaster. These two above mentioned issues, together with the fact that the users had to update the program every month because it was still under development, were the reasons to rewrite the entire program as a web application. Because the program has to handle many user inputs, e.g. resizing rows, switching columns, and merging cells; one criterion was that most of the program runs in JavaScript on the client computer, and not on the server. The question was now, whether this program should be implemented in JavaScript using jQuery, or in Java using GWT. One point in favor of the GWT toolkit was the impressive showcases of GXT (as mentioned in section 3.1). Another fascinating feature is the powerful GWT debugger, also that writing a GWT web application is very similar to writing a Swing one. However one point can be really annoying: The setup of a MySQL connection with the internal Jetty server. Changing the Jetty server code to allow SQL pools as shown in forums did not work. Creating an extra dynamic web project, which runs on Tomcat for the server-side part, and letting the GWT client part run on Jetty did not work either, because of Google's same origin policy, which meant that all remote calls failed. It took a long time to find out how the GWT development mode should run under Tomcat, as shown in section 6.2. Firstly, the web application was developed with GWT 2.4 and GXT 2.5. The advantage of this combination is that the GWTDesigner in Eclipse can be used to build views. After the update to GWT 2.5 and GXT 3.0 the declarative layout can be used; but the GWTDesigner does not support the GXT widgets anymore. Due to the examples in the GWT and GXT showcases it is easy to start programming with these two frameworks. Getting more detailed information, like handling own drag and drop events for the asynchronous tree grid to update the MySQL database is much more complicated, as no examples exist and so the GXT API has to be debugged. The advantage of all GWT based libraries is that the GWT compiler can only translate Java code and no byte code; this means there is always the opportunity to read the source code to find out when what event types are fired. Besides the many advantages of the GXT 3 library, it has one disadvantage in that it

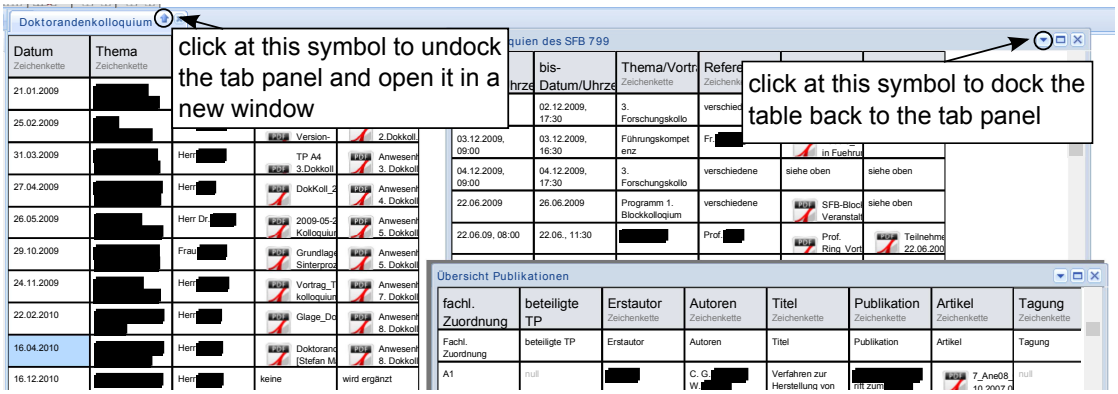


Figure 50: Screenshot showing the extended TabPanel allowing to undock TabItems

contains many private and package functions, which are not easy to extend an existing GXT widget like the TabPanel one. This means creating the com.sencha.gxt.theme.base.client.tabs package structure in the SFB project and having to copy the entire TabPanel widget, just to add some special behavior like undocking tab items from the tab panel and open them into a separate window as shown in figure 50. The usage of the GWT 2.5 and GXT 3 framework was the right decision for the database client project.

This passage makes some remarks about the GWT framework. Something really positive is that that the application can be debugged with Eclipse, and that the Eclipse plugin does a lot of code generation, e.g. the GWT Remote service. Less convenient is the fact that the compilation needs a long time when the project becomes large, even if the CPU has four processors. This is fine, as the web application can be tested in the development mode; but the web browser (tested with Firefox 16) needs a long time in reloading the website since the class validation needs very long for a large project. One of the drawbacks of GWT is the built in serialization mechanism, which sometimes causes unwanted behavior: (1) forgetting to add a Java type to the GWT's whitelist, causing the server not to send an answer; (2) having not a zero argument constructor resulting in complex GWT error messages due to the errors in the generated code. And even worse: the GWT deserialization mechanism contains bugs[45] when deserializing arrays or hash maps of the type java.io.Serializable.

The SFB web application showed that it was the right decision to write the project with the Google Web Toolkit, because the framework allows the creating of awesome web user interfaces which act really like a normal desktop app. Since the source code of the database web client (see figure A71) contains more than a million lines, this section shows only the two most interesting parts:

- How the project handled the copy-paste mechanism between the web application and Microsoft Office, and
- How to write an annotation processor using GWT deferred binding generator to show all desired UTF-8 block sings.

The next paragraphs explain how to enable normal copy and paste between the web application and other desktop ones. Firstly an HTML paste widget and a copy FocusPanel have to be created as shown in listing 150. The HTML attribute contenteditable="" is important because it allows the user to insert rich text with images and tables. At the beginning of the application, the visibility of both objects have to be set to false, and they have to be added to the RootPanel. The function setCopyHTML is shown in listing 151, and sets the HTML content to the copy

[45] I published the bug at [Mica].

widget, so that the user will copy it with control + C into its clipboard. The markText func-

```
1 private static final FocusPanel copy = new FocusPanel();
2 private static final HTML paste = new HTML("<div id=\"
    paste13112012\" contenteditable=\"\" style=\"width:100px;height
    :100px\"></div>");
```

Listing 150: Java code showing the creation of the HTML paste widget and the copy FocusPanel.

```
1  public static void setCopyHTML(String html) {
2    copy.setVisible(true);
3    HTML Html = new HTML();
4    if(Window.Navigator.getUserAgent().toLowerCase().contains("
       safari")) {
5      Html.setHTML("<div style=\"height:1px;width:1px\">"+ html + "
         </div>");
6    } else {
7      Html.setHTML("<div contenteditable=\"\" style=\"height:1px;
         width:1px\">"+ html + "</div>"); // must implement deferred
          binding, because in Chrome we cannot allow contenteditable
         , 'safari' should be selected (as it uses WebKit as Chrome)
         .
8    }
9    copy.setWidget(Html);
10   markText(Html.getElement());
11   copy.setFocus(true);
12 }
```

Listing 151: Java code of setCopyHTML method.

tion is copied from [z00]. The hideCopy function sets the visible attribute of the copy widget to false. The important part of the onKeyDown function in the TableView::TableController class is displayed in listing 152. In this function, the user's key down event will be intercepted

```
1  @Override
2  public void onKeyDown(KeyDownEvent event) {
3    controlKeyDown = (event.getNativeKeyCode() == KeyCodes.KEY_CTRL)
       || event.isControlKeyDown();
4    ...
5
6    if((controlKeyDown || event.isMetaKeyDown()) && event.
       getNativeKeyCode() == 67) {// CTRL + C or COMMAND + C
7      copyDataToClipboard();
8    } else if((controlKeyDown || event.isMetaKeyDown()) && event.
       getNativeKeyCode() == 86) { // CTRL + V or COMMAND + V
9      pasteDataFromClipboard();
10   }
11 }
```

Listing 152: Java code of onKeyDown function in the TableView::TableController class.

because it is called before the browser's standard action has been invoked. If the user did not reconfigure the browser standard actions, then it will copy the selected content after the user has

pressed control + C. The copyDataToClipboard function creates a Microsoft Office compatible table with many specific style commands like mso-number-format (see [cos03] and [Micb][46]). The most important code is shown in listing 153. The paste version is the inverse to the copy

```
protected void copyDataToClipboard() {
  Integer[] selCells = model.getSelectedCellRectangleInfo();
  if(selCells != null && excelRender != null) {
    String baseURL = "https://"+Location.getHost() + Location.
        getPath().substring(0, Location.getPath().lastIndexOf("/")
        +1);
    ...
    MainView.setCopyHTML(txt.replace("href=\"", "href=\"" +
        baseURL).replace("src=\"", "src=\""+ baseURL));
    Timer t = new Timer() {
      @Override
      public void run() {
        MainView.hideCopy();
        parent.setFocus(true);
      }
    };
    t.schedule(100);
  }
}
```

Listing 153: Java code of copyDataToClipboard function in the TableView::TableController class.

one. This means if the user presses control + V, then the paste HTML widget is shown, the innerHTML is set to an empty string, and the focus is activated to it. After 100 ms the user inserts the content into the contenteditable widget, meaning the content can be received by reading the innerHTML value and the paste object can be hidden.

This section ends by showing how to create own annotations and how to process them, just like the Google's @Source or @ImageOptions ones. First of all the annotation interfaces have to be defined as shown in listing 154. Annotation interfaces are defined with the @interface Java

```
public @interface Block {
  int start();
  int end();
}

public @interface UnicodeBlock {
  String name();
  Block[] uniCodeBlocks();
}

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface UnicodeView {
  UnicodeBlock[] unicodeBlocks();
}
```

Listing 154: Source code showing how to define Java annotations.

[46]have to download, extract the help file and open Style Attributes as shown in figure A72

keyword; and contain methods which have no parameter, do not throw any exception, and the return type is one of the following:

1. All primitive types like byte, short, int, ...; but no wrappers like Integer
2. String
3. Class
4. All enum types
5. Other annotations
6. One dimensional array of the types 1. - 5.

The annotation in listing 154 can be used to generate a widget, which shows all the buttons of different Unicode blocks and handles their click events. Listing 155 displays the code how to create a view as shown in figure A73.

```
1 @UnicodeView(unicodeBlocks = {
2   @UnicodeBlock(name = "Kyrillisch", uniCodeBlocks = { @Block(
        start=1024, end=1154), @Block(start=1160, end=1299) }),
3   @UnicodeBlock(name = "Griechisch␣und␣Koptisch", uniCodeBlocks =
        { @Block(start=890, end=1023) }),
4   ...
5     }
6 )
7 public interface UnicodeCharView extends Display { }
```

Listing 155: Source code showing how to use the self created annotations.

The advantage of using the annotations is that Unicode blocks can be easily added or removed. Naturally the same behavior could be achieved using a class and passing the wanted Unicode blocks as constructor parameters. The difference between these two methods is, that the HTML string containing all the buttons are created at

- Compile time when the annotation version is used or
- Runtime when the Java class version is used.

If there are 2000 UTF-8 signs, then the JavaScript code has to run a loop 2000 times to generate the HTML string. Since the buttons are created at runtime, the loop time is always saved, but the size of the *.cache.html files increases. As already explained in section 3.3.2, the following lines (see listing 156) have to be added to the module xml file.

```
1 <generate-with class="de.tu_freiberg.client.view.
      UnicodeCharViewGenerator">
2     <when-type-assignable class="de.tu_freiberg.client.view.
          UnicodeCharView" />
3 </generate-with>
```

Listing 156: XML code defining the generator class which should handle the annotations

The first plan was to create a large UiBinder file containing the declarative layout of the buttons, but I could not register the generated *.ui.xml file to the GWT resourceOracle (I asked this question at stack overflow [Micb], but I did not get an answer as to how to solve this problem until March 12th 2013). Because section 3.3.2 already explained how to generate code with the GWT compiler deferred binding, this paragraph only describes the createView function of the UnicodeCharViewGenerator (full source code is available in listing A146) in listing 157.

Line 5 iterates over all classes or interfaces in the module. The next two lines check if the

```
1  public void createView(String typeName, GeneratorContext context,
       SourceWriter sw) throws NotFoundException {
2    sw.println("public␣Widget␣createView()␣{"); sw.indent();
3    sw.println("com.google.gwt.user.client.ui.StackLayoutPanel␣st␣=␣
       new␣com.google.gwt.user.client.ui.StackLayoutPanel(com.google
       .gwt.dom.client.Style.Unit.PX);");
4    int i = 0;
5    for(JClassType type: context.getTypeOracle().getTypes()) {
6      UnicodeView unicodeView = type.getAnnotation(UnicodeView.class
         );
7      if(unicodeView != null) {
8        for(UnicodeBlock unicodeBlock: unicodeView.unicodeBlocks())
           {
9          StringBuilder sb = new StringBuilder();
10         for(Block block:unicodeBlock.uniCodeBlocks()) {
11           for(int unicode=block.start(); unicode <= block.end();
               unicode++) {
12             sb.append("<button␣class='gwt-Button'>&#").append(
                 unicode).append(";</button>");
13           }
14         }
15         sw.println("st.add(new␣com.google.gwt.user.client.ui.HTML
             (\"" + sb.toString() + "\"),␣\"" + unicodeBlock.name()
             + "\",␣20);");
16         sw.println("DOM.setStyleAttribute(st.getHeaderWidget("+ i
             + ").getElement(),␣\"fontSize\",␣\"0.9em\");");
17         sw.println("DOM.setStyleAttribute(st.getWidget("+ i + ").
             getElement(),␣\"overflowY\",␣\"scroll\");");
18         i++;
19       }
20       break;
21     }
22   }
23   sw.println("return␣st;");
24   sw.outdent(); sw.println("}");
25 }
```

Listing 157: Source code showing the createView function of the UnicodeCharViewGenerator.

interface or class is annotated with the @UnicodeView interface. Line 8 iterates over all @UnicodeBlock interfaces inside the UnicodeView annotation. The same thing is done in line 11. Line 12 creates the large HTML string using the StringBuilder class. The self-defined annotation interfaces can be used in the same way as normal Java ones. The usage of the Generator class in the deferred binding class also has some disadvantages:

- It is quite uncomfortable to write code: The code sw.println() has to be written every time, which reminds one of the HTML code generation in the Servlet era. It would be more convenient if the direct Java output code could be written into the file without the usage of the println command as the HTML code can be directly written in the JSP file. Another advantage of writing the generated code directly and without the usage of print functions, is that Eclipse can detect Java errors and the auto complete functionality would

work. Until now the output code is a string, which will not be checked and so syntax errors will only be noticed at compile time.

- The error messages when the programmer uses the annotations the wrong way are mostly confusing for users which do not understand the deferred binding process. Strange compiler errors occur when the code generation is successful, but the generated code cannot be compiled later on. As an effect of this, the GWT compiler generates a snapshot file, which contains the generated code. However this file can reference to other generated files, which are not available anymore. If this happens, then the -gen compiler option should be added to store all generated files in a separate folder. Copying some generated sub folders into the Eclipse source folder merges the Java files; this has the advantage that the Eclipse features are available, e.g. jumping to other classes by pressing the control key and click at any class or interface type.

8 Conclusions

8.1 Summary

In general it was possible to convert the Java program into a web application with the same behavior. The most important success factor was the widget offer of the different GWT based libraries like GXT. An equivalent widget exists for every Java Swing user interface component. Even lacking a certain widget behavior is not a problem as due to the available source code for all GWT based libraries, it is not difficult to extend any widget.

GWT's powerful remote procedure mechanism was another reason for the success of the database web project because it works similarly to the Java RMI technology. This means the developer does not have to take care of the data serialization and the remote method invocation at all.

The Agricola Model-View-Presenter example showed that it is possible to create large scale applications in GWT. Google's decision to choose the Model-View-Presenter pattern as default user interface design pattern makes it easy to create specific layouts for desktop, tablet, and mobile computers.

The JavaScript Native Interface allows importing different JavaScript libraries. This enables including the pdf.js library, which gives the user the opportunity to read PDF files without an extra Adobe plugin.

However, there are also limitations of the Google Web Toolkit. Deferred Binding uses the GWT as Java reflection replacement. This mechanism allows generating class specific properties, which uses e.g. the RPC mechanism to serialize the JavaScript objects to String values. Since the compiler creates the language or browser specific classes which are invoked by the bootstrap mechanism, there is no way to inspect Java variables and function names at runtime. This means GWT is not suitable for writing an integrated development environment like Eclipse, because these programs need to inspect private variable names in order to automatically generate getter and setter functions.

Other limitations of GWT are that only 2MB of model data can be stored in the URL or in the web storage and that the rich internet application has no direct access to clipboard data.

The answer to the question, whether it is possible to create a desktop-like web application in an efficient way with the Google Web Toolkit, can only be a partial yes. The reason is not that Google was not able to create a powerful framework. The answer to the problem cannot be completely positive, because the JavaScript engine does not have as much functionality as the Java Runtime Environment. Most of the missing JavaScript methods are not available due to security reasons. What I fail to understand is: Why does JavaScript not have access to the clipboard after pressing Ctrl+C or Ctrl+V, or why is the web storage or the URL length limited, despite Request for Comments (RFC) 2616 claiming "The HTTP protocol does not place any prior limit on the length of a URI" [R.].

Perhaps the W3C committee will solve these two drawbacks of web applications soon. Ten years ago, nobody imagined that the new HTML5 and ECMAScript 5.1 would support drawing 3-D figures in web browsers using WebGL, uploading files per drag and drop with the FileReader class, opening sockets to receive messages from servers, or executing different threads using Web Workers.

This section finishes by giving some benefits and drawbacks of web applications. Websites have several advantages for the user over conventional desktop programs, such as access from anywhere with the internet. Data entered in web pages does not get lost if the local computer crashes because the information is stored on the server. Rich Internet Applications (RIAs) are more suited for tablets and phones as no installation is needed and the hard disk size is very small on mobile devices. Since most browsers execute websites in a sandbox and do not allow

them access to the client's file system, internet pages cannot harm the user's computer and are preferred by many persons.

RIAs also have advantages for developers. Bug fixing is a lot easier in websites because there is no need to distribute any new version. Another benefit of web pages is not having to deal with several versions, since the web browser always uses the latest version. The developers are not forced to update their programs when a new Java runtime version appears; serialization errors caused that the client software running on JRE 1.7 could not communicate with the server running on JRE 1.6, and updating the server library to JRE 1.7 implies that every user also has to update their JRE 1.6 to the newer one. Integrating other web services like amazon login is much easier for web applications. The web page developer only has to force the web server to use Transport Layer Security (TLS) and does not have to deal with encryption in detail. Since software running on a server cannot be copied so easily, web applications are better protected from piracy than desktop programs.

One disadvantage of internet pages is the loss of speed, as JavaScript code is slower than native codes like Java, C++ or Adobe Flash. The user loses control over its data because it is not known what companies do with the uploaded files. Websites may become very slow at prime time, when many users cause the server to be overstretched. As mentioned above, desktop programs have better user experiences.

More information about advantages or disadvantages of web and desktop applications can be found at [And08].

8.2 Future work

The comparison of the execution speed showed that the Java program multiplying two 1000x1000 dimensional matrices is only slightly faster than the corresponding GWT application running in Chrome. The test also presented big differences in the execution speed of different web browsers. One future work could investigate the JavaScript code compared to the browser's execution time. An interesting topic would be how the browsers deal with the new JavaScript Web Workers; do they really open a new thread, or is the code only running scheduled on the same core.

Khronos announced WebCl [Khr] allowing JavaScript to run on the Graphics processing unit (GPU). This must also be compared to the Java GPU support using Aparapi [AMD] or the AMD & Oracle Sumatra project [Ora12] which might be available in Java 9. Intel also published a prototype of its RiverTrail [Int] project, allowing JavaScript to run on several cores and to use hardware vector instructions. My guess is that JavaScript will become more and more powerful, and in a few years nearly every application can run in the web browser. This means JavaScript is the new assembler, and the web browser is the new operating system. Nowadays, the JavaScript library pdf.js [Moz03b] can already replace the Adobe Reader, zip.js [Gil02] can zip and unzip files, Doppio [CJ 01] is an approach letting JavaScript interpret JVM byte code to execute Java programs in the web browser without needing an installed Java version on the local PC, and Shumway [Moz03a] is Mozilla's new JavaScript project emulating the Flash VM to allow tablet users to play existing flash games.

A Appendix

A 1 Configure the Google Web Toolkit framework in Eclipse

A 1.1 Install the Java Developer Kit

The latest Java JDK version can be downloaded at [Oraa].

After the License Agreement has been accepted, the Java version for computer can be downloaded. It is important to install jdk-7u21-windows-x64.exe file for any 64bit Windows version. The system type (32-bit or 64-bit Operating System) can be found at ControlPanel -> System and Security -> System.

If the Java installation has been successful, then the DOS command `java –showversion` prints a message as displayed in listing A1; otherwise an error message as demonstrated in listing A2 will be shown.

```
1 w:\>java -showversion
2 java version "1.7.0_05"
3 Java(TM) SE Runtime Environment (build 1.7.0_05-b06)
4 Java HotSpot(TM) 64-Bit Server VM (build 23.1-b03, mixed mode)
```

Listing A1: Java installation succeeded.

```
1 w:\>java -showversion
2 'java' is not recognized as an internal or external command,
3 operable program or batch file.
```

Listing A2: Java installation failed.

A 1.2 Download Eclipse

The Eclipse IDE for Java EE Developers is available at [FI]. It is important to download and unpack the Enterprise Edition (EE) as illustrated in figure A1, because the GWT plugin cannot be installed in the standard edition.

Figure A1: Download Eclipse IDE for Java EE Developers.

If the host computer has a lot of RAM, then it is useful to add the following options -Xms400m and -Xmx5120m into the eclipse.ini file.

A double click at eclipse.exe opens the IDE. The best way is to create a workspace folder in the eclipse one in order to have all the files and configurations at one central place. After the initial start, the workspace has to be chosen as displayed in figure A2.

A 1.3 Install the GWT plugin in Eclipse

Since the GWT compiler interprets the files as UTF-8 encoded, the Text File Encoding (Window -> Preferences -> General -> Workspace -> Text File Encoding -> Other: UTF-8) should be changed to UTF-8.

This paragraph describes how to install the GWT Eclipse plugin. Firstly, the menu Help -> Install New Software ... should be opened. Secondly, the Add ... button has to be clicked to

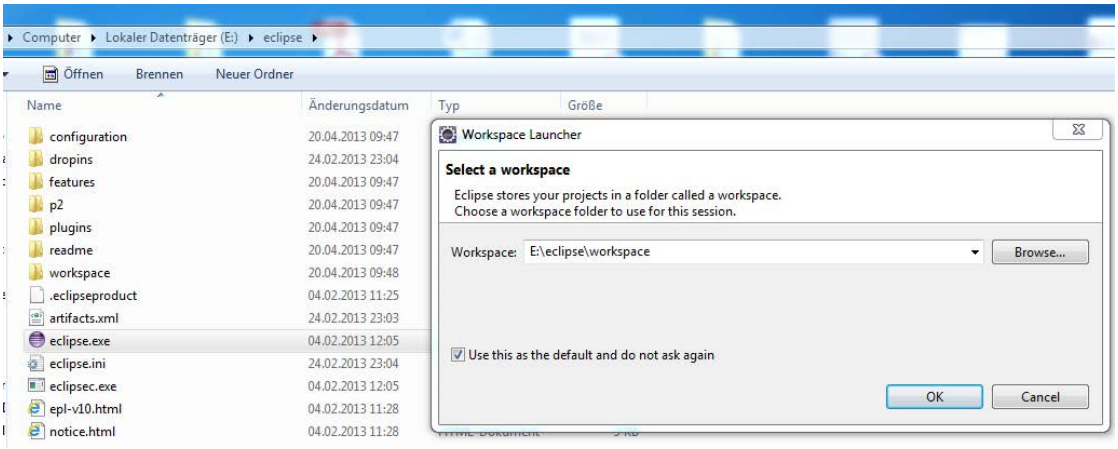


Figure A2: Screenshot of Eclipse file structure and dialog of Workspace Launcher.

insert the GWT repository as shown in table A1. Thirdly, the entries Google Plugin for Eclipse, GWT Designer for GPE, and SDK as displayed in figure A3 should be selected. All security warnings during the installation process have to be accepted with the OK button.

Table A1: Add GWT Plugin repository.

Name:	GWT Plugin
Location:	http://dl.google.com/eclipse/plugin/4.2

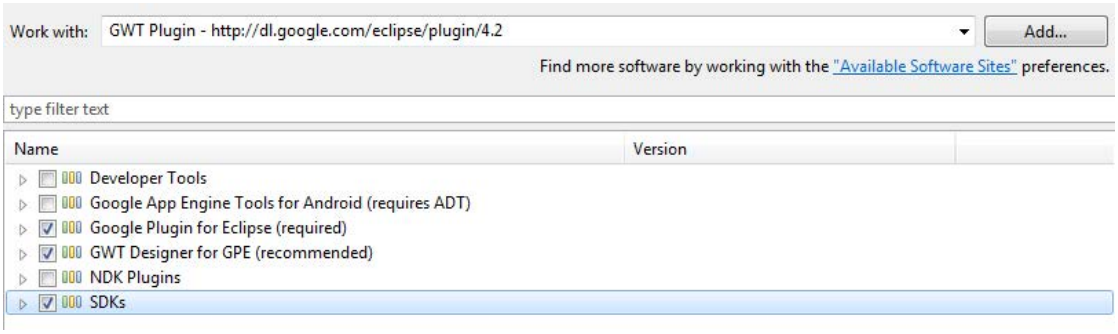


Figure A3: Selected entries of the GWT Eclipse plugin, which should be installed.

This passage explains how to install the GWT Designer Eclipse plugin. First of all the Install New Software dialog should be opened and the GWT Designer repository has to be added as shown in table A2. It is sufficient to install the GWT Designer and the WindowBuilder Engine as illustrated in figure A4.

A 1.4 Create first GWT Java Project

Firstly, a new GWT Java Project (File -> New -> Project ... -> WindowBuilder -> GWT Designer -> Model -> GWT Java Project) should be created. Secondly, "FirstProject" should be entered as name for this GWT project as shown in figure A5. Thirdly, check the option Create GWT module as displayed in figure A6. After the project has been created, figure A7 illustrates how the project can be executed as Web Application. Later on, an arbitrary browser opens the URL http://127.0.0.1:8888/ImageViewer.html?gwt.codesvr=127.0.0.1:9997.

Table A2: Add GWT Designer Plugin repository.

Name:	GWT Designer Plugin
Location:	http://dl.google.com/eclipse/inst/d2wbpro/latest/4.2

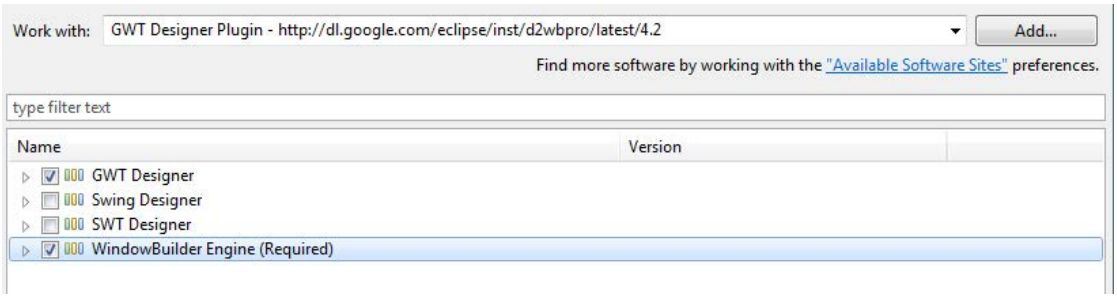


Figure A4: Selected entries of the GWT Designer Eclipse plugin, which should be installed.

The first time a GWT project is opened in a web browser, the Google Web Toolkit Developer Plugin has to be installed. Figure A8 shows the screenshot on how to download and install this browser plugin. After the plugin has been installed and the browser has been restarted, the GWT web application is available. A click on the button opens a pop-up window. A click at the red square icon (shown in figure A9) in the development mode stops the web application. The GWT Java project can also be launched in debug mode (right click at FirstProject -> Debug As -> Web Application). A breakpoint (displayed as blue circle) can be set by double clicking at the left frame in the source code window as illustrated in figure A10. The web browser should load the same URL as in the paragraph above. A click at the button intercepts the web application execution and opens the debug mode at the previous set breakpoint position.

Figure A11 displays how to compile the GWT project to JavaScript and HTML. The GWT compiler options -localWorkers 4 says that four threads are used in the compilation process. It is advisable to use as many threads as the computer has physical cores. The compilation result is available at the war folder as shown in figure A12. Opening the ImageViewer.html file executes this web application.

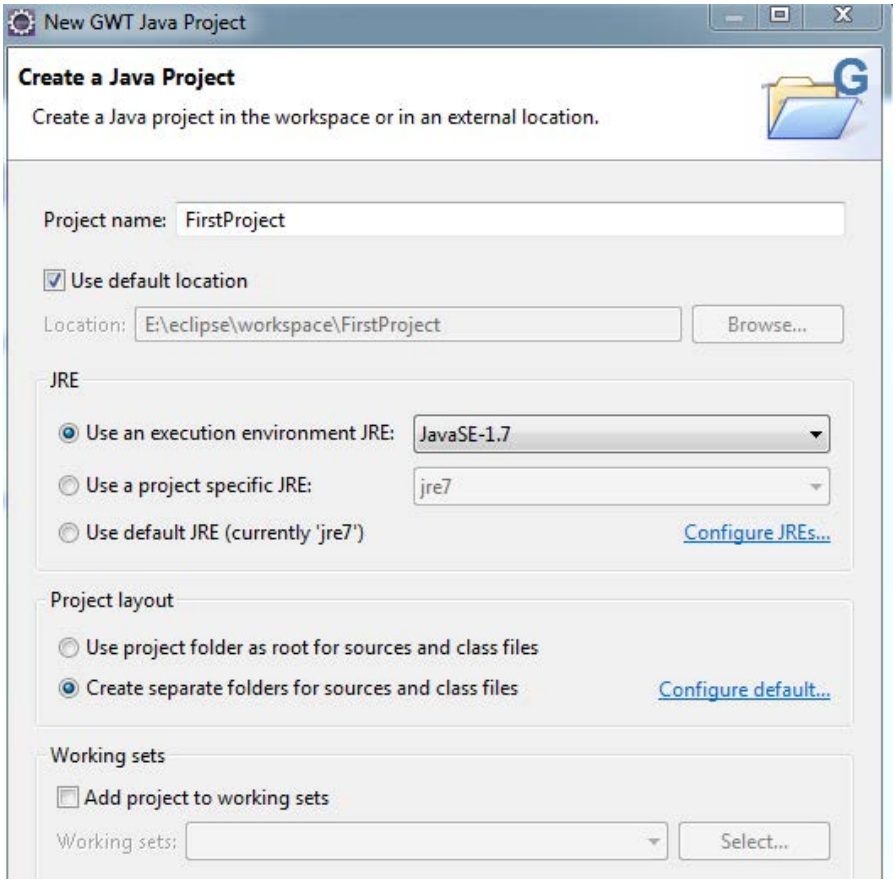


Figure A5: Properties of GWT project.

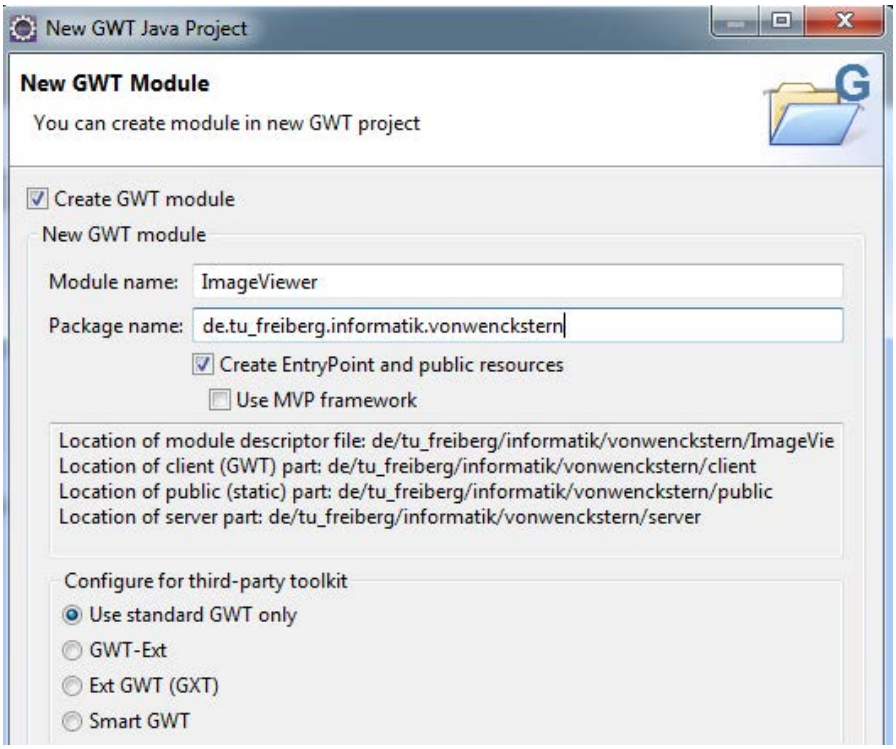


Figure A6: Create GWT module in FirstProject.

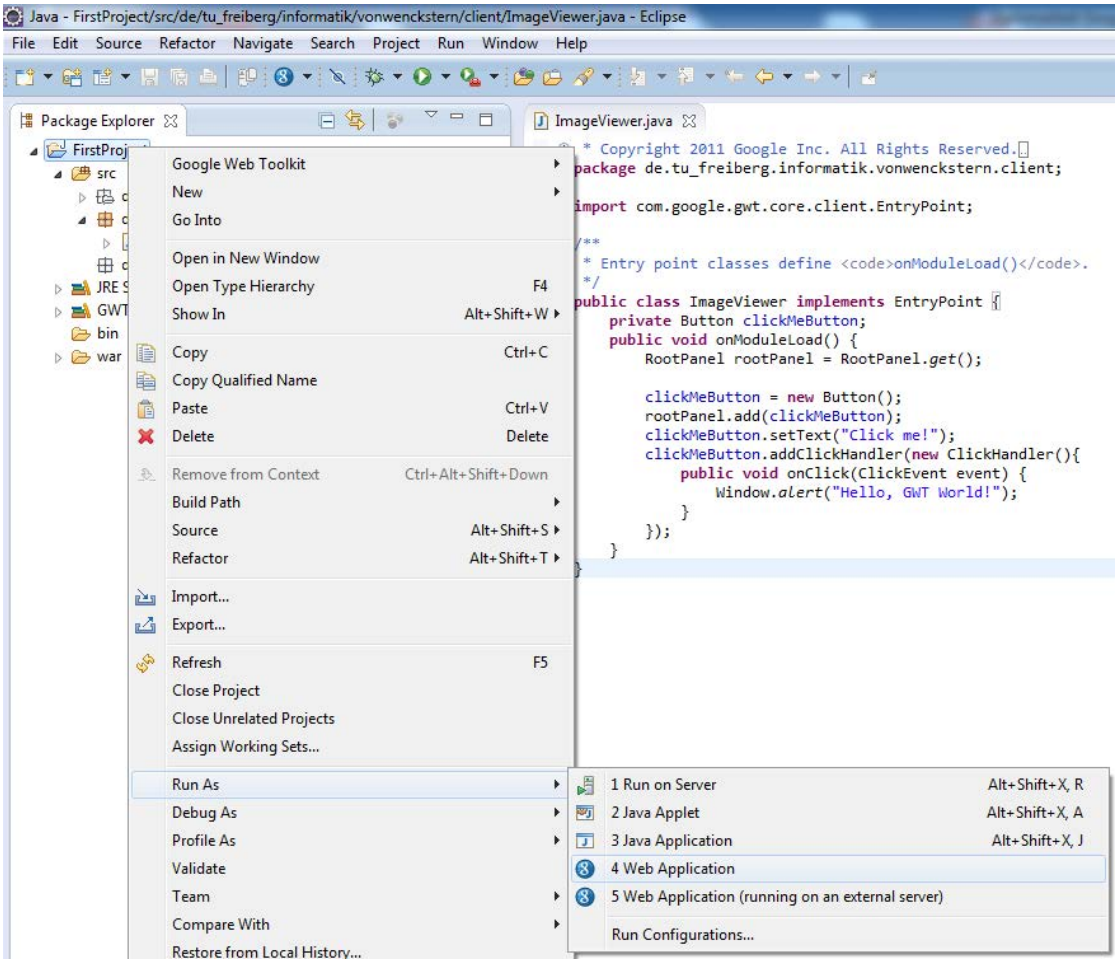


Figure A7: Right click at FirstProject opens popup menu. The menu entry Run As -> Web Application should be selected.

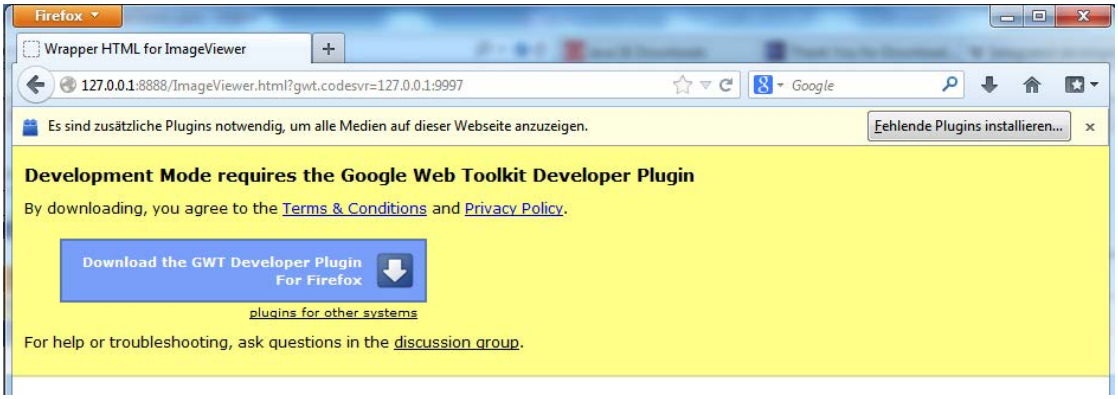


Figure A8: Screenshot of how to install the Google Web Toolkit Developer Plugin in Firefox 20.

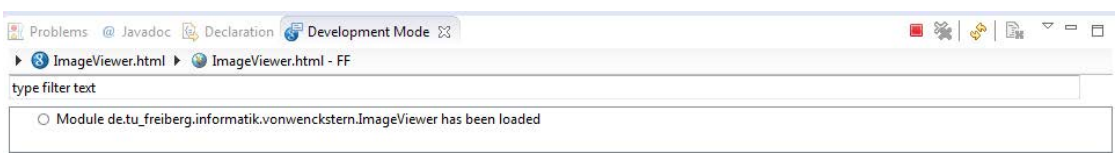


Figure A9: Left click at the right icon terminates the web application.

```
        rootPanel.add(clickMeButton);
        clickMeButton.setText("Click me!");
        clickMeButton.addClickHandler(new ClickHandler(){
            public void onClick(ClickEvent event) {
                Window.alert("Hello, GWT World!");
            }
        });
    }
}
```

Figure A10: Double click at the left frame in the source code window creates a breakpoint, which is displayed as blue circle.

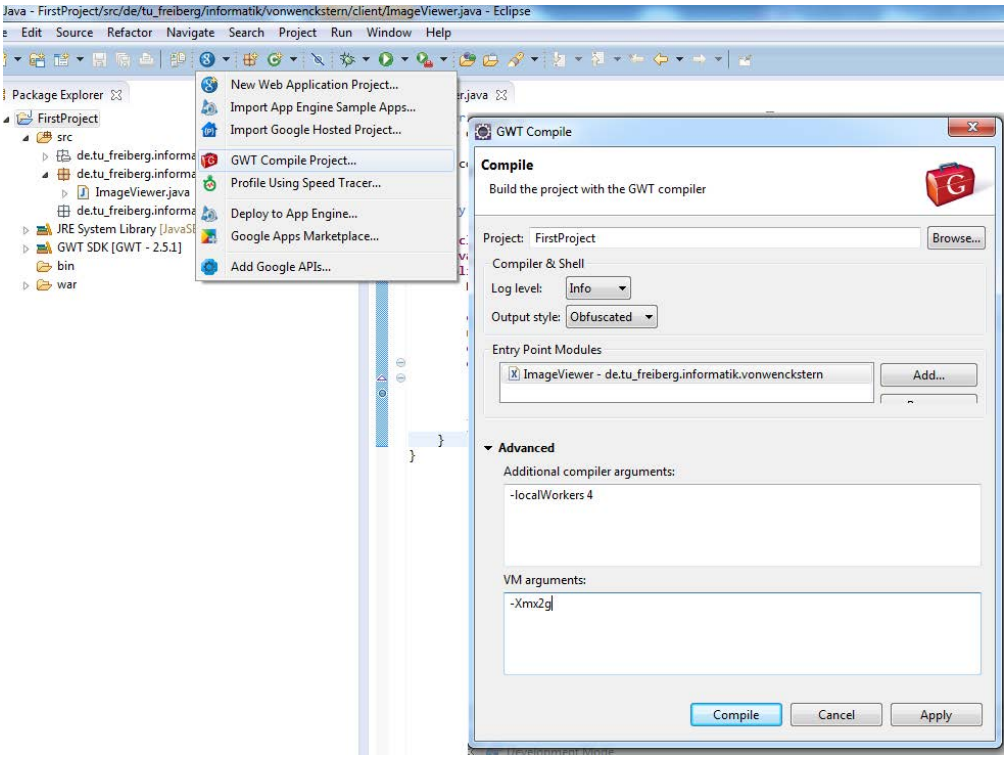


Figure A11: Screenshot showing how to compile the GWT project.

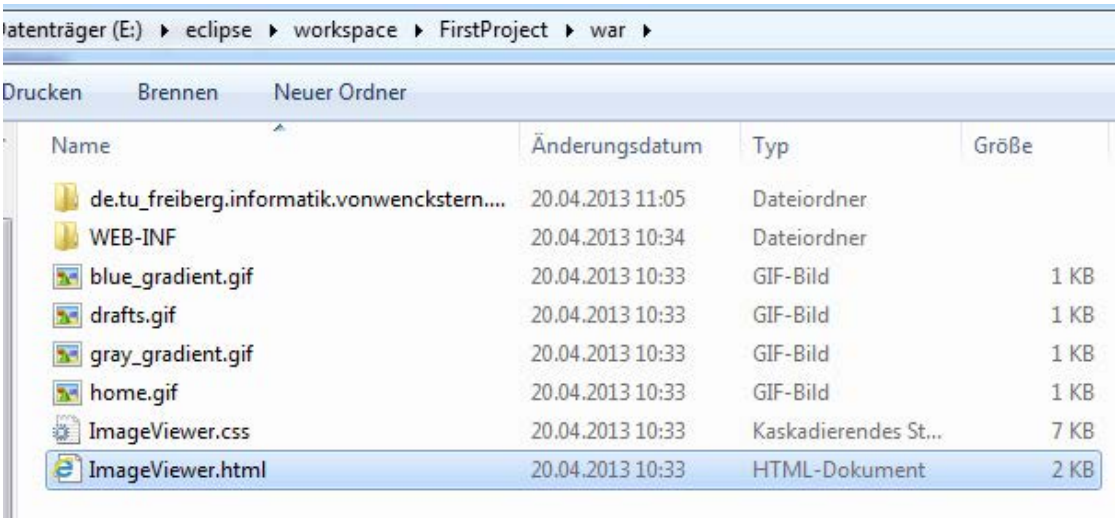
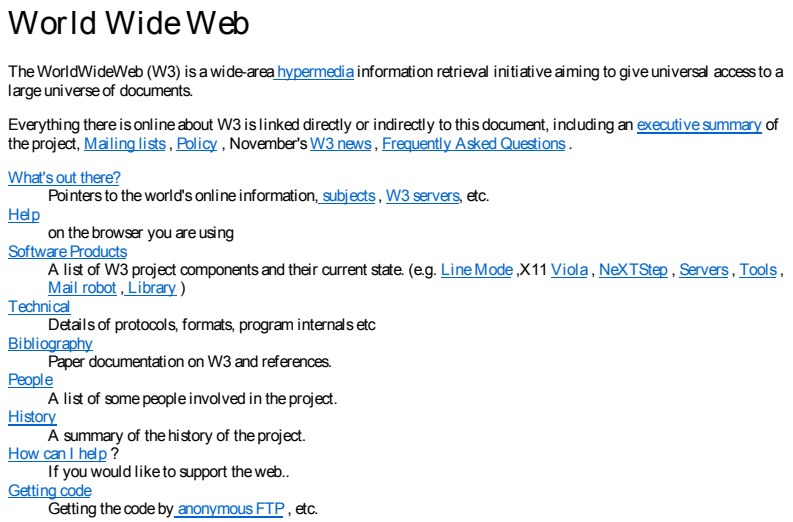


Figure A12: War folder contains the result of the compilation process.

A 2 Figures



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Figure A13: First website; source [Tim12]

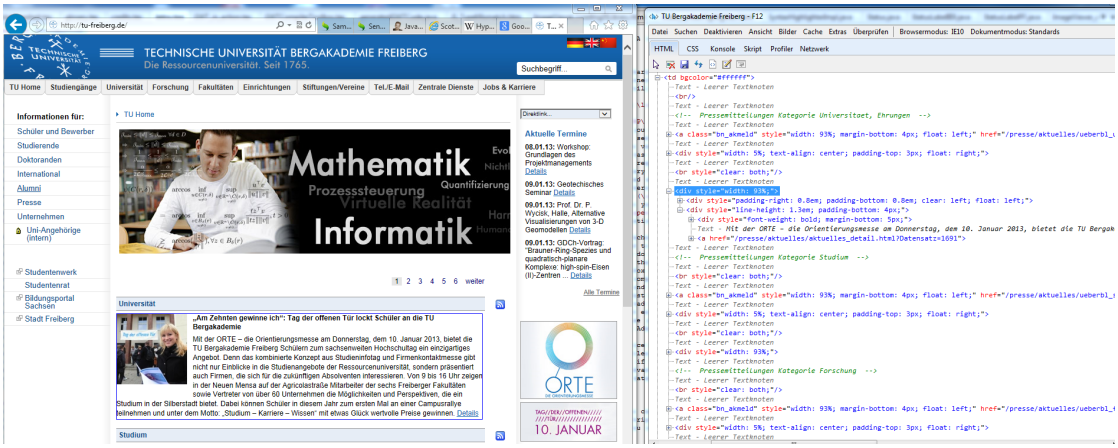


Figure A14: Screenshot of the homepage of the Technische Universität Freiberg (left) and the corresponding HTML code (right); source http://tu-freiberg.de/

ECMAScript Language – test262

ECMAScript **Language** test262 ECMAScript.org

Home | Run | Results | Development

Please click on the Run All button to run all the tests. Once you start the test you may pause the test anytime by clicking on the Pause button. You can click on the Results tab once the test is completed or after pausing the test. The Reset button is for restarting the test run. You may run individual tests by clicking the Run button next to the tests listed below. If you wish to run several chapters in sequence, but not the entire test suite, click the Select button for the chapters you wish to run and then click the Run Selected button.

Testing complete! Run All Run Selected Tests

Tests To run: **11573** | Total tests ran: **11573** | Pass: 11565 | Fail: 8 | Failed to load: 0

Chapter - ch06 (1 tests)	Select Loading test file: json/ch06.json	Run Loading test file: json/ch06.json
Chapter - ch07 (715 tests)	Select Loading test file: json/ch07.json	Run Loading test file: json/ch07.json

S15.5.4.7_A1_T11	Instance is Date(0) object	Fail
15.5.4.9_CE	Tests that String.prototype.localeCompare returns 0 when comparing Strings that are considered canonically equivalent by the Unicode standard.	Fail
S15.9.3.1_A5_T1	2 arguments, (year, month)	Fail
S15.9.3.1_A5_T2	3 arguments, (year, month, date)	Fail
S15.9.3.1_A5_T3	4 arguments, (year, month, date, hours)	Fail
S15.9.3.1_A5_T4	5 arguments, (year, month, date, hours, minutes)	Fail
S15.9.3.1_A5_T5	6 arguments, (year, month, date, hours, minutes, seconds)	Fail
S15.9.3.1_A5_T6	7 arguments, (year, month, date, hours, minutes, seconds, ms)	Fail

Test suite version: **ES5.1** | Test suite date: **2013-03-24**

© Ecma International

http://test262.ecmascript.org/#[12.04.2013 20:44:55]

Figure A15: Screenshot of ECMAScript Language test262 with Internet Explorer 10; test available at [Ecm03]

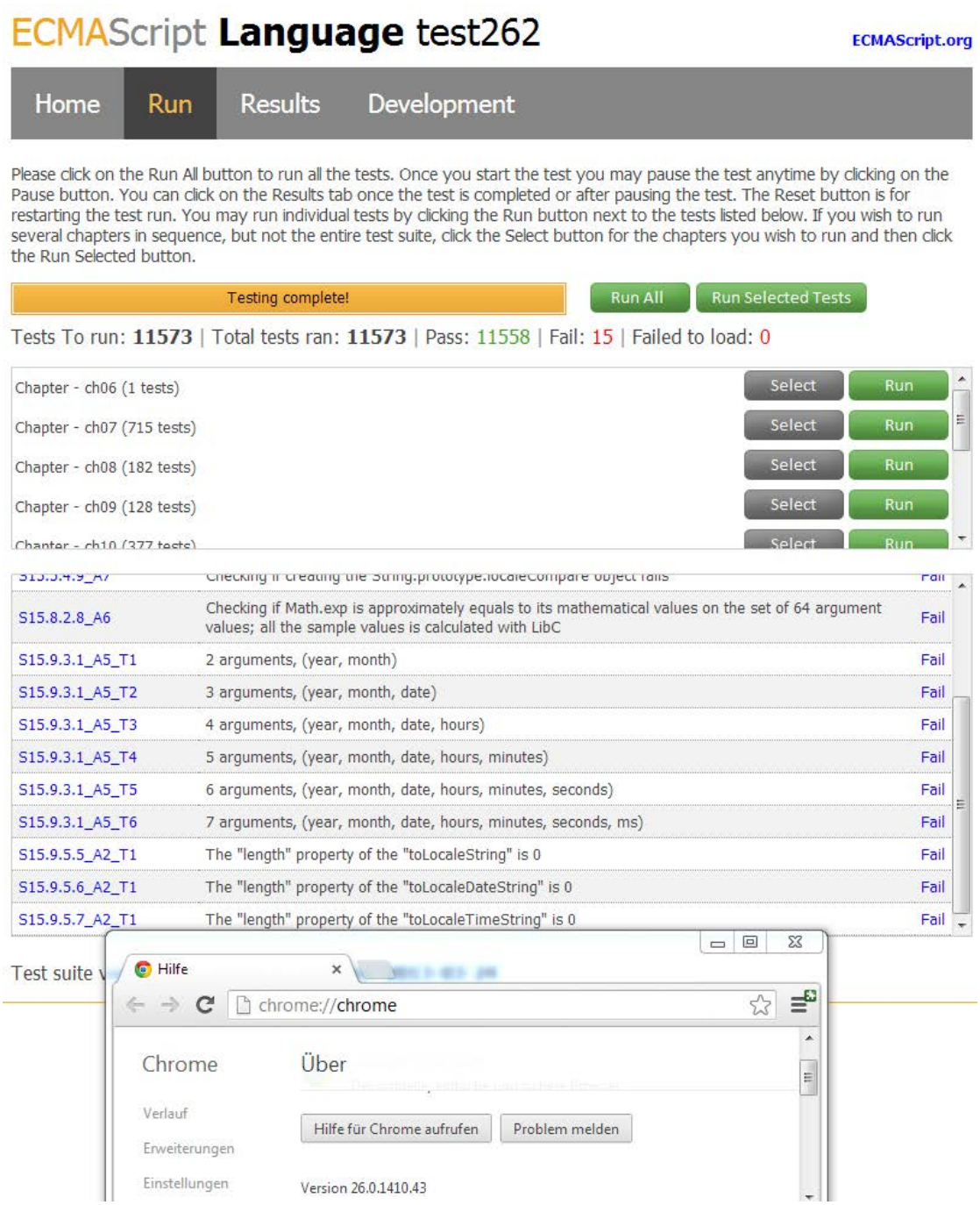


Figure A16: Screenshot of ECMAScript Language test262 with Chrome 26; test available at [Ecm03]

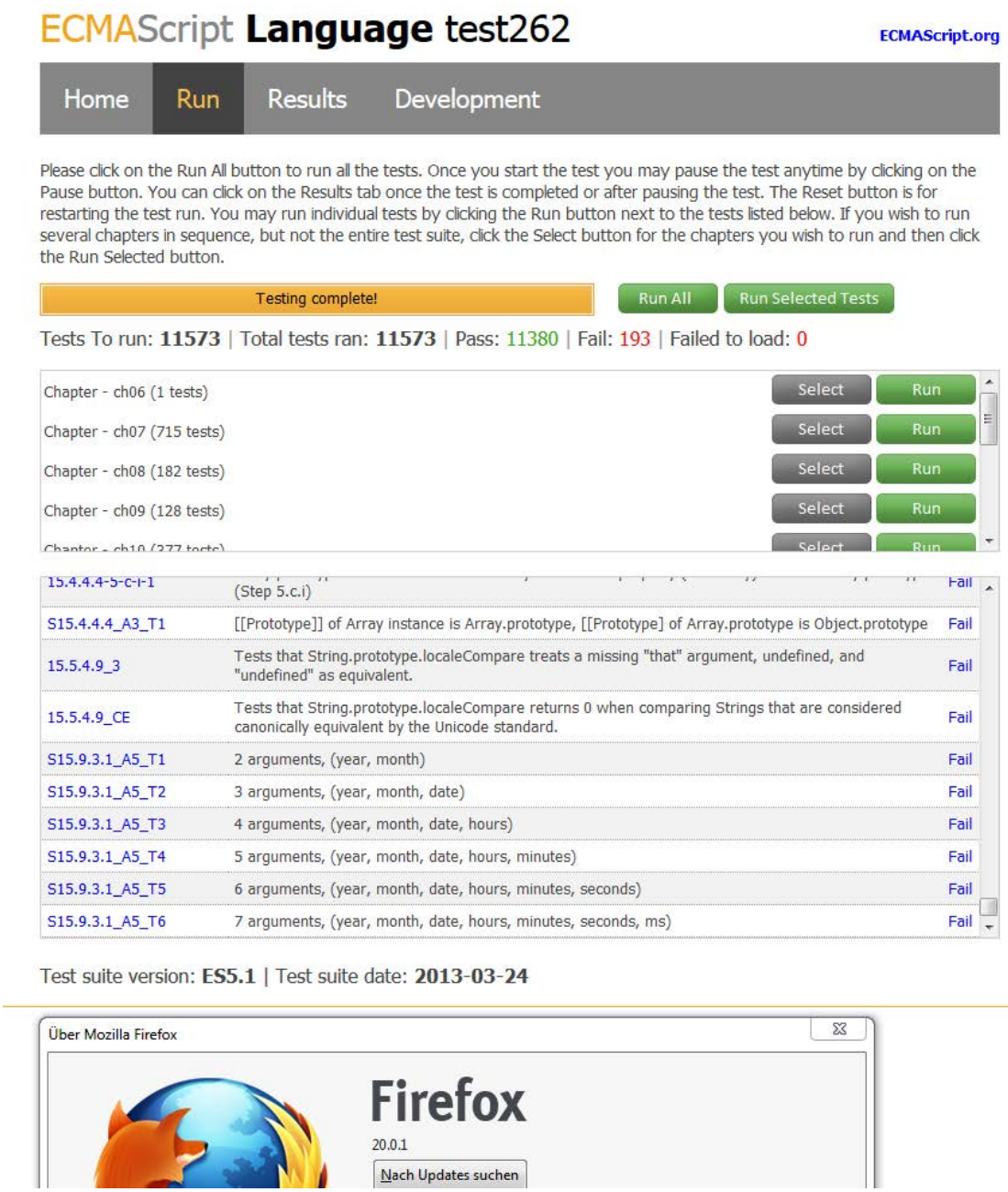


Figure A17: Screenshot of ECMAScript Language test262 with Firefox 20; test available at [Ecm03]

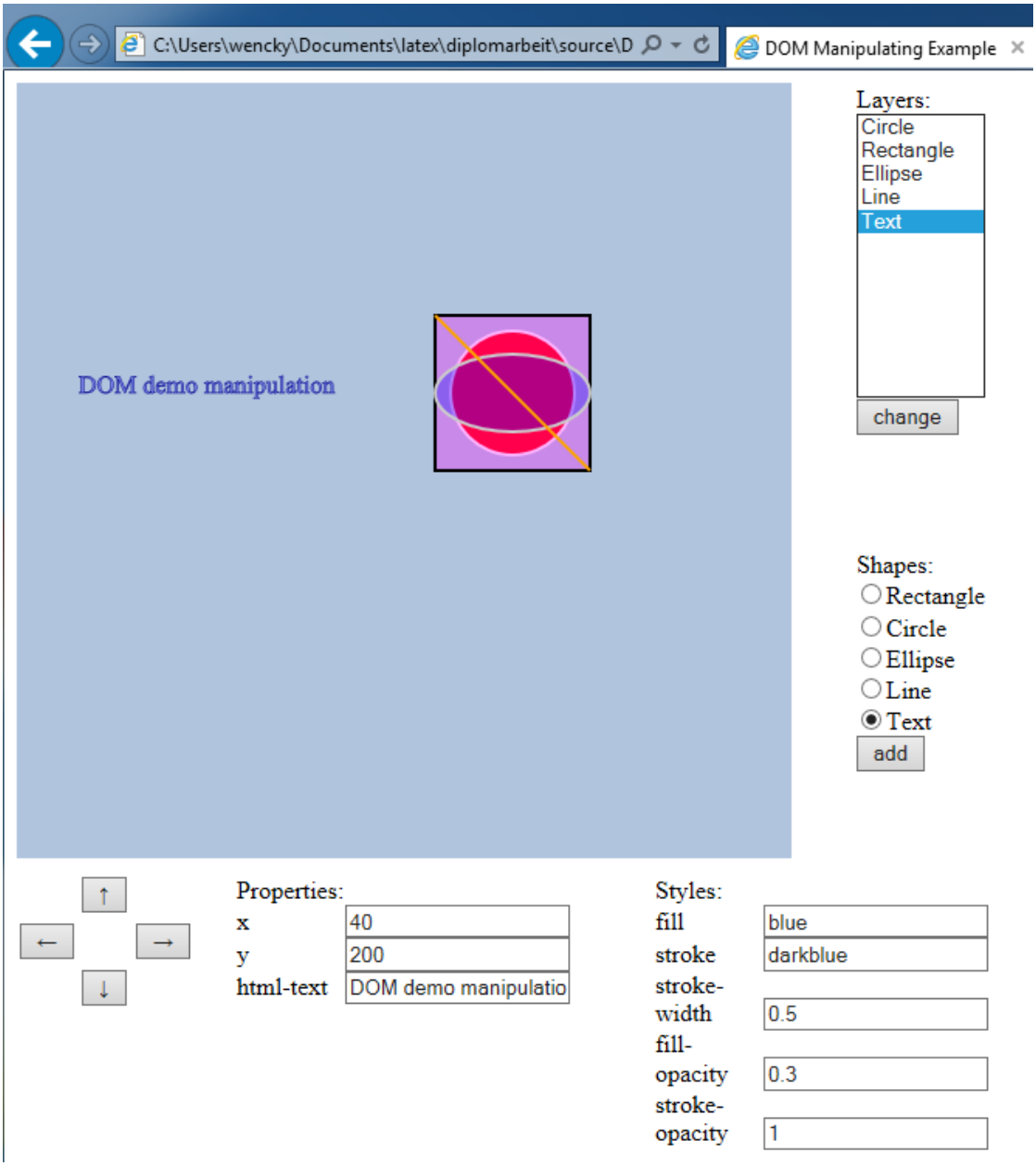


Figure A18: Screenshot of DOM manipulating demonstration.

Figure A19: Screenshot of AJAX example, part I.

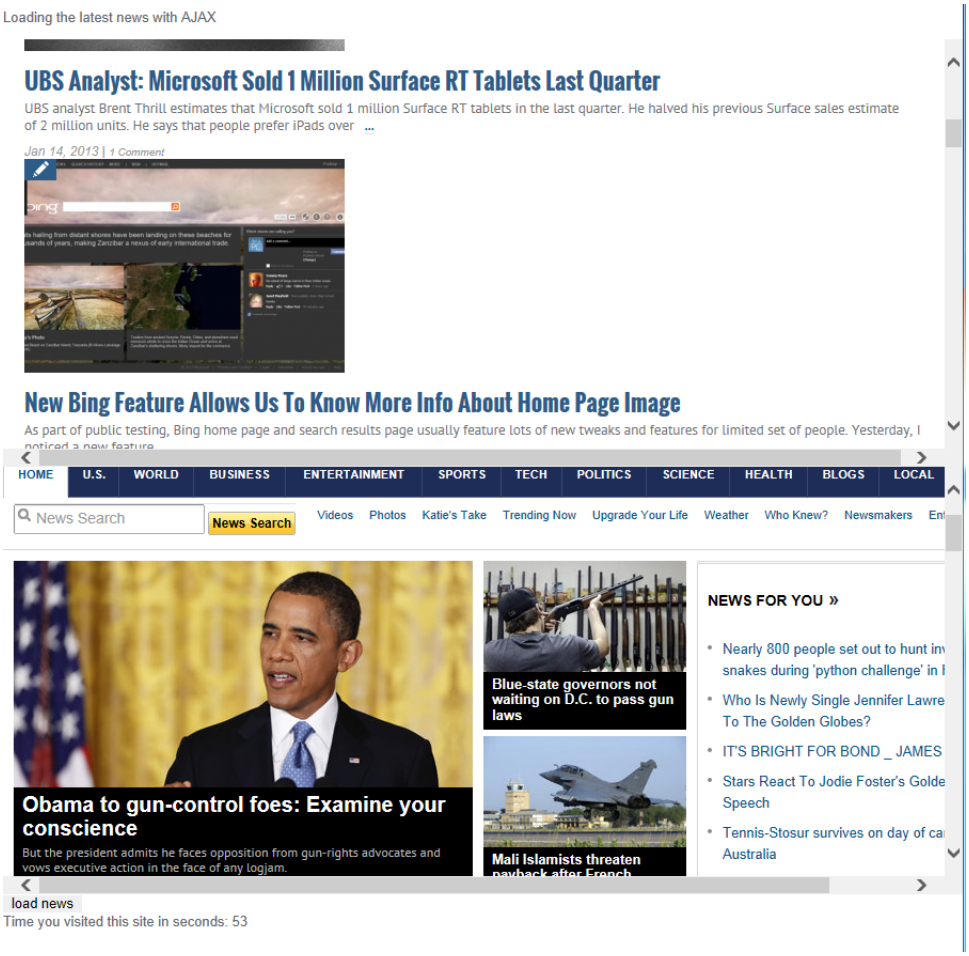


Figure A20: Screenshot of AJAX example, part II.

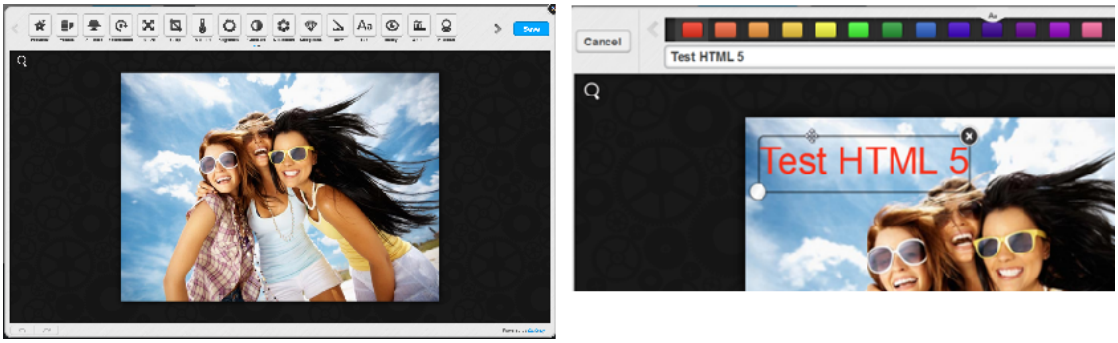


Figure A21: Screenshot of the web interface to paint pictures, the text element can be dragged around the canvas; source [Avib]

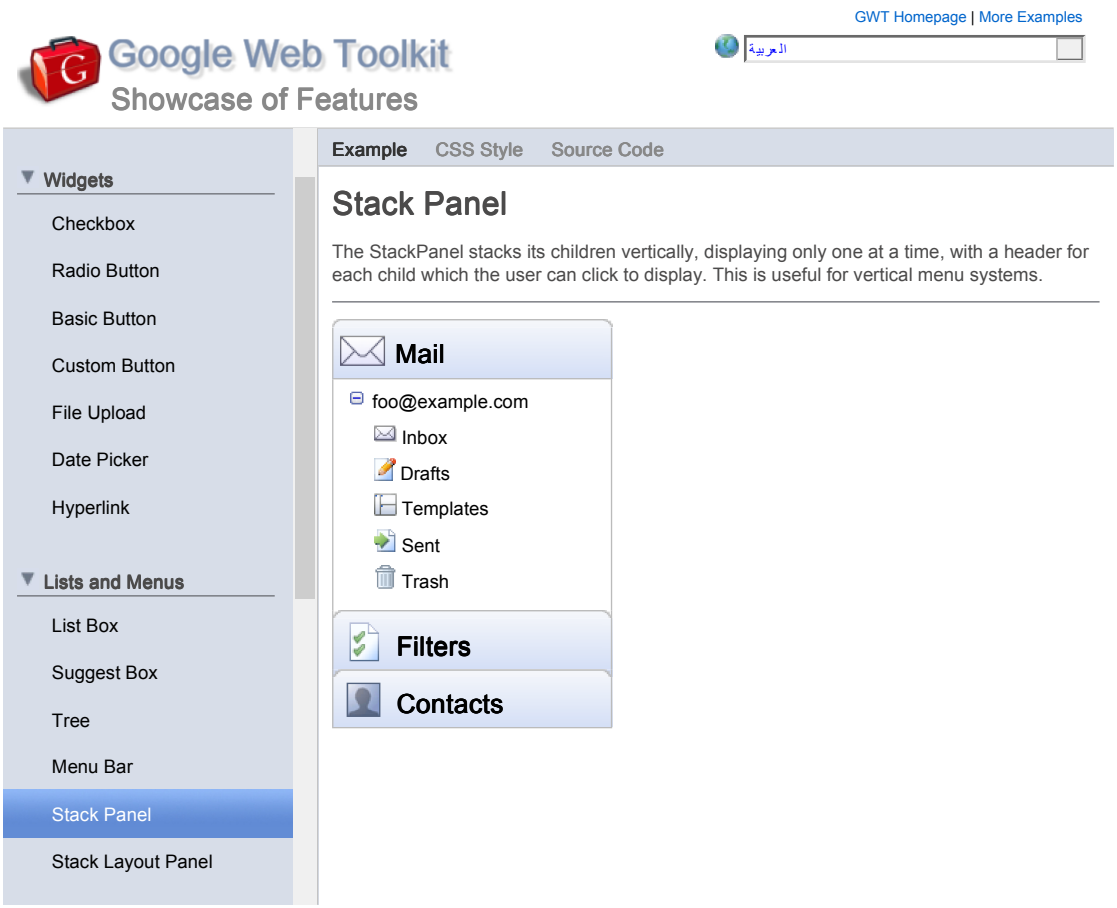


Figure A22: Screenshot of GWT showcase testing Stack Panel component; source [Goo11]

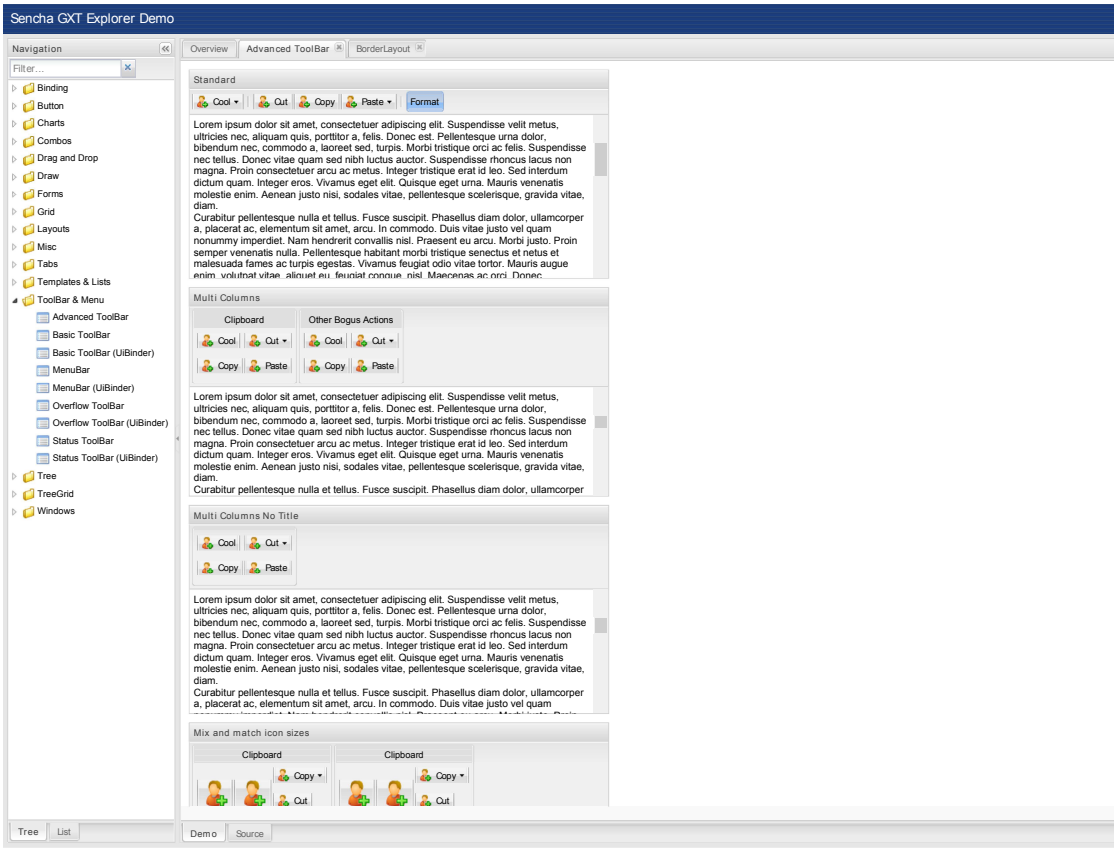


Figure A23: Screenshot of GXT showcase testing advanced toolbar component; source [Sen06b]

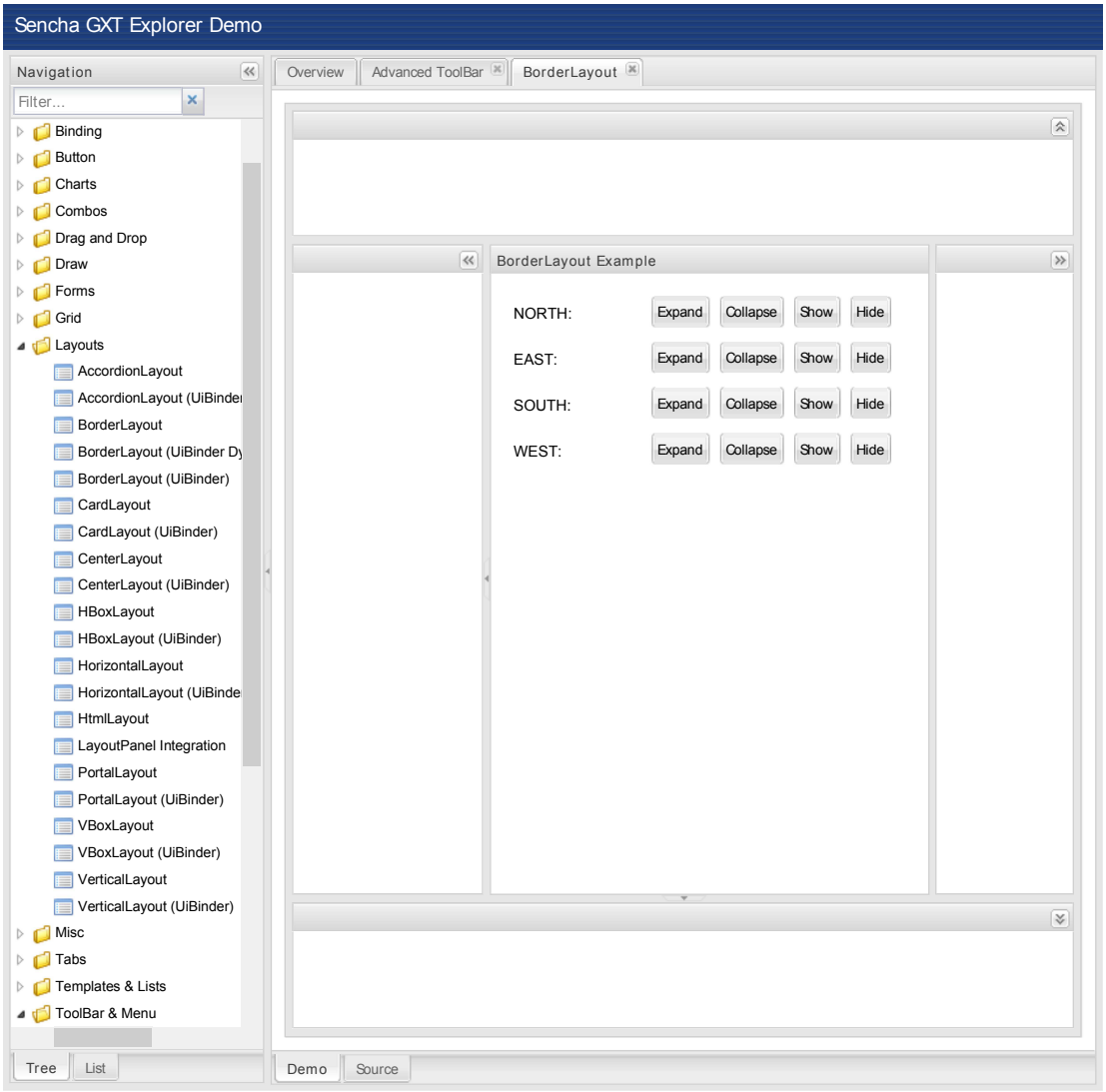


Figure A24: Screenshot of GXT showcase testing border layout; source [Sen06a]

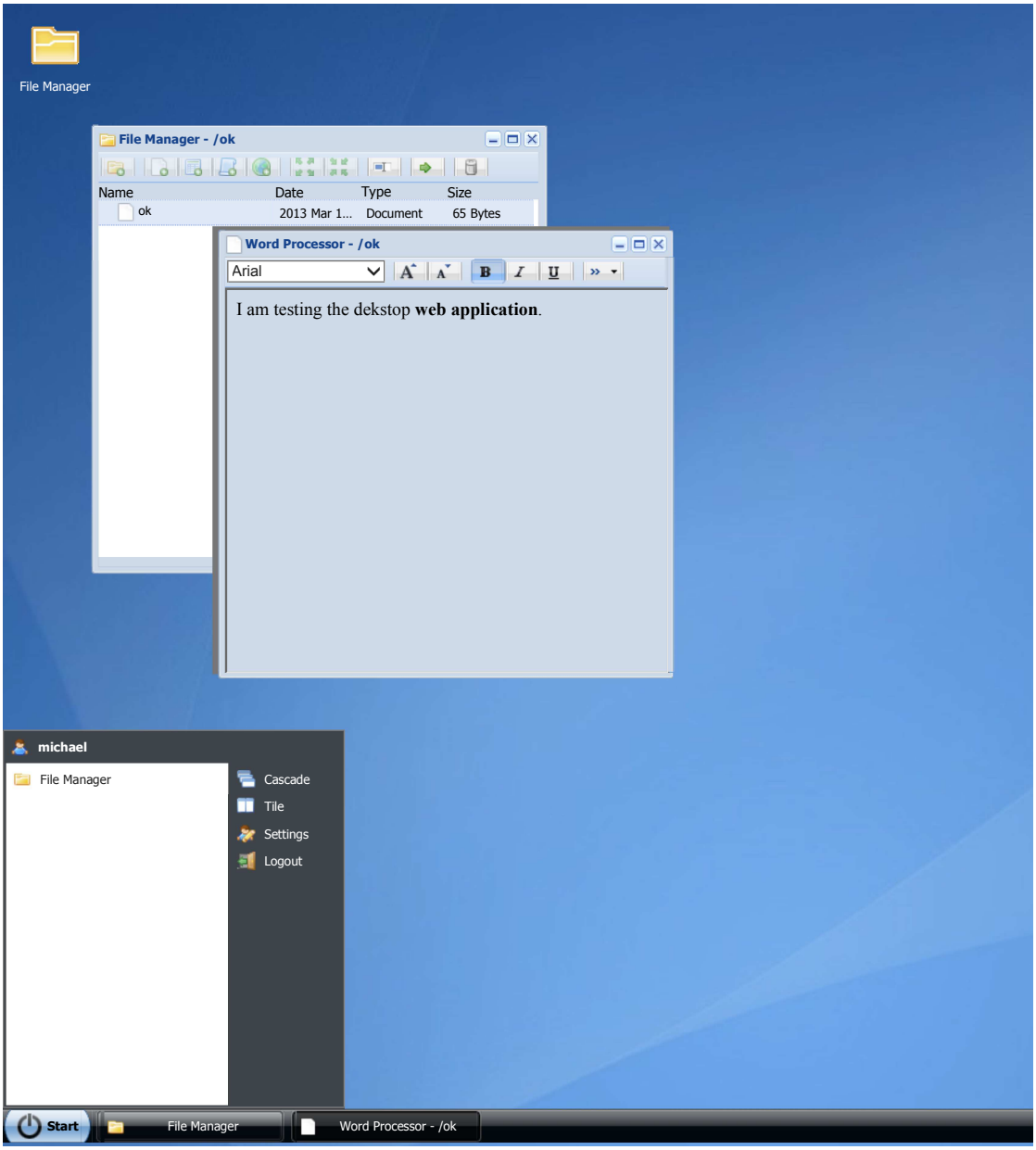


Figure A25: Screenshot of GXT desktop showcase; source [Sen08a]

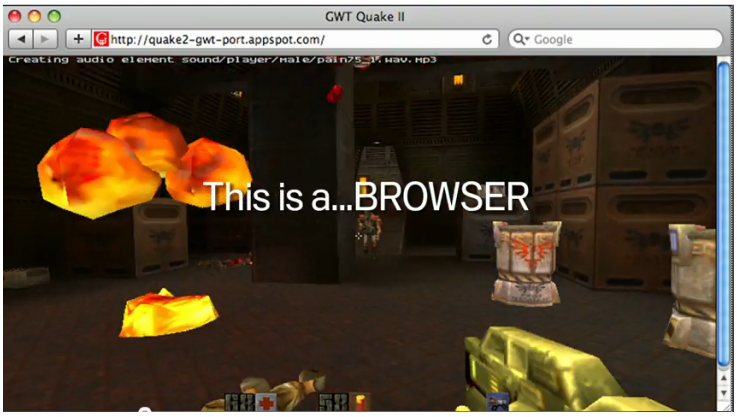


Figure A26: Screenshot of youtube video playing Quake 2 in the web browser; source [Ste]

Figure A27: Screenshot of youtube video testing mgwt showcase on an iPhone simulator; source [Dan06]

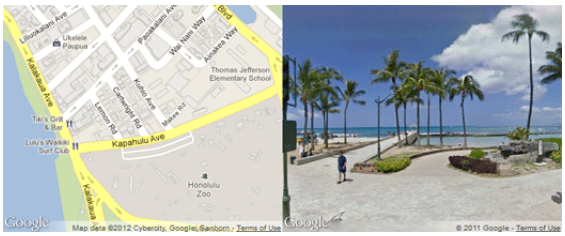


Figure A28: Picture showing interactions between GWT and Google Maps API V3; source [Kei09]

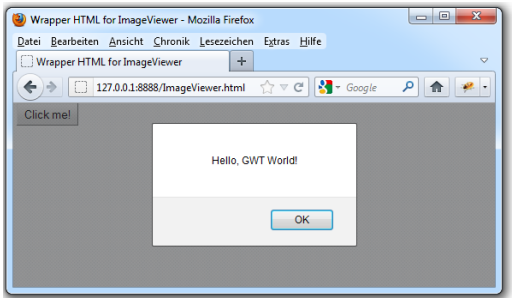


Figure A29: Screenshot of generated website by Eclipse

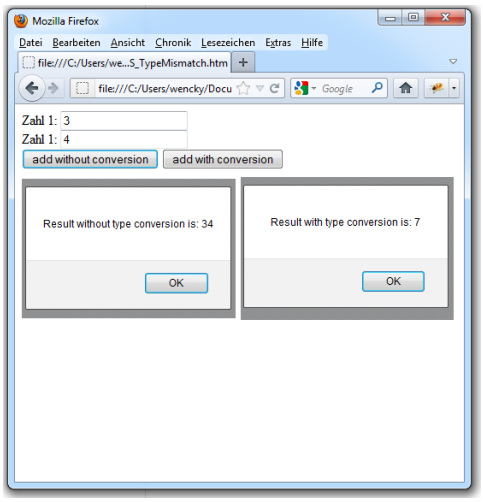


Figure A30: Screenshot of type conversion in JavaScript

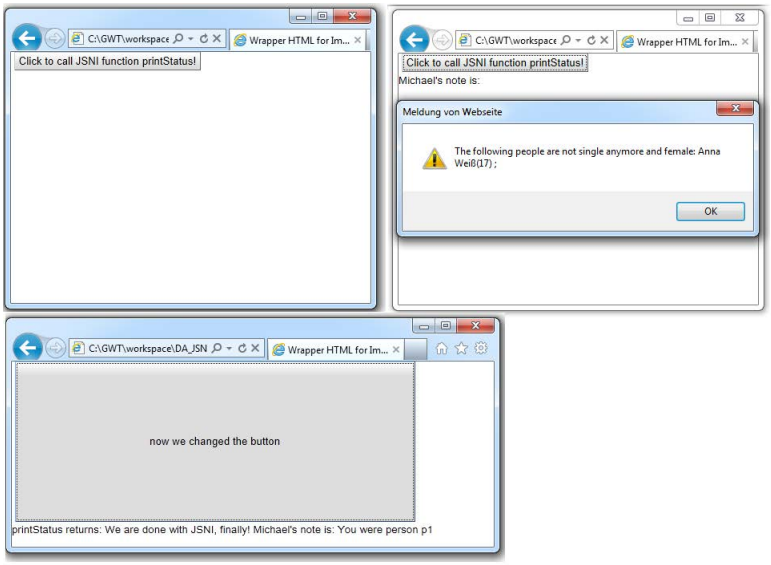


Figure A31: Screenshot of JSNI example.

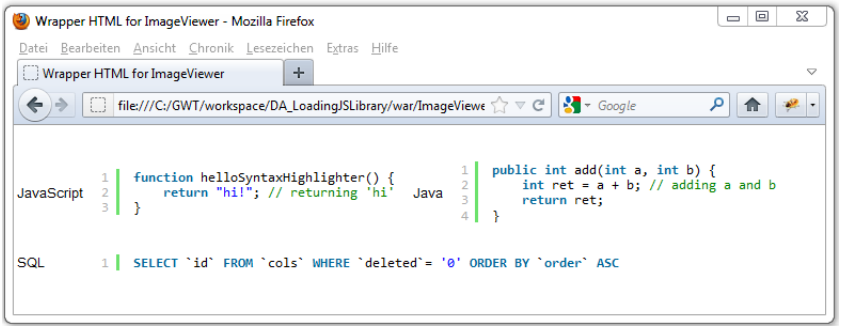


Figure A32: Screenshot testing wrapped syntaxhighlighter JavaScript library in GWT

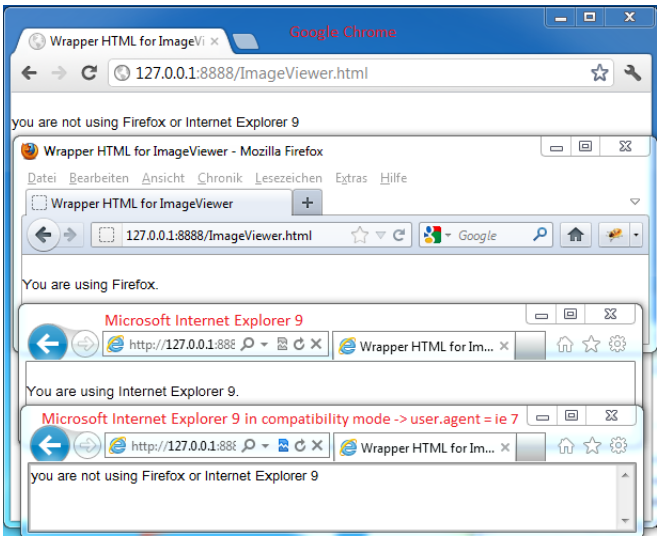


Figure A33: Screenshot testing deferred binding with replacement

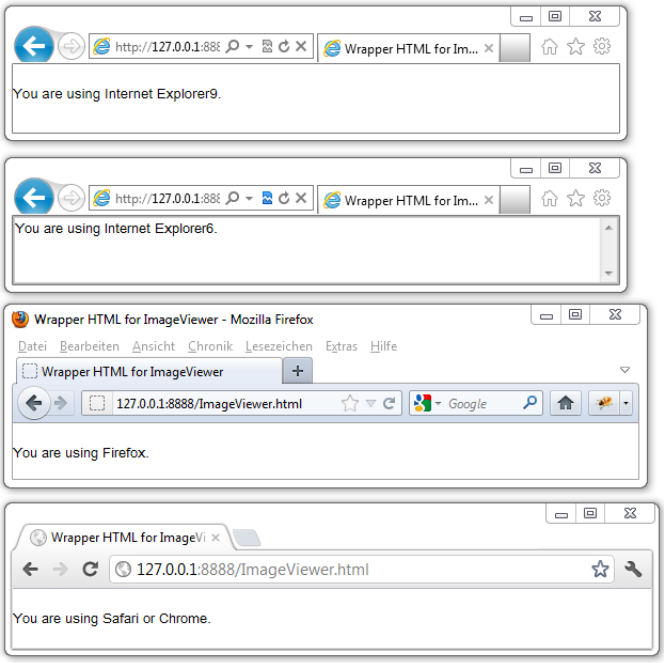


Figure A34: Screenshot testing deferred binding with generator

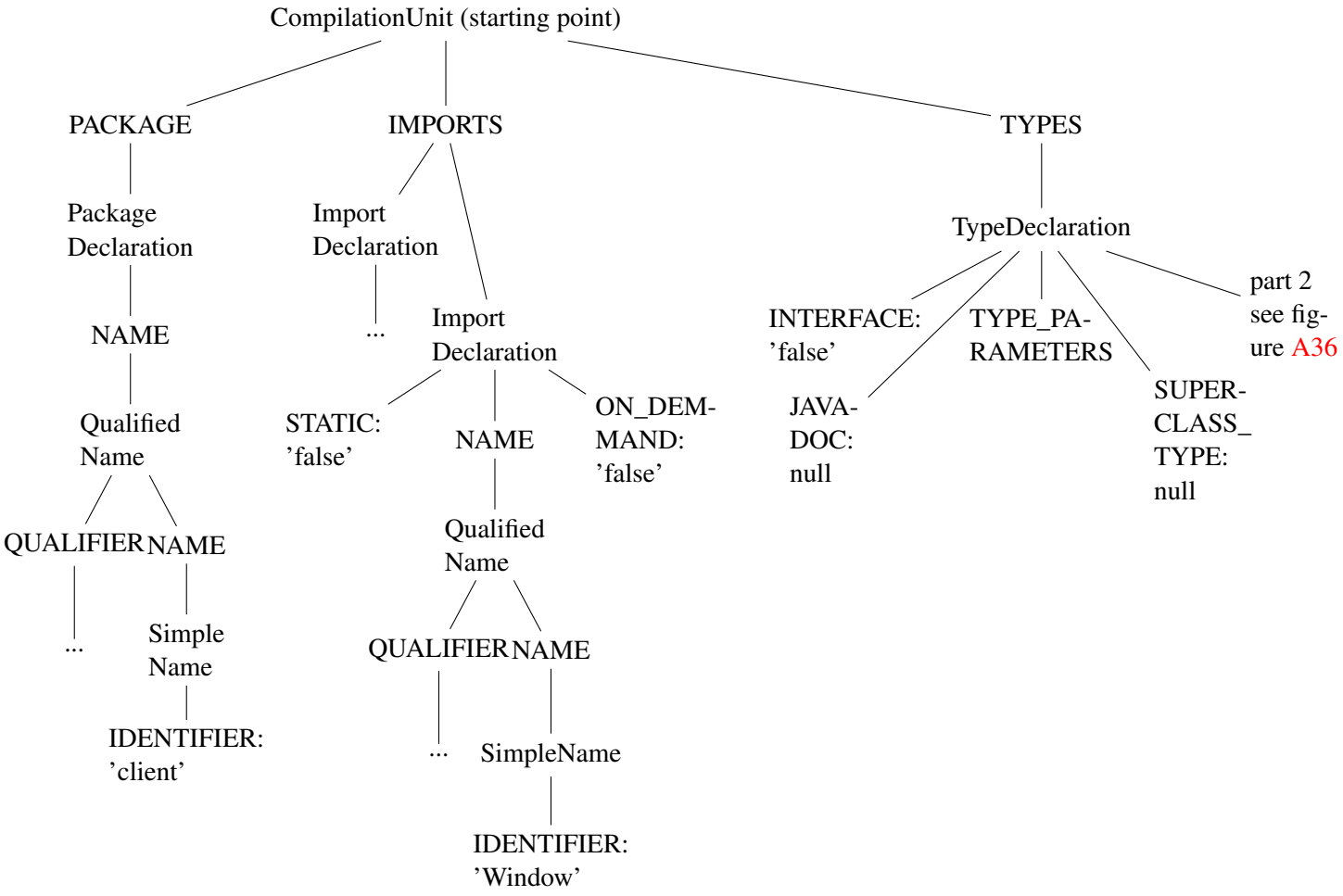


Figure A35: Eclipse Java AST of listing A20 (created with Eclipse plugin ASTView)
Table A3 explaines the ASTNode types.

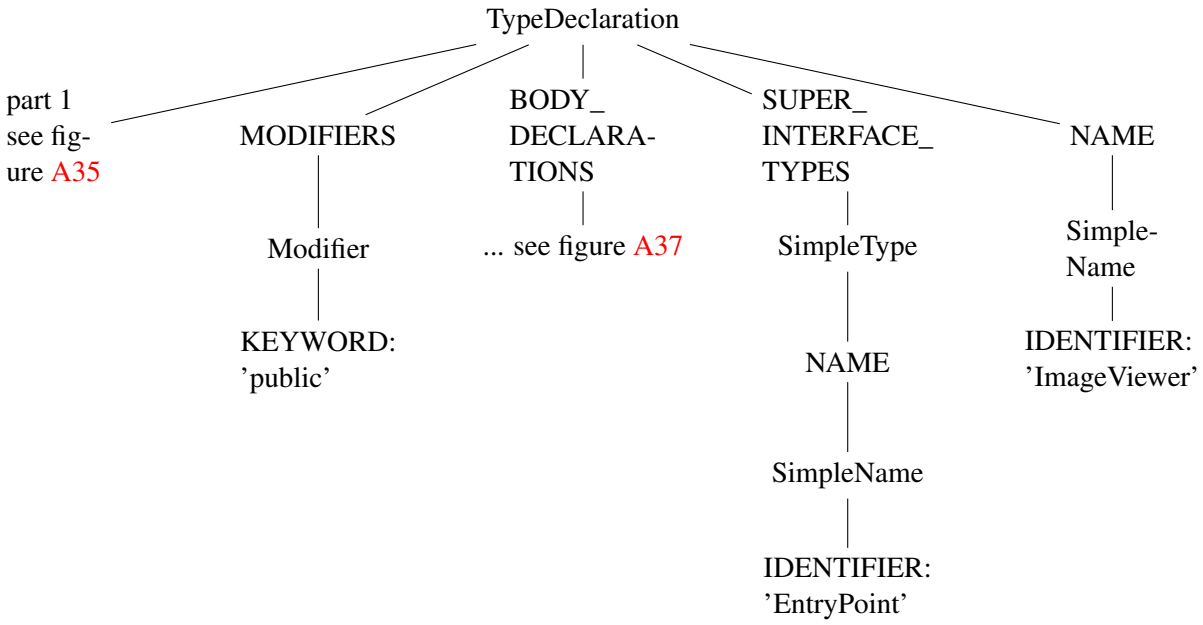


Figure A36: Eclipse Java AST of listing A20 (created with Eclipse plugin ASTView)[continued] Table A3 explaines the ASTNode types.

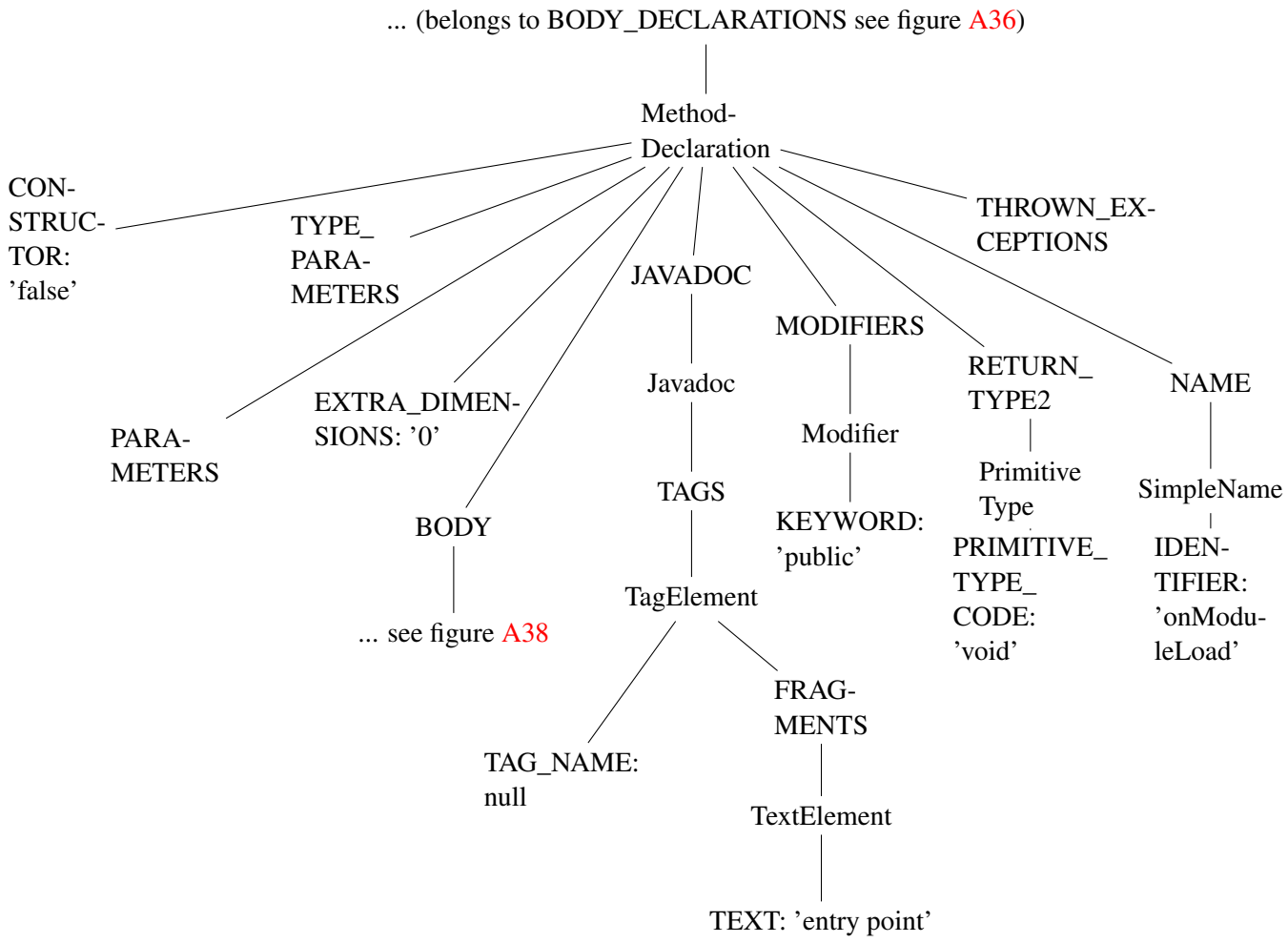


Figure A37: Eclipse Java AST of listing A20 (created with Eclipse plugin ASTView)[continued] Table A3 explaines the ASTNode types.

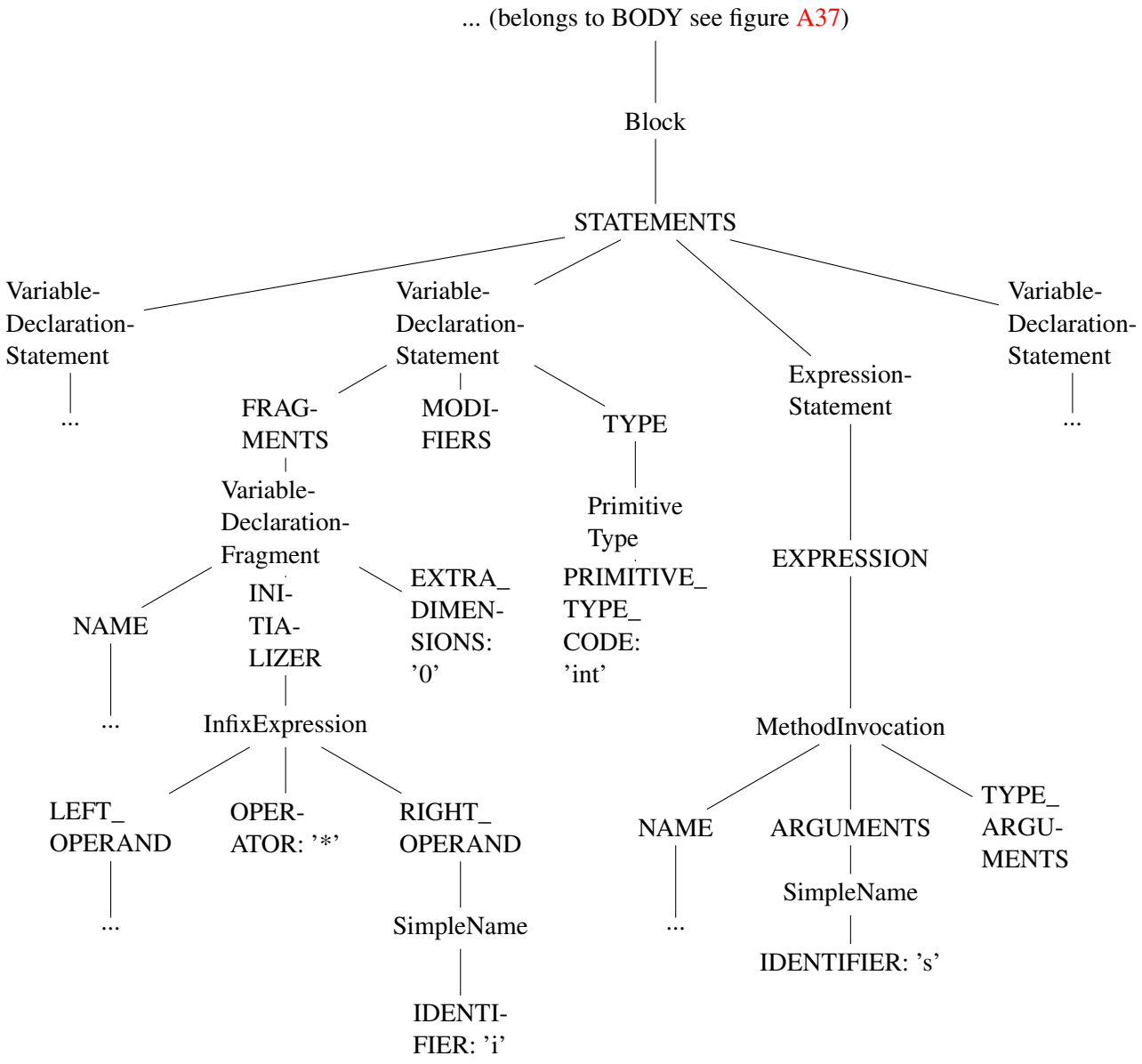


Figure A38: Eclipse Java AST of listing A20 (created with Eclipse plugin ASTView)[continued]
Table A3 explaines the ASTNode types.

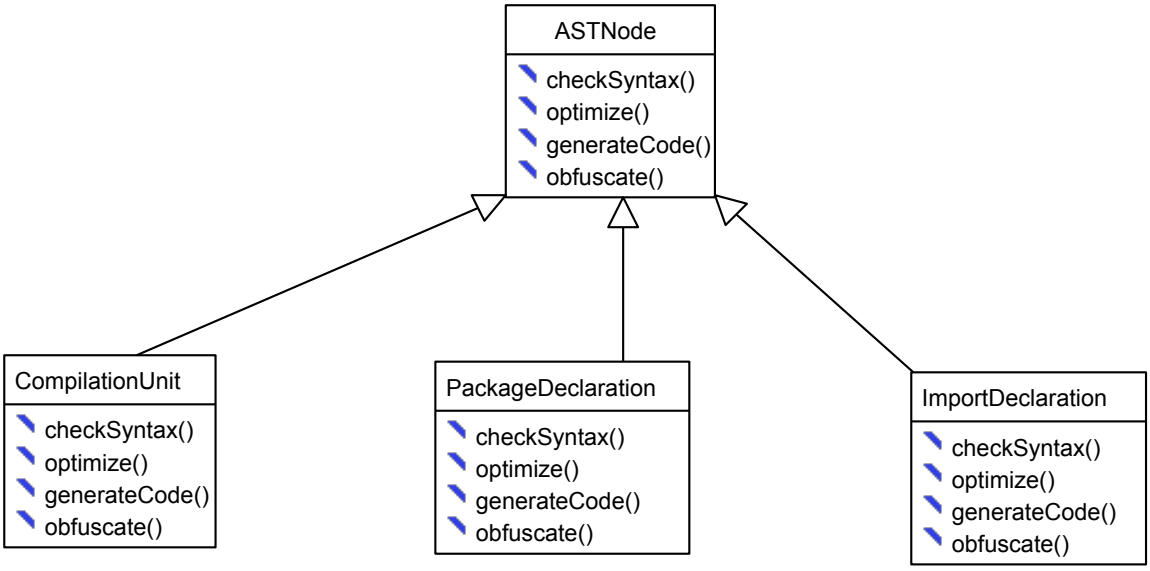


Figure A39: Inheritance pattern for AST

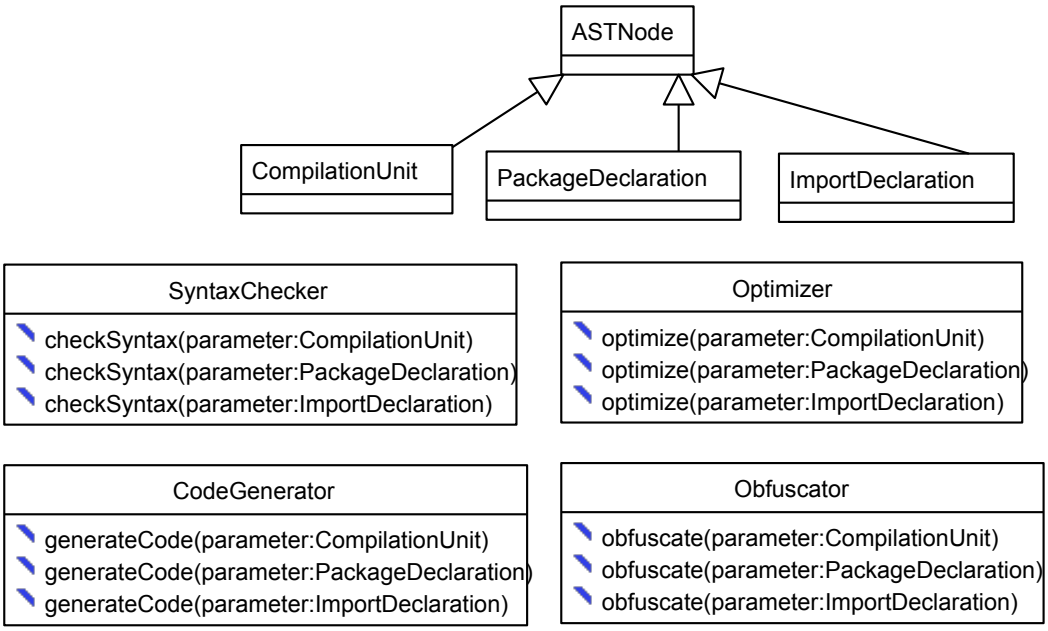


Figure A40: Double dispatch pattern for AST

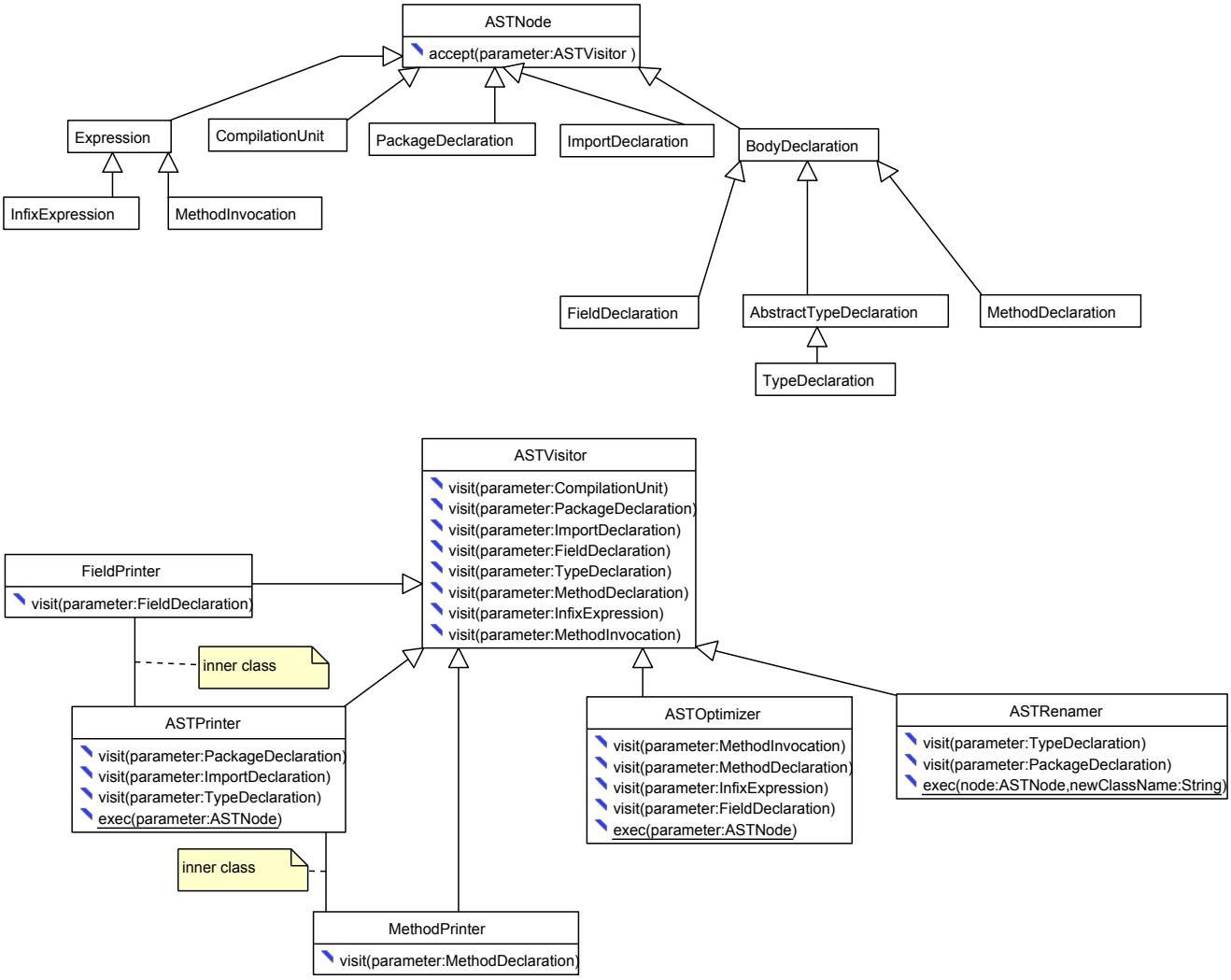


Figure A41: Example how to use Eclipse AST library together with visitor pattern

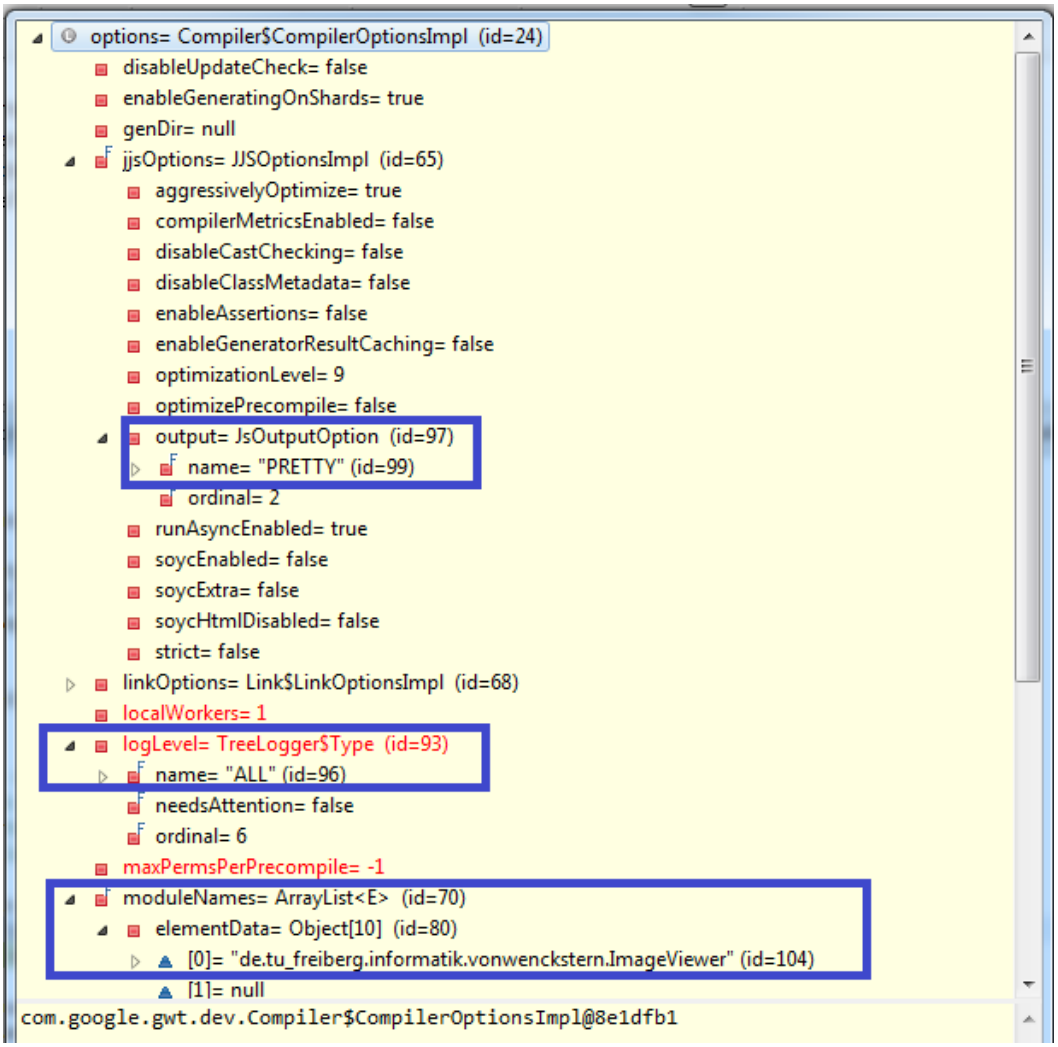


Figure A42: CompilerOptions class attributes for running the compiler with the arguments "-logLevel ALL -style PRETTY de.tu_freiberg.informatik.vonwenckstern.ImageViewer"

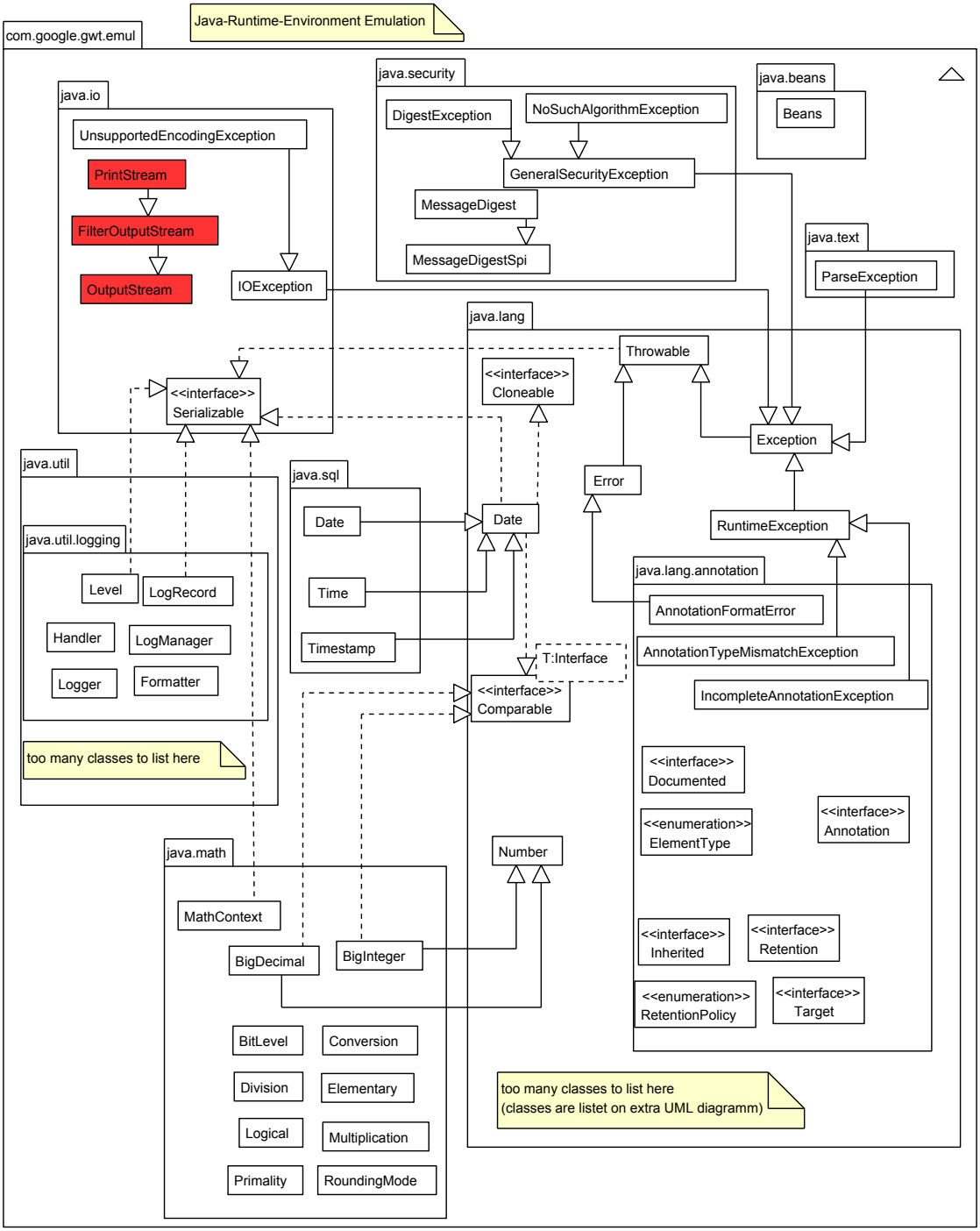


Figure A43: UML diagram of Java-Runtime-Environment Emulation (JREE)
Red classes are "empty" classes, which will be skipped at compile time.

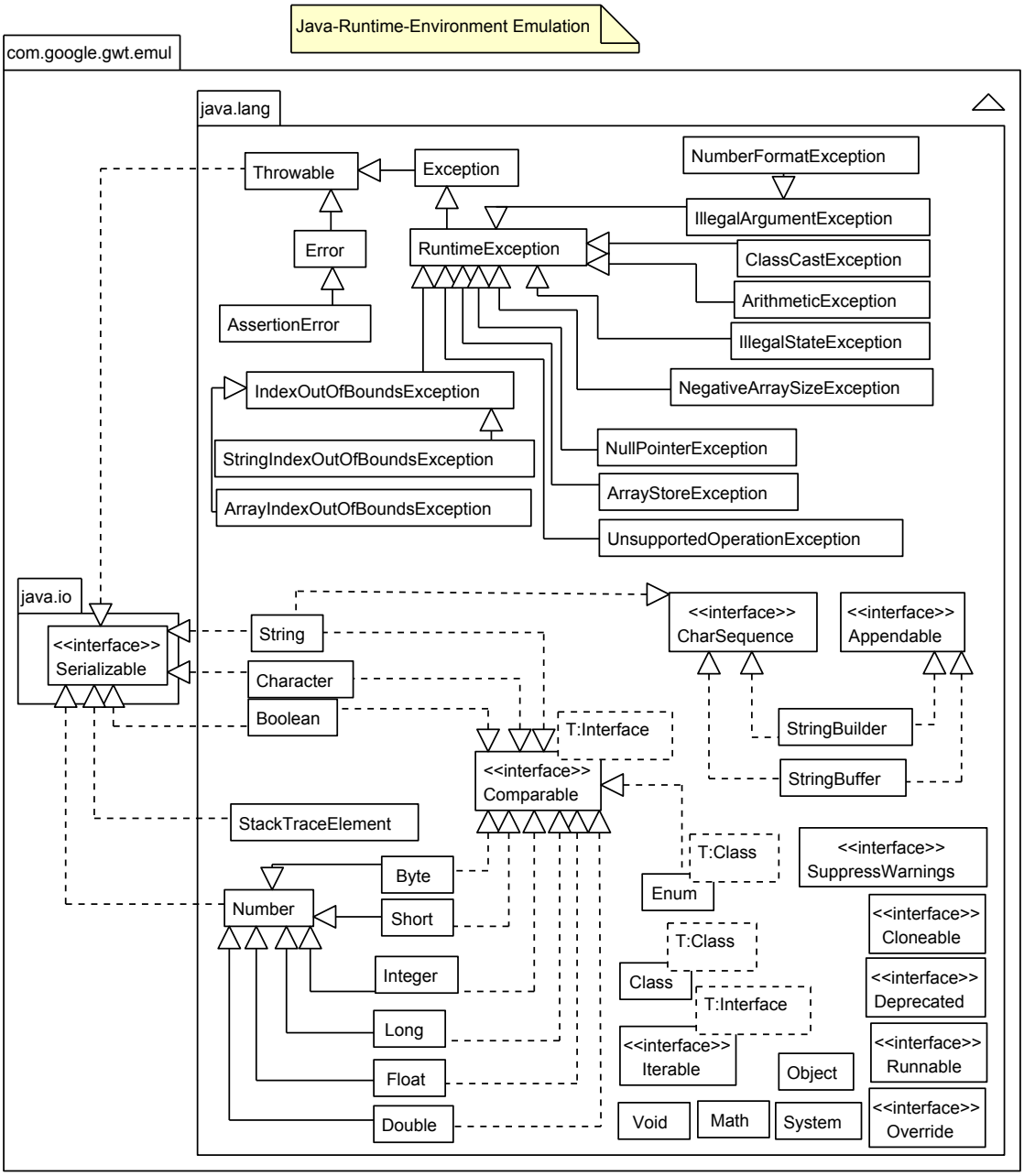


Figure A44: UML diagram of java.lang package of the Java-Runtime-Environment Emulation (JREE)

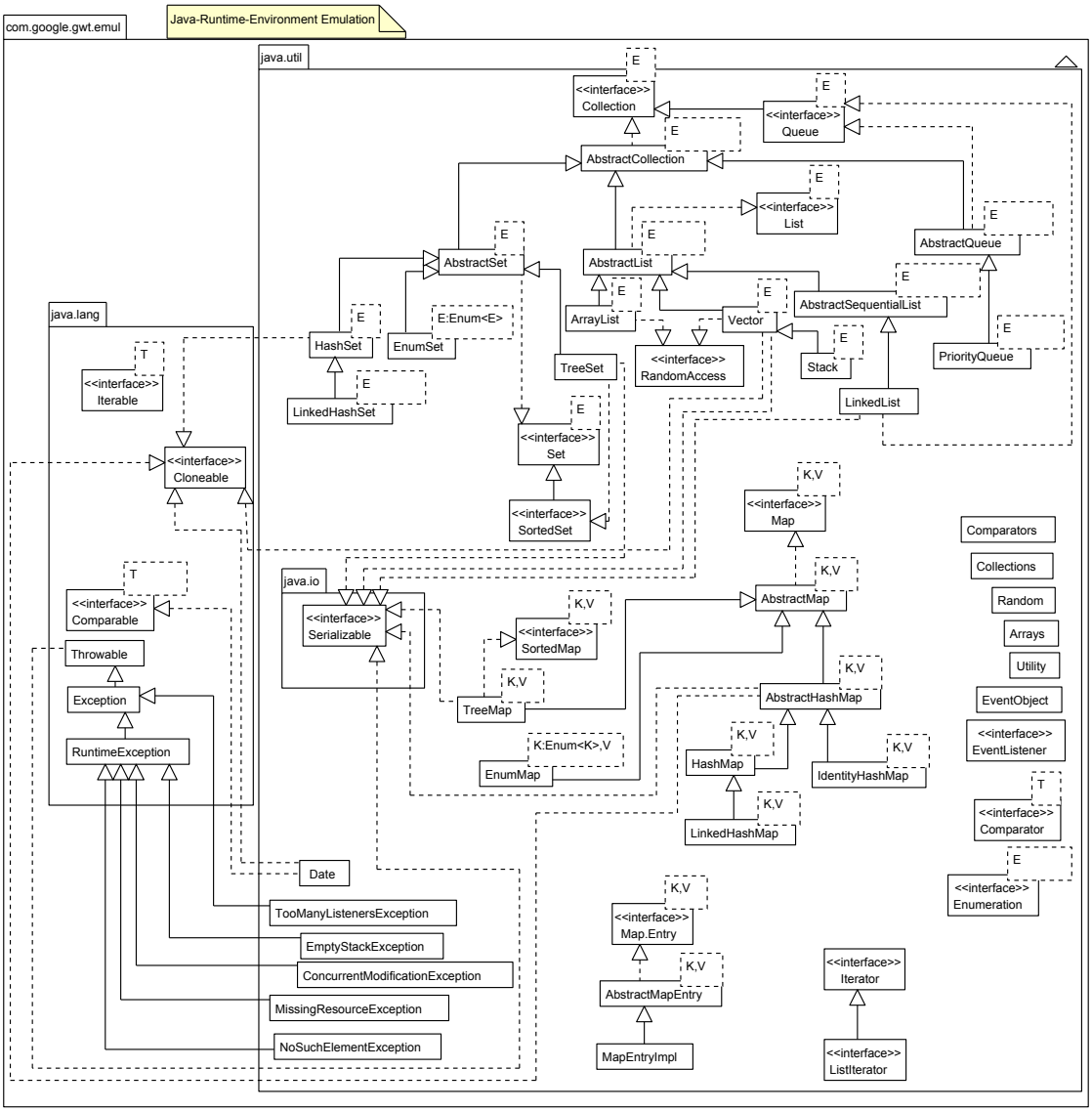


Figure A45: UML diagram of java.util package of the Java-Runtime-Environment Emulation (JREE)

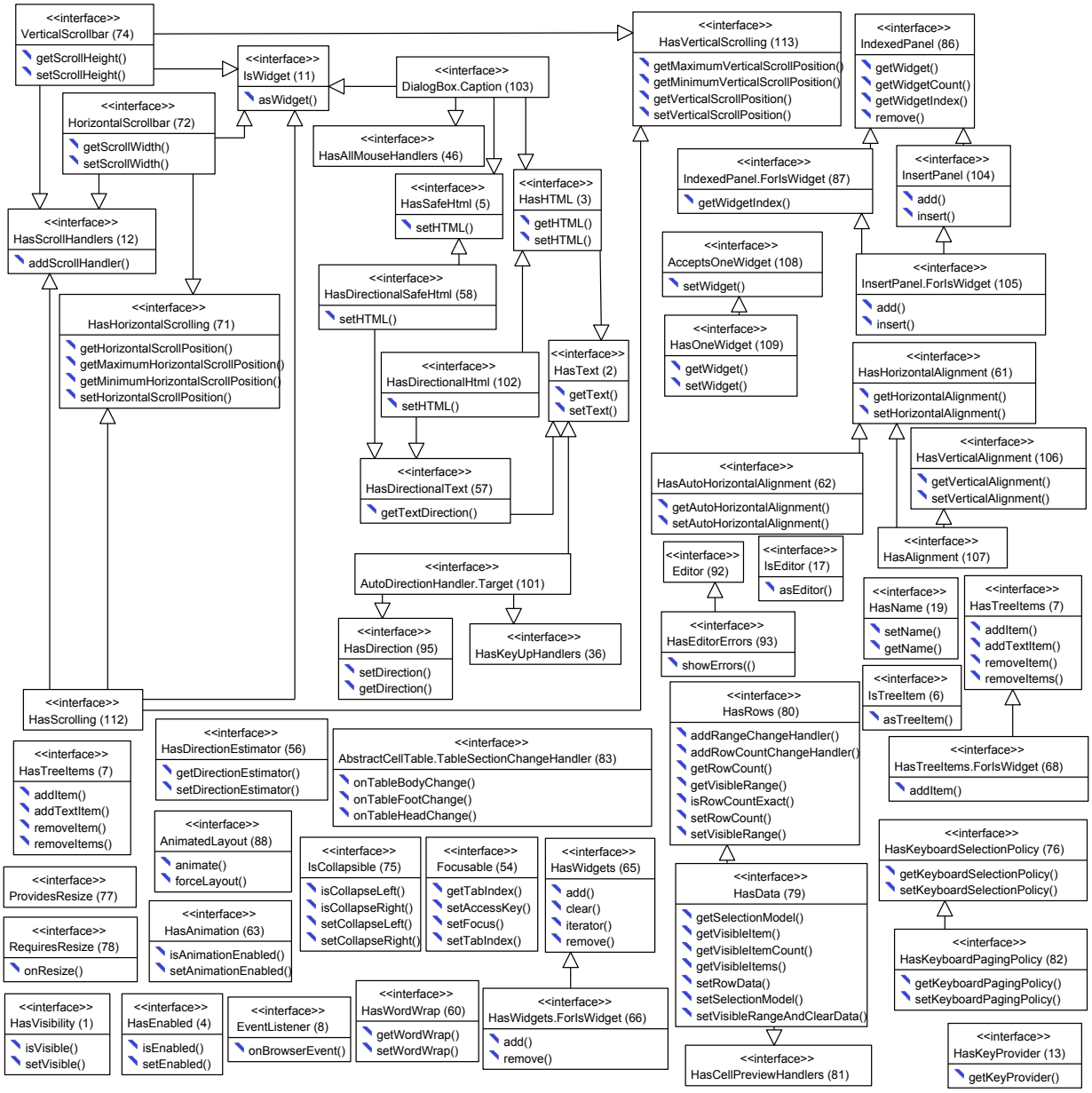


Figure A46: Operation interfaces of GWT widget hierarchy

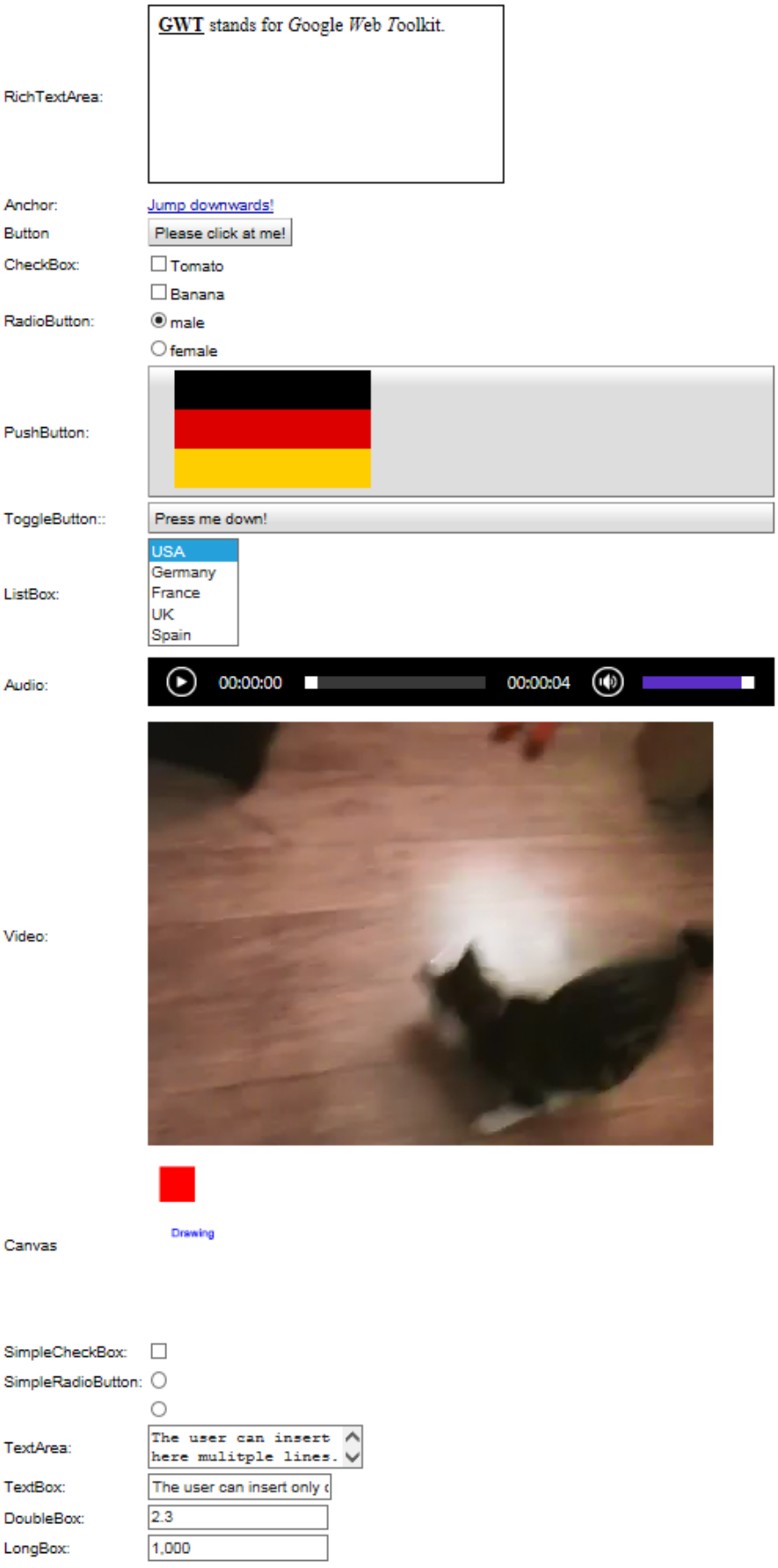


Figure A47: Screenshot of FocusWidget example.

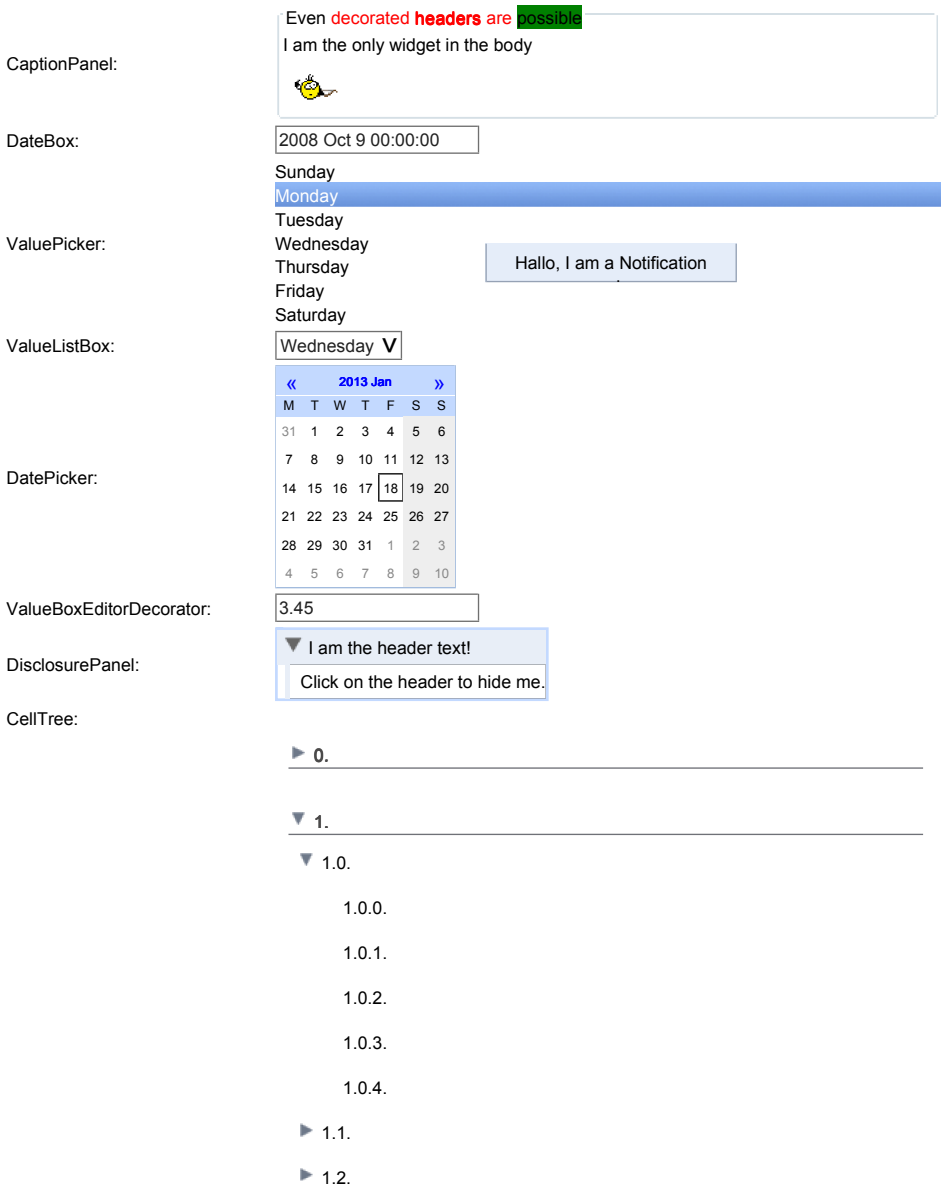


Figure A48: Screenshot of Composite example, part I.

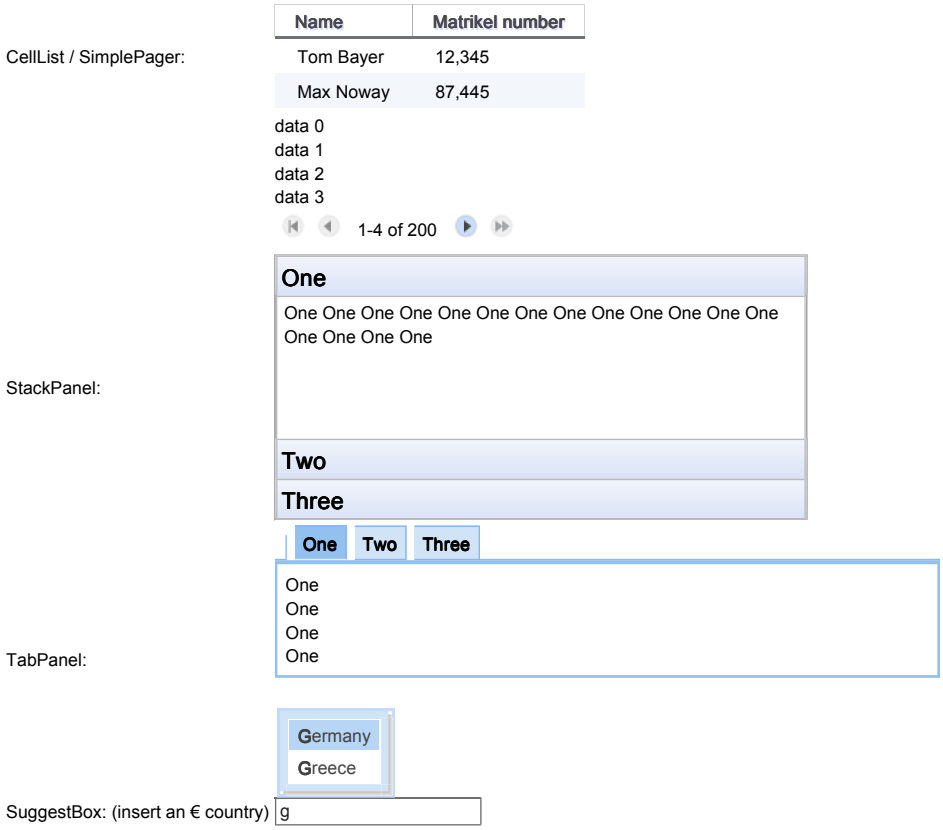


Figure A49: Screenshot of Composite example, part II.

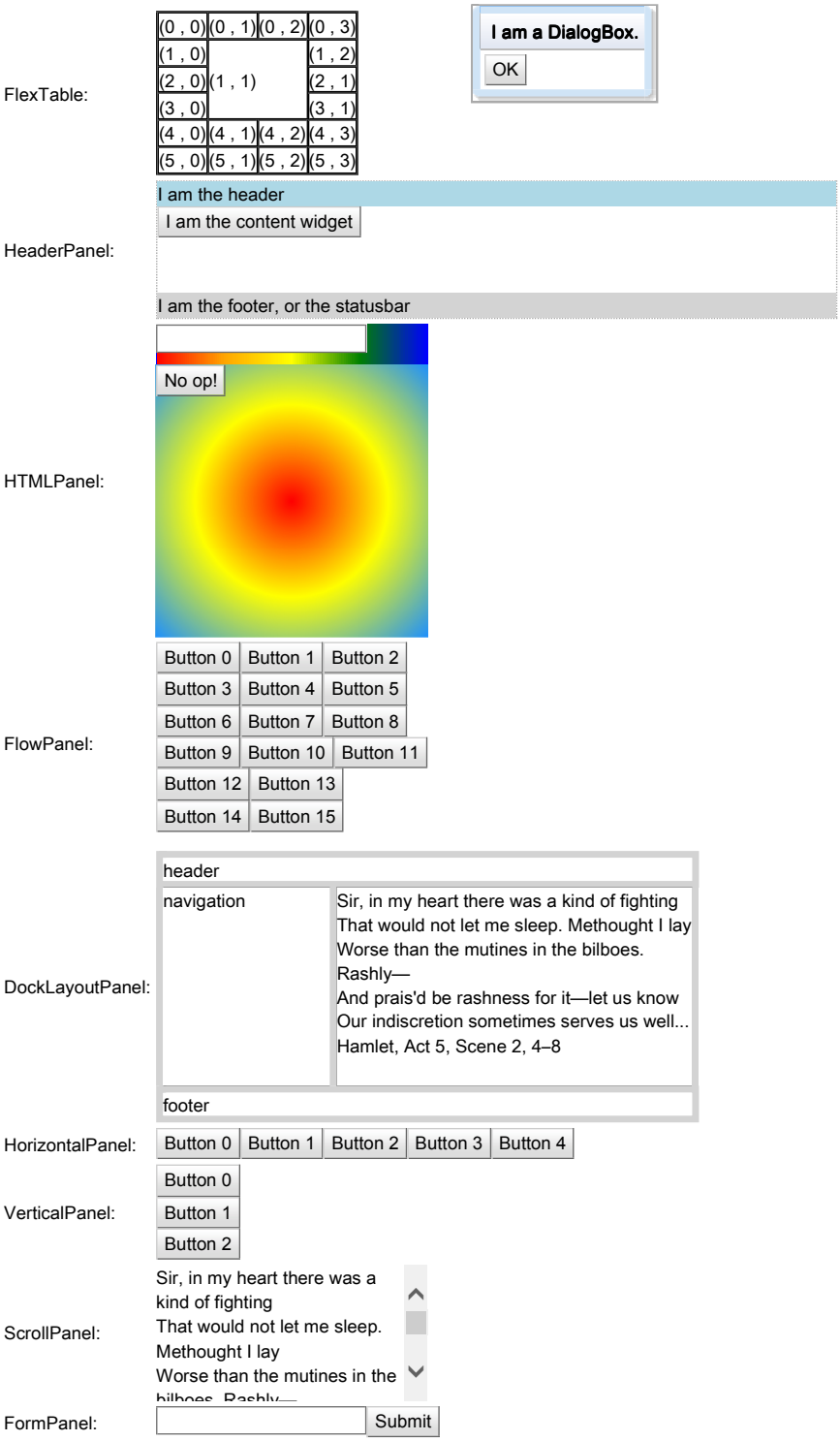


Figure A50: Screenshot of Panel example.

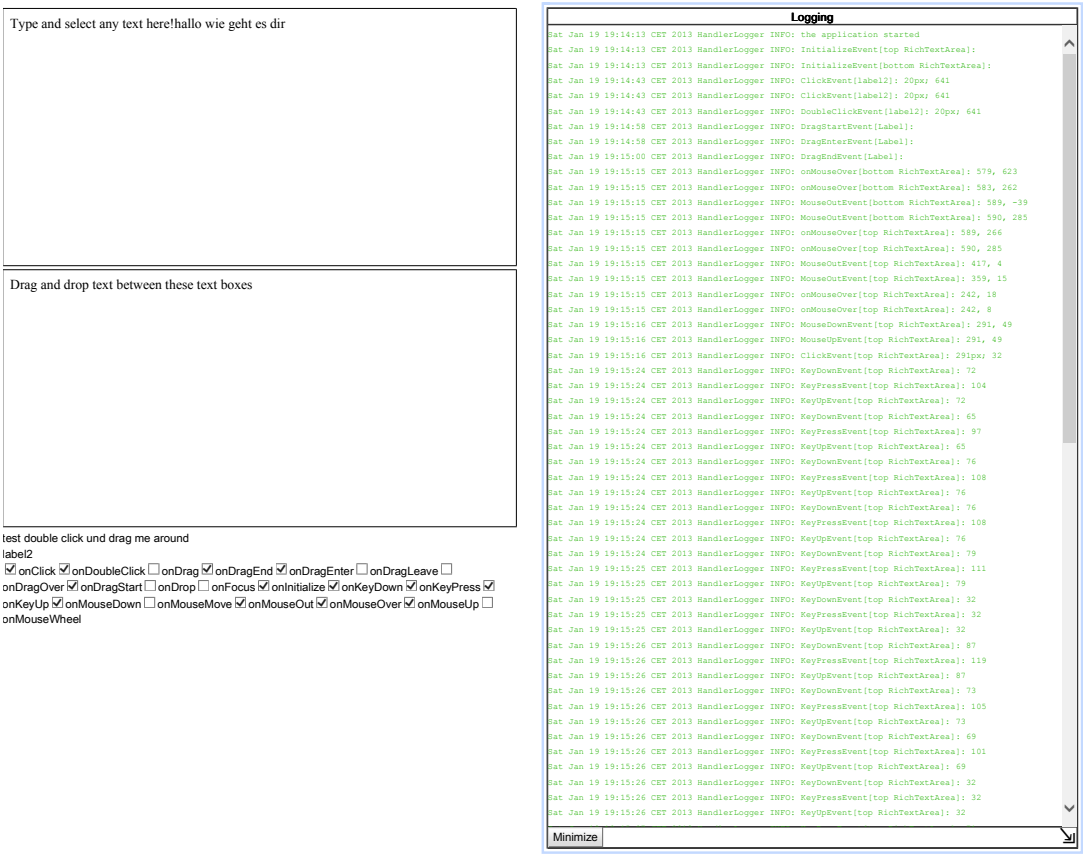


Figure A51: Screenshot of EventHandler example.

(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)

Left click on a cell to select it. Right click to select the entire row.

Figure A52: Screenshot of DOMManipulation example.

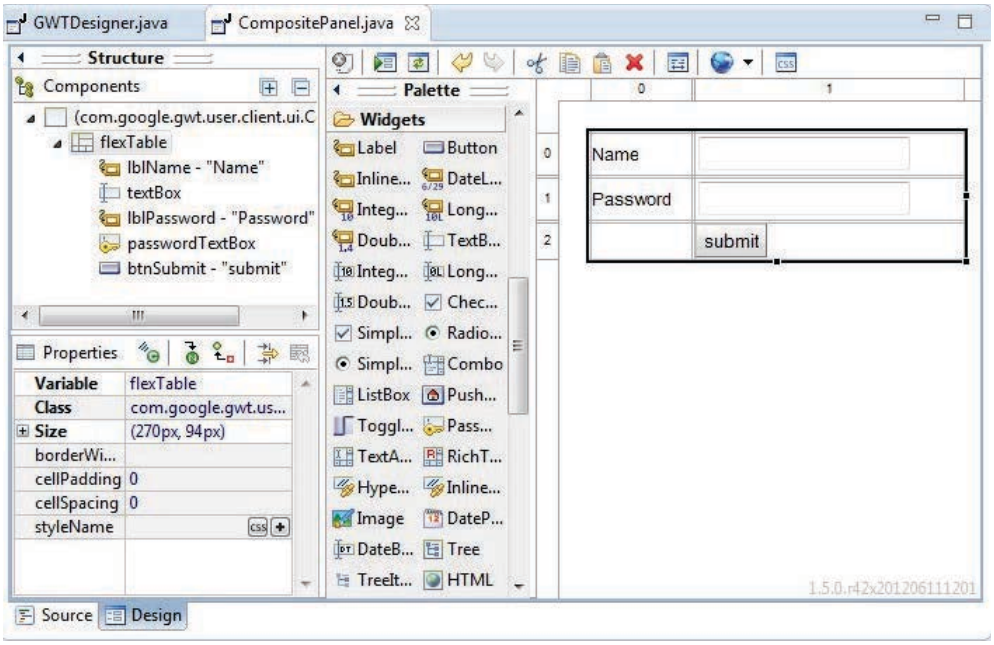


Figure A53: Screenshot of creating view in GWT Designer.

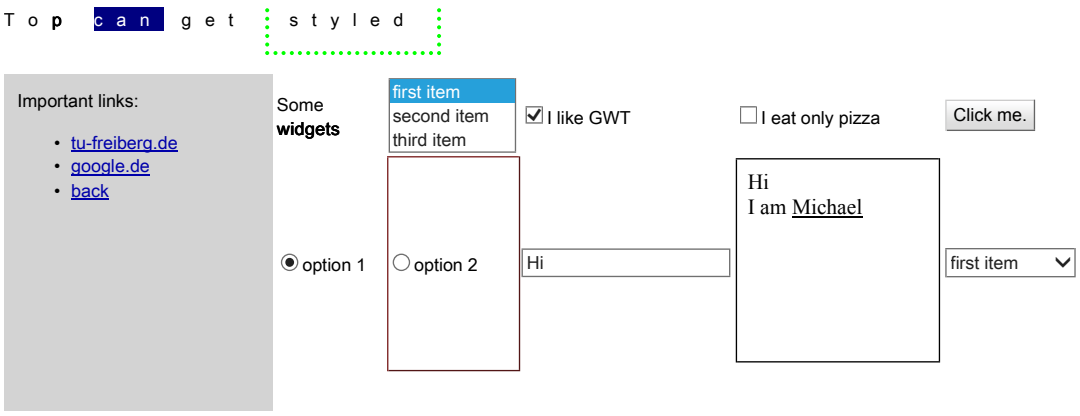


Figure A54: Screenshot of UiBinder example.

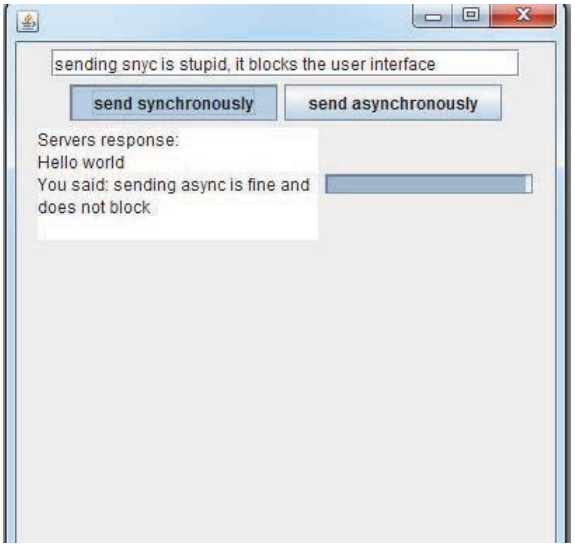


Figure A55: Screenshot of RMI example.

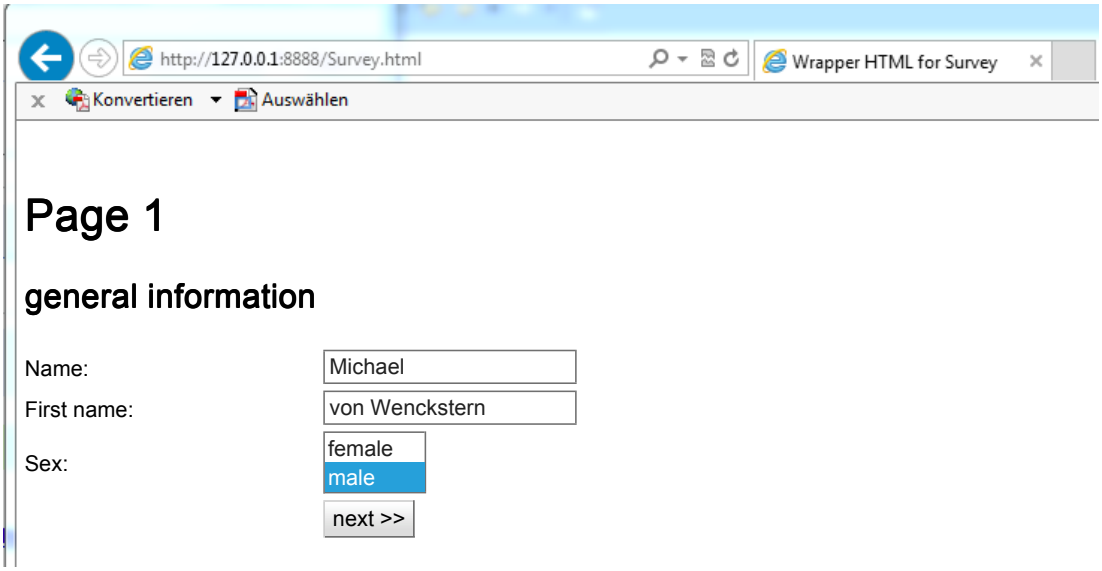


Figure A56: Screenshot of Survey example.
Inserting data for first page. URL contains no data tokens yet.

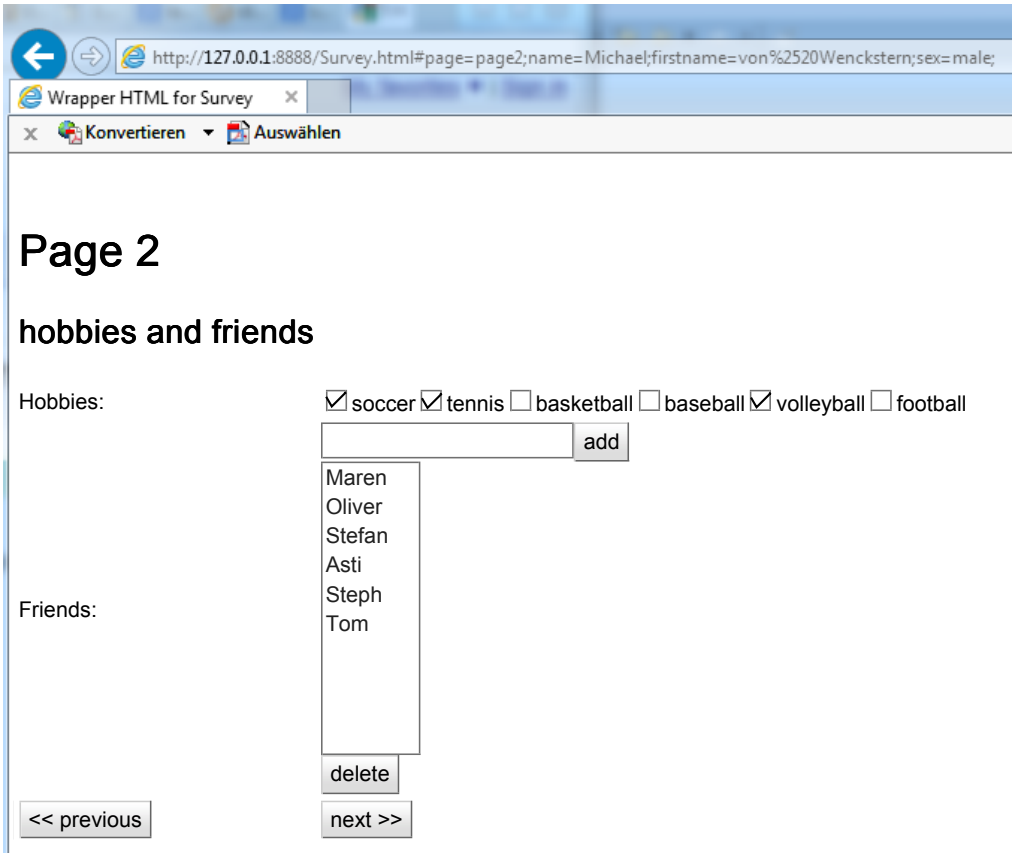


Figure A57: Screenshot of Survey example.
Inserting data for second page. URL contains data tokens of first page yet.

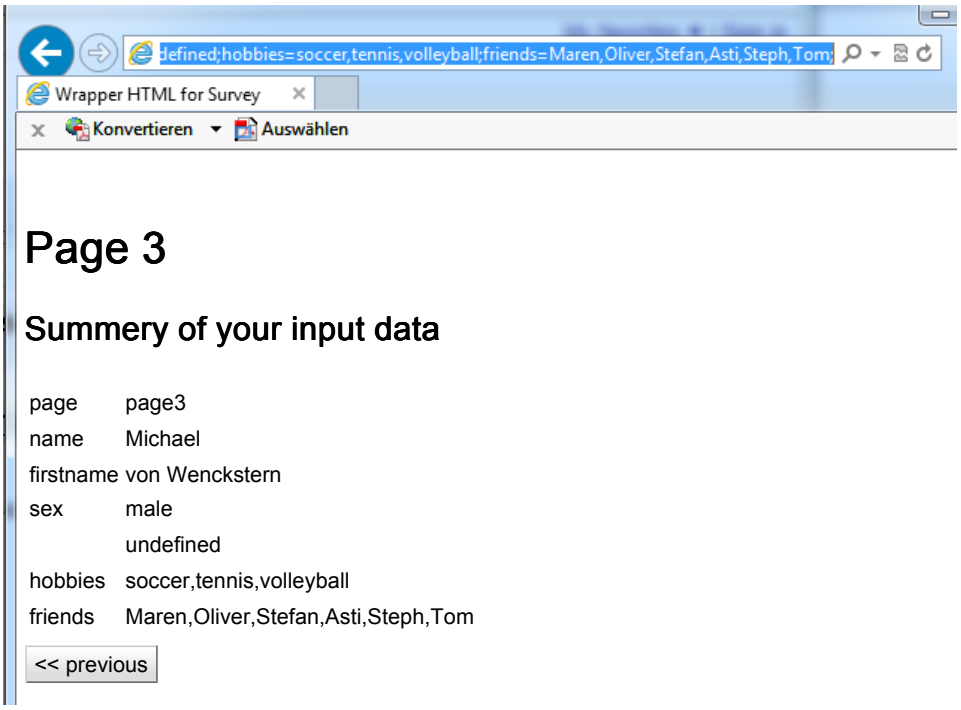


Figure A58: Screenshot of Survey example.
Page three sumerizes all the data stored in the URL. URL contains data tokens of first and second page yet.

Figure A59: Screenshot of Survey example.
User clicked went back to first page using the browser's back button (in the picture is now the browser's forward button enabled after the user pressed the back one)

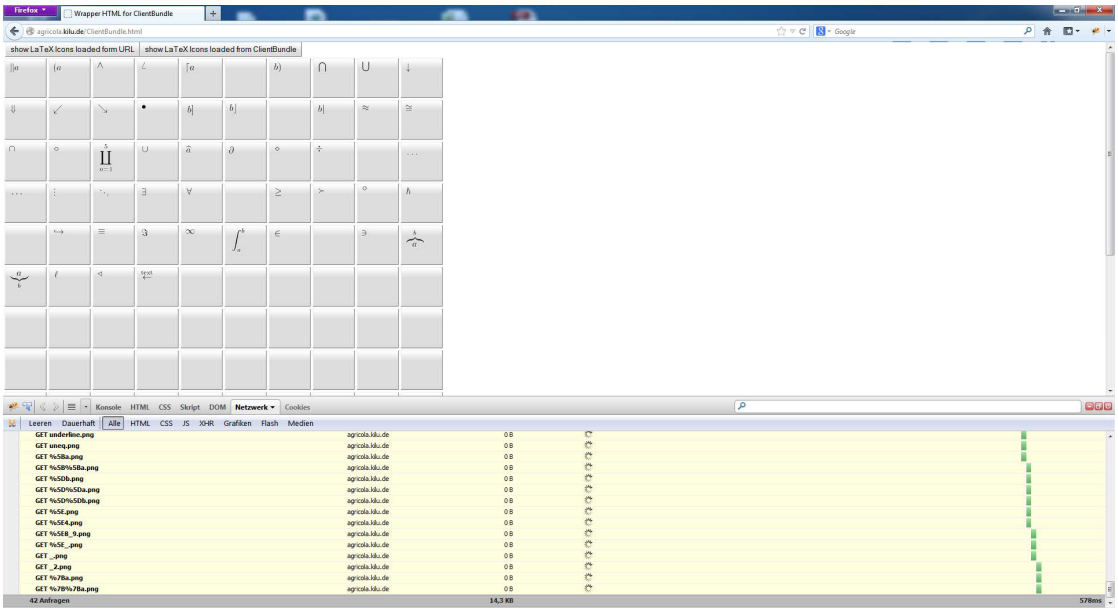


Figure A60: Screenshot of Firefox when it has to load all LaTeX icons from the server

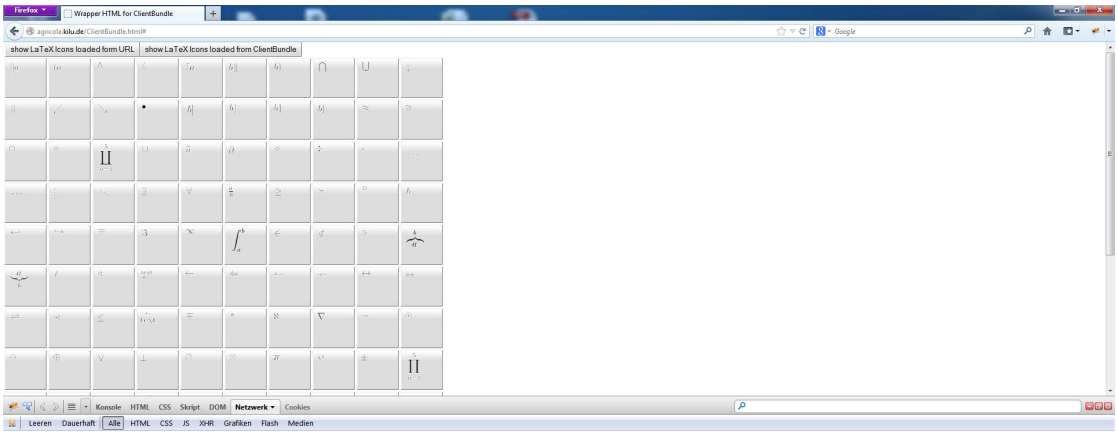


Figure A61: Screenshot of Firefox when it shows the LaTeX icons from the inlined ClientBundle interface

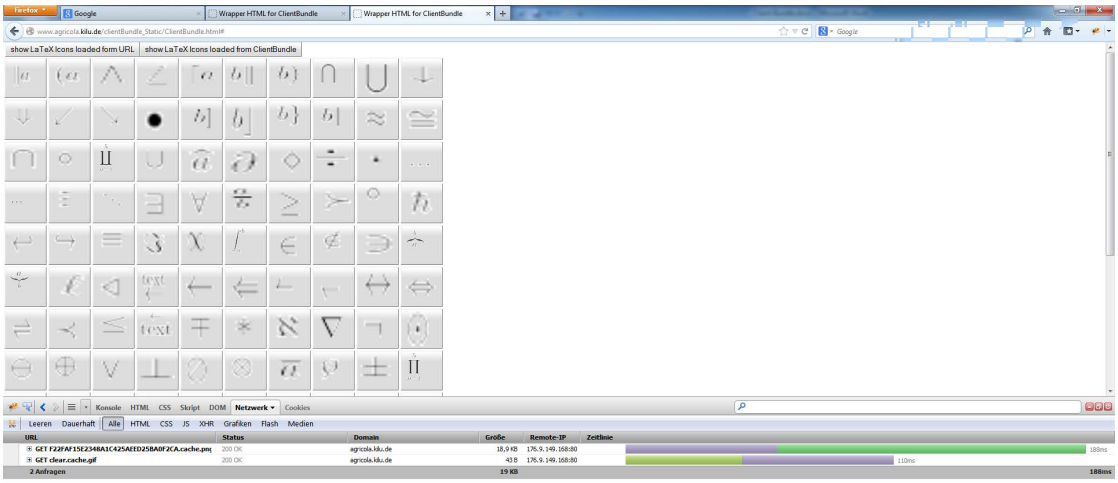


Figure A62: Screenshot of Firefox when it shows the LaTeX icons from the ClientBundle interface with disabled inlining option.

Figure A63: Screenshot of IE 10 without the usage of conditional CSS.

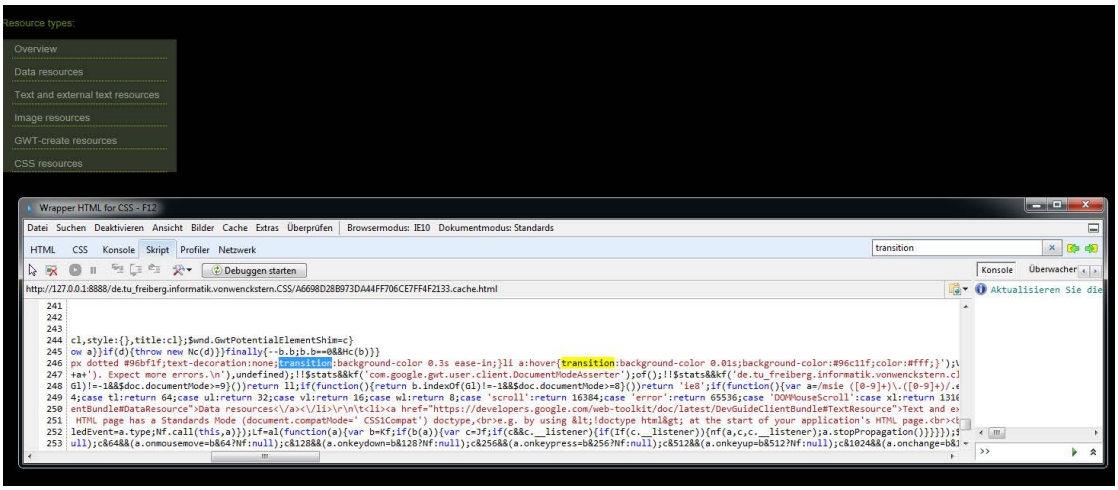


Figure A64: Screenshot of IE 10 with the usage of conditional CSS.

Figure A65: Screenshot of Google Chrome with the usage of conditional CSS.

Figure A66: Screenshot of Firefox with the usage of conditional CSS.

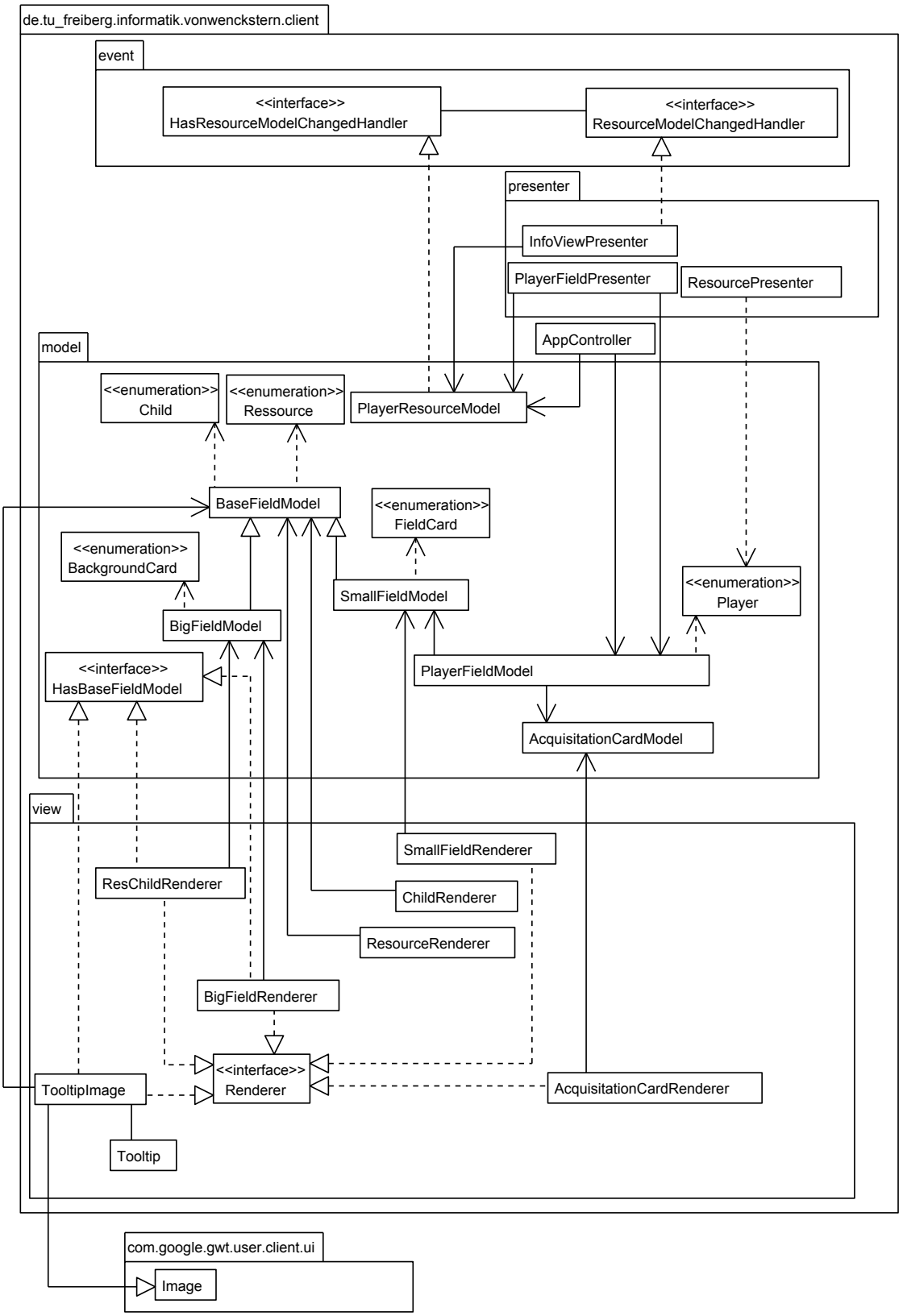


Figure A67: Model classes

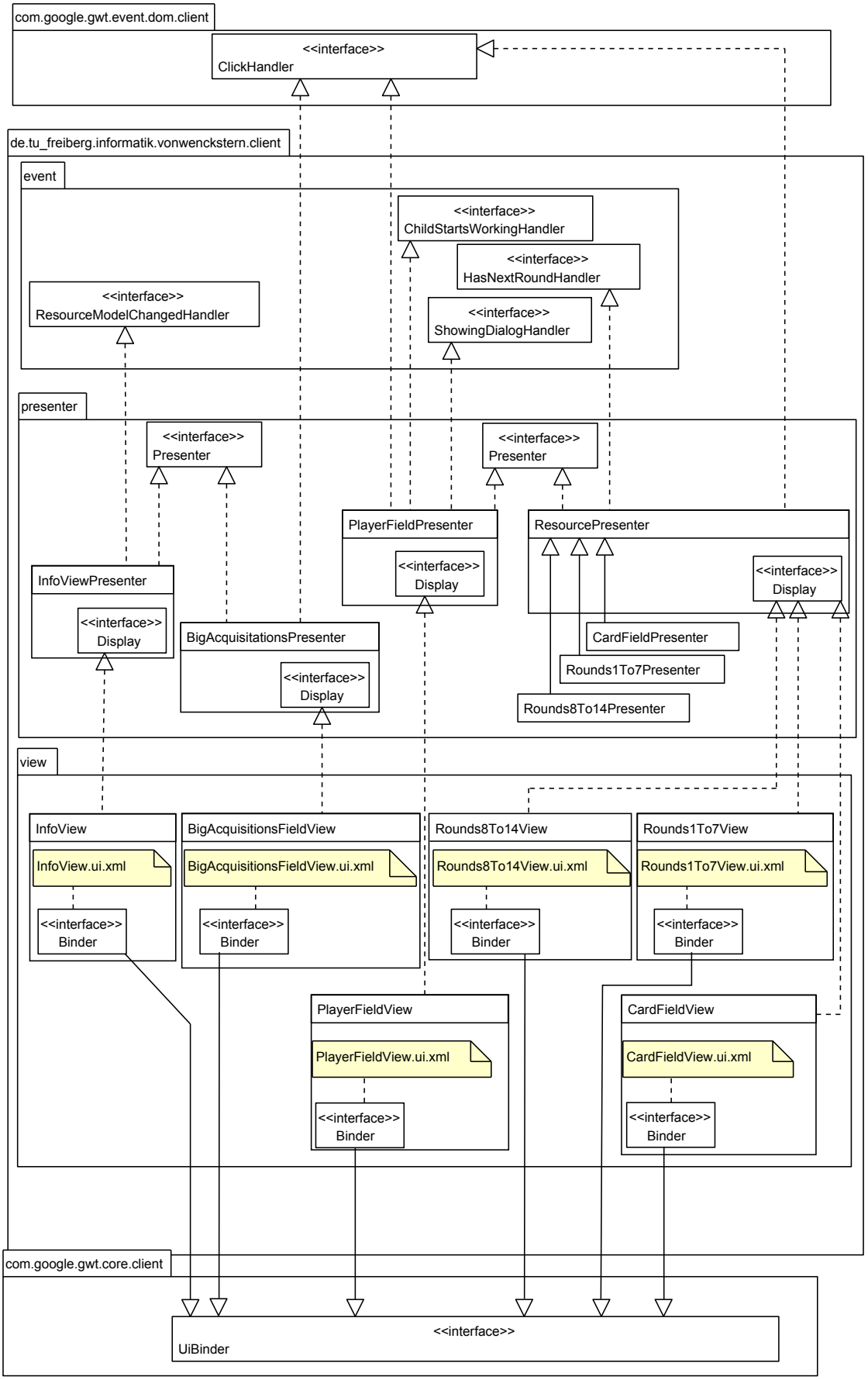


Figure A68: Presenter classes

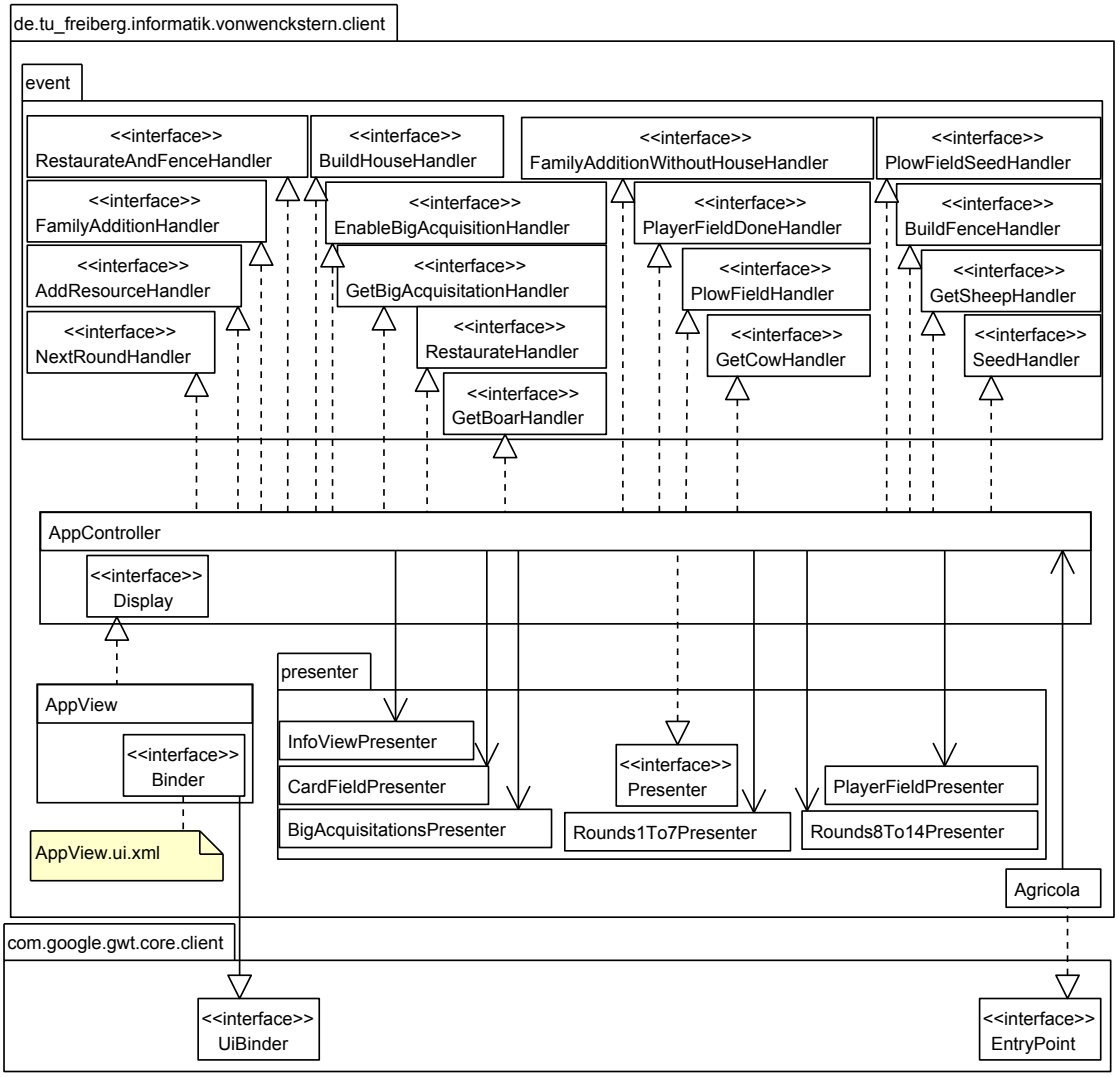


Figure A69: UML diagram displaying the AppController's links to the other presenters

Topic Comments on: You'll miss me when I've gone
http://www.thesun.co.uk/sol/homepage/sport/football/480166
Wenger-tells-fans-Youll-miss-me-when-Im-gone.html

back to topics overview

Milena Burova
Tue Feb 19 00:00:00 CET 2013
Tomorrow's headline: Wenger says it could have been so different.

Undoubtedly, Wenger is and probably will be the greatest manager ever for Arsenal. It would be sad to see the day he leaves the club he has managed to take on a roller coaster ride through his reign. Arsenal fans doubting this man should remember he is the reason why Arsenal are considered such a force in England and he is the main reason why you expect the team to win and play good football because he has set them to those high standards. I understand its disheartening for the clubs, players and fans alike for a club like arsenal when you dont win anything for close to 8 years, but its imperative not to lose faith in the people fighting for it.

avdvyver
Tue Feb 19 00:00:00 CET 2013
As a United fan, when my club loses or even goes without winning trophies, we tend to support our club more because our fans know, that losing is part of the game, but how we rise and fight back is what matters.....and United always fight back . Its imbedded in the DNA of each and every player by the coach and fueled by us fans.

I really hope Arsene wins against Munich today and also handsomely just so that people could get off his case also because I would want another english club progressing in the cup.

Ibelievein_United
Tue Feb 19 00:00:00 CET 2013
Cant wait to watch bayern thrash arsenal in the cl blew their only chance of a trophy in the fa cup

arealmanutdfan
Tue Feb 19 00:00:00 CET 2013
Arsenal are the only club stupid enough to make wenger the highest paid manager in the pl on £7 million a season when he has won nothing since 2005 !!

Manutd1999Triple
Tue Feb 19 00:00:00 CET 2013
If wenger gone..it will be the same musical chair as Chelsea and Man city..4/5 coaches in one year

afc85
Tue Feb 19 00:00:00 CET 2013
The media is always on Wengers back to ridicule him, they put him under more pressure than the EPL! However he has done well with the lack of resources and funding available to compete with billionaire clubs, but this is still not good enough for some fans.

Also its funny how Arsenal FC is one story and Arsene Wenger is always another story, the media and generally all football fans love reading about the Arsenal ship sinking and Wenger is french moaner etc etc.

Gunsandbullets1
Tue Feb 19 00:00:00 CET 2013
Arsene is a top top manager no doubt about that, but I think he contributes to his problems , the sales of Cesc, RVP , Ebue and so many to mention has not be invested in top quality players, rather Arsenal buy average or no name players. He has FFP in mind but why we wait for that to come he should have blend the team up with some good high profile name. and I think he has to break the bank to sign Begovic as the goal keepers in Arsenal are all rubbish with exception they can send the young italian on loan and ship out the 2 poles and the deadwood like Djourou and co. But the man always defends his players in good and bad times.
Although i am a United fan,i have to say that Wenger is not the problem,its just football in general that's the problem,the price tags on players these days are just silly,Arsenal use to be a good team there is no doubt about that.(on their day they still can be)

But the owners are just using the club as a cash cow,as are the Glazers,half of these owners do not really care about football,they only care at how much money the club can make for them.(apart from the obvious 2 other clubs in the league).

ur_nightmare
Tue Feb 19 00:00:00 CET 2013
If you look at half the age of the Arsenal players they are relatively young,and have been purchased relatively cheap,granted some of the signings have not been great,but that's the problem now,great players cost great amount's of money,and if Arsenal are not willing to pay the prices that are on demand,then look to the owners not the manager.

In fact,Wenger has done a great job in keeping Arsenal where they are now,and until the owners allow Wenger to bring in quality players,then the results are not just going to happen.

If Wenger was to get the boot,i can't see there being any other manager out there,that can do what he has done in terms of keeping Arsenal in and around the top 3/4/5 positions.

Manu man
Tue Feb 19 00:00:00 CET 2013
ha ha yes we will, he makes us laugh so much in the big clubs....loser goonies manager needs a holiday fast....or he will start singing I am a tea pot ....

When I was just a little MANAGER I asked THE BOARD

what will I GET? Will IT BE MONEY

JessicaHolland
Tue Feb 19 00:00:00 CET 2013
Will I be $RICH$ Here's what THEY said to me.

Que WENGER ,WENGER,,

Whatever will be, will be .....Bayern Munich BOOM The End

planetharris
Tue Feb 19 00:00:00 CET 2013
The pressure for him to leave come from the media and pundits who influence weak minds! Will the pressure really mount, if we can't beat Bayern? All the experts are saying we can't win, so where's the pressure? Admittedly we do need to beat Villa!

If we win 6-0 tonight, it will not make up for giving away a chance at the FA cup. Wenger (not Kroenke, Gazidis or Hill-Wood) made the decision to start with a weak side as he has done so many times before. As as we have so many times before, we turned a winning team into a losing one.

gunnerjas
Tue Feb 19 00:00:00 CET 2013
Almost every season that we have competed in the Champions league, we have collapsed at this stage as Wenger starts rotating the squad too much.

Time for Wenger to move upstairs and give someone new a chance. My money is on David Moyes.

johnchallener
Tue Feb 19 00:00:00 CET 2013
Yes Arsen...But the Shareholders will miss you even more !

Why am I suddenly getting the feeling that AW thinks he is God's gift to Arsenal? I have admired the man for what he HAS done for Arsenal BUT.....that was then this is now. He has not proven himself for the past 8 years and Arsenal have won nothing and although I never believe the manager should always take the kicks for what happens on the field a manager MUST have one thing and that is motivation. It seems to me that he has lost this skill and the players go merrily on their way also believing THEY are somebody special. Well, either AW is wrong OR the players are. I think its AW.

Mr Kipling
Tue Feb 19 00:00:00 CET 2013
AW, you will NOT be missed when you are gone. ALL managers must go sometime and I think you should get used to the idea that your days at over. I have watched Arsenal since I was 15 when Tom Whittaker was manager and as I recall none of them stayed as long as AW. Therein lies the problem. AW is no longer the man or the manager he was and it is time to go but HE doesn't want to. He MUST consider the fans, they are the ones who week after week wonder if they are going to win their next game. That is wrong. AW has to sit down with himself and answer a lot of questions about himself and all those that come from the fans. Most of them he won't like. The fans come first - they always have. However, AW must also understand that what he HAS done for Arsenal in the golden years cannot sustain his reputation. Football lives from one season to another and it time that AW realised that his time has come.

Mahobhojohnfisher
Tue Feb 19 00:00:00 CET 2013
@mrdynamite this media plays a dirty hand, the same one that lost England hosting the World cup. They thrive on negative reporting (as it sells more). Wenger may not be winning now but he is one of the best managers, but like any human he feels the pressure and the media guys are making him feel it more, but only trolls like u find their treatment of him as something to laugh at @russbinoz u and which real gunners??

Jack the hat

Figure A70: Screenshot illustrating the topic entries of "Comments on: You'll miss me when I've gone"

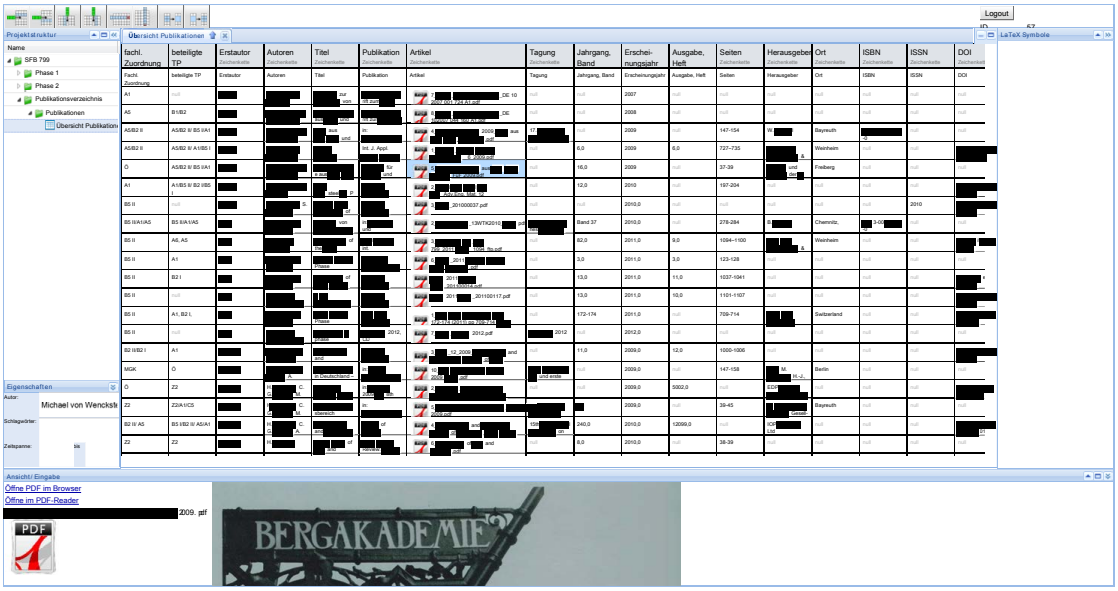


Figure A71: Screenshot of the new SFB799 web application

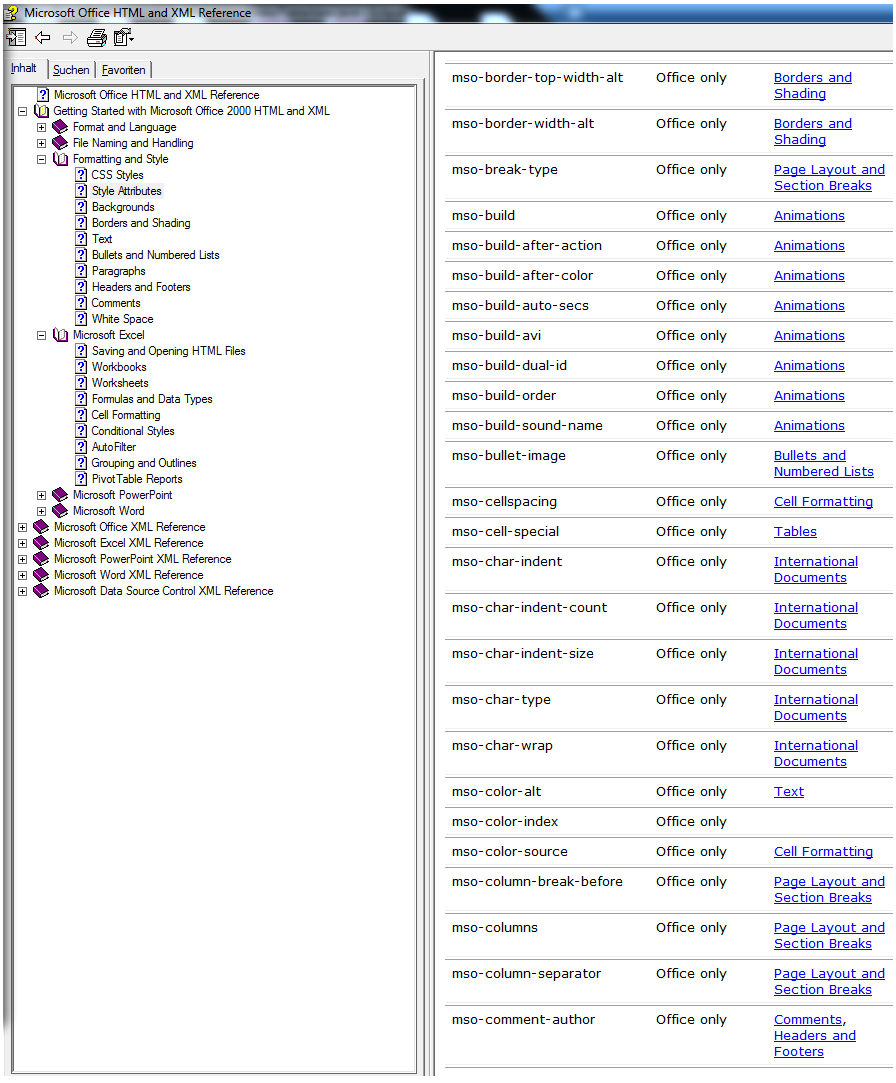


Figure A72: Screenshot of Microsoft Office specific style definitions in the help file

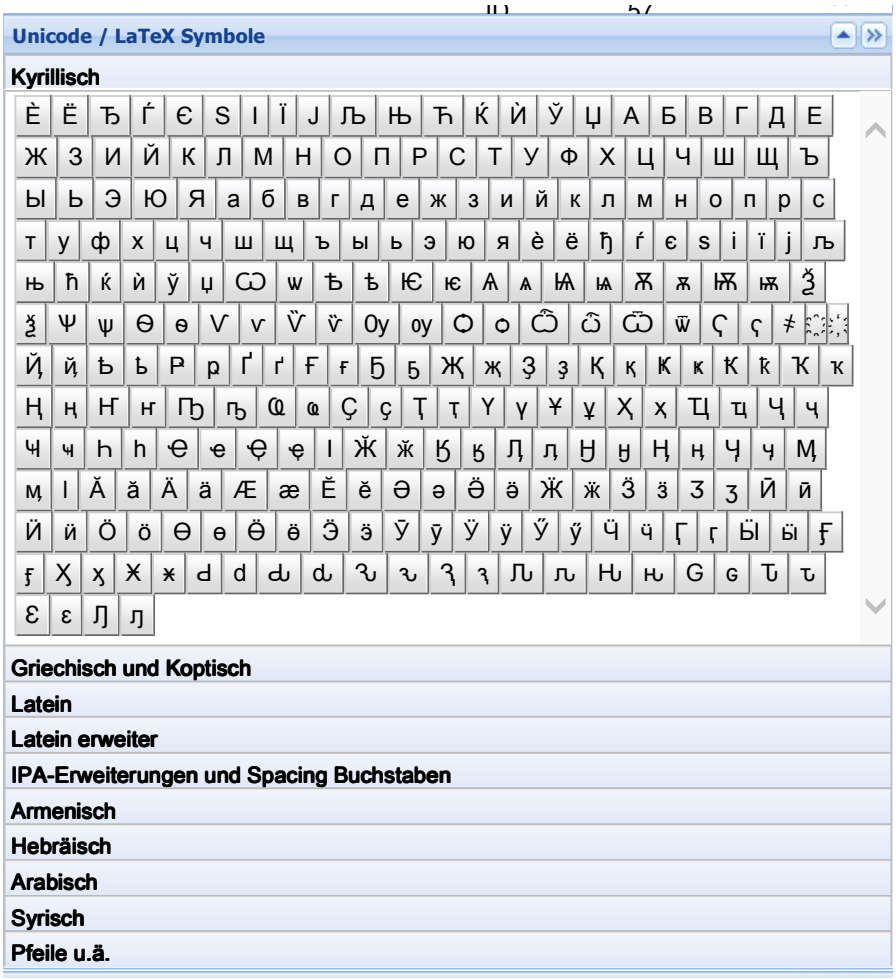


Figure A73: Screenshot of widget which gets generated depending on the used Java annotations.

A 3 Listings

```
<HEADER>
<TITLE>The World Wide Web project</TITLE>
<NEXTID N="55">
</HEADER>
<BODY>
<H1>World Wide Web</H1>The WorldWideWeb (W3) is a wide-area<A
NAME=0 HREF="WhatIs.html">
hypermedia</A> information retrieval
initiative aiming to give universal
access to a large universe of documents.<P>
Everything there is online about
W3 is linked directly or indirectly
to this document, including an <A
NAME=24 HREF="Summary.html">executive
summary</A> of the project, <A
NAME=29 HREF="Administration/Mailing/Overview.html">Mailing lists</A>
, <A
NAME=30 HREF="Policy.html">Policy</A> , November's <A
NAME=34 HREF="News/9211.html">W3 news</A> ,
<A
NAME=41 HREF="FAQ/List.html">Frequently Asked Questions</A> .
<DL>
<DT><A
NAME=44 HREF="../DataSources/Top.html">What's out there?</A>
<DD> Pointers to the
world's online information,<A
NAME=45 HREF="../DataSources/bySubject/Overview.html"> subjects</A>
>
, <A
NAME=z54 HREF="../DataSources/WWW/Servers.html">W3 servers</A>,
etc.
<DT><A
NAME=46 HREF="Help.html">Help</A>
<DD> on the browser you are using
<DT><A
NAME=13 HREF="Status.html">Software Products</A>
<DD> A list of W3 project
components and their current state.
(e.g. <A
NAME=27 HREF="LineMode/Browser.html">Line Mode</A> ,X11 <A
NAME=35 HREF="Status.html#35">Viola</A> , <A
NAME=26 HREF="NeXT/WorldWideWeb.html">NeXTStep</A>
, <A
NAME=25 HREF="Daemon/Overview.html">Servers</A> , <A
NAME=51 HREF="Tools/Overview.html">Tools</A> ,<A
NAME=53 HREF="MailRobot/Overview.html"> Mail robot</A> ,<A
NAME=52 HREF="Status.html#57">
Library</A> )
<DT><A
NAME=47 HREF="Technical.html">Technical</A>
<DD> Details of protocols, formats,
program internals etc
<DT><A
```

```
52 NAME=40 HREF="Bibliography.html">Bibliography</A>
53 <DD> Paper documentation
54 on W3 and references.
55 <DT><A
56 NAME=14 HREF="People.html">People</A>
57 <DD> A list of some people involved
58 in the project.
59 <DT><A
60 NAME=15 HREF="History.html">History</A>
61 <DD> A summary of the history
62 of the project.
63 <DT><A
64 NAME=37 HREF="Helping.html">How can I help</A> ?
65 <DD> If you would like
66 to support the web..
67 <DT><A
68 NAME=48 HREF="../README.html">Getting code</A>
69 <DD> Getting the code by<A
70 NAME=49 HREF="LineMode/Defaults/Distribution.html">
71 anonymous FTP</A> , etc.</A>
72 </DL>
73 </BODY>
```

Listing A3: HTML code of first website, source [Tim12]

```
1 <!DOCTYPE html>
2 <html>
3 <head><title>DOM Manipulating Example</title>
4 <style>
5 svg { background-color:#b0c4de; width: 500px; height: 500px;
      overflow:hidden; }
6 span { width: 70px; display: inline-block; }
7 .layers { position: absolute; left: 550px; top: 10px; }
8 #shapes { position: absolute; left: 550px; top: 310px; }
9 #left { position: absolute; left: 10px; top: 550px; }
10 #up { position: absolute; left: 50px; top: 520px; }
11 #down { position: absolute; left: 50px; top: 580px; }
12 #right { position: absolute; left: 85px; top: 550px; }
13 #props { position: absolute; left: 150px; top: 520px; }
14 #styles { position: absolute; left: 420px; top: 520px; }
15 </style>
16 <script>
17 var propAttr = ['x','y','width','height','cx','cy','r','rx','ry','
      x1','y1','x2','y2'];
18 var styleAttr = ['fill','stroke','stroke-width', 'fill-opacity', '
      stroke-opacity'];
19 var posAttrX = ['x','cx','x1','x2' ];
20 var posAttrY = ['y','cy','y1','y2' ];
21 var shapeArray = [ ['Rectangle','rect',['x','y','width','height
      ']], ['Circle','circle',['cx','cy','r']],['Ellipse','ellipse
      ',['cx','cy','rx','ry']],['Line','line',['x1','y1','x2','y2
      ']],['Text','text',['x','y','html-text']] ];
22 var layers;
23
24 document.onkeydown = keyDown;
25
26 function createRadioButtons() {
```

```
27    layers = [ document.getElementById('circ1') ];
28    var s = document.getElementById('shapes');
29    var b = document.getElementById('add');
30    var index;
31    for(index = 0; index < shapeArray.length; ++index) {
32      var radioInput = document.createElement('input');
33      radioInput.setAttribute('type', 'radio');
34      radioInput.setAttribute('name', 'shapes');
35      radioInput.setAttribute('onclick', 'showProps();');
36      radioInput.setAttribute('id', shapeArray[index][1]);
37      s.insertBefore(radioInput,b);
38      s.insertBefore(document.createTextNode(shapeArray[index][0]),b
          );
39      s.insertBefore(document.createElement('br'),b);
40    }
41    document.getElementById(shapeArray[0][1]).checked = true;
42    showProps();
43  }
44
45  function removeAllChildren(id) {
46    var node = document.getElementById(id);
47    var len = node.childNodes.length;
48    for(var i=len-1; i>=0; --i) {
49      if(node.childNodes[i].className == 'inProp') {
50        node.removeChild(node.childNodes[i]);
51      }
52    }
53  }
54
55  function addProps(props, value) {
56    var p = document.getElementById('props');
57    var textInput = document.createElement('input');
58    textInput.setAttribute('type', 'text');
59    textInput.setAttribute('id', props);
60    textInput.className = 'inProp';
61    if(value != null) {
62      textInput.setAttribute('value', value);
63    }
64    var span = document.createElement('span');
65    span.innerHTML = props;
66    span.className = 'inProp';
67    p.appendChild(span);
68    p.appendChild(textInput);
69    var br = document.createElement('br');
70    br.className = 'inProp';
71    p.appendChild(br);
72  }
73
74  function showProps() {
75    var p = document.getElementById('props');
76    for(var index = 0; index < shapeArray.length; ++index) {
77      if(document.getElementById(shapeArray[index][1]).checked) {
78        removeAllChildren('props');
79        var props = shapeArray[index][2];
80        for(var i=0; i<props.length; ++i) {
81          addProps(props[i], null);
```

```
82        }
83        break;
84      }
85    }
86  }
87
88  function add() {
89    var p = [document.getElementById('props'),document.
          getElementById('styles')];
90    var svg = document.getElementById('svg');
91    var shape = null;
92    var index;
93    for(index = 0; index < shapeArray.length; ++index) {
94      if(document.getElementById(shapeArray[index][1]).checked) {
95        shape = document.createElementNS('http://www.w3.org/2000/svg
              ', shapeArray[index][1]);
96        break;
97      }
98    }
99    if(shape == null) return;
100   for(var k=0; k<p.length; ++k) {
101     for(var i=0; i<p[k].childNodes.length; ++i) {
102       var n = p[k].childNodes[i];
103       if(n.tagName == 'INPUT') {
104         if(n.value.length == 0) {
105           alert('Did not fill out all property fields');
106           return;
107         }
108         if(n.id == 'html-text') {
109           shape.appendChild(document.createTextNode(n.value));
110         } else {
111           shape.setAttribute(n.id, n.value);
112         }
113       }
114     }
115   }
116   layers.push(shape);
117   var l = document.createElement('option');
118   l.id = 'layer' + layers.length;
119   l.innerHTML = shapeArray[index][0];
120   document.getElementById('layers').appendChild(l);
121   svg.appendChild(shape);
122 }
123
124 function layerSelChanged() {
125   var la = document.getElementById('layers');
126   var id = la.childNodes[la.selectedIndex].id;
127   var shape = layers[parseInt(id.substr(5))-1];
128   var a = shape.attributes;
129   removeAllChildren('props');
130   for(var i=0; i<a.length; ++i) {
131     if(propAttr.indexOf(a[i].nodeName) != -1) {
132       addProps(a[i].nodeName, a[i].nodeValue);
133     }
134     if(styleAttr.indexOf(a[i].nodeName) != -1) {
135       document.getElementById(a[i].nodeName).setAttribute('value',
```

```
            a[i].nodeValue);
136       }
137     }
138     if(shape.tagName == "text" && shape.firstChild != null) {
139       addProps("html-text", shape.firstChild.data);
140     }
141   }
142
143   function changeProps() {
144     var la = document.getElementById('layers');
145     var id = la.childNodes[la.selectedIndex].id;
146     var shape = layers[parseInt(id.substr(5))-1];
147     var a = shape.attributes;
148     for(var i=0; i<a.length; ++i) {
149       if(propAttr.indexOf(a[i].nodeName) != -1) {
150         a[i].nodeValue = document.getElementById(a[i].nodeName).
              value;
151       }
152       if(styleAttr.indexOf(a[i].nodeName) != -1) {
153         a[i].nodeValue = document.getElementById(a[i].nodeName).
              value;
154       }
155     }
156     if(shape.tagName == "text" && shape.firstChild != null) {
157       shape.firstChild.data = document.getElementById("html-text").
            value;
158     }
159   }
160
161   function move(dx,dy) {
162     var la = document.getElementById('layers');
163     var id = la.childNodes[la.selectedIndex].id;
164     var shape = layers[parseInt(id.substr(5))-1];
165     var a = shape.attributes;
166     for(var i=0; i<a.length; ++i) {
167       if(posAttrX.indexOf(a[i].nodeName) != -1) {
168         a[i].nodeValue = Number(a[i].nodeValue) + Number(dx);
169       }
170       if(posAttrY.indexOf(a[i].nodeName) != -1) {
171         a[i].nodeValue = Number(a[i].nodeValue) + Number(dy);
172       }
173     }
174   }
175
176   function keyDown(e) {
177     switch(e.keyCode) {
178       case 38: move(0,-10); break;
179       case 40: move(0,10); break;
180       case 37: move(-10,0); break;
181       case 39: move(10,0); break;
182       default: break;
183     }
184   }
185
186   </script>
187   </head>
```

```
188 <body onLoad="createRadioButtons()">
189 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" id="svg">
190    <circle cx="100" cy="50" r="40" stroke="#ffffff"
191    stroke-width="2" fill="#ff0000" id="circ1">
192  </svg>
193  <div class="layers">Layers:<br><select size="12" id="layers"
       onchange="layerSelChanged();"><option id="layer1" selected>
       Circle</option></select><br><input type="button" value="change
       " onclick="changeProps();"></div>
194  <div id="shapes">Shapes:<br><input type="button" id="add" value="
       add" onclick="add()"><br></div>
195  <div class="controller"><input type="button" id="left" value="&
       larr;" onclick="move(-10,0);"><input type="button" id="up"
       value="&uarr;" onclick="move(0,-10);"><input type="button" id=
       "down" value="&darr;" onclick="move(0,10);"><input type="
       button" id="right" value="&rarr;" onclick="move(10,0);"></div>
196  <div id="props">Properties:<br></div>
197  <div id="styles">Styles:<br><span>fill</span><input type="color"
       id="fill" value="#FF0000"><br><span>stroke</span><input type="
       color" id="stroke" value="#000000"><br><span>stroke-width</
       span><input type="number" id="stroke-width" value="2"><br>
198  <span>fill-opacity</span><input type="number" id="fill-opacity"
       value="1"><br> <span>stroke-opacity</span><input type="number"
        id="stroke-opacity" value="1"><br></div>
199 </body>
200 </html>
```

Listing A4: HTML code of DrawDomExample.html.

```
<!DOCTYPE html>
<html>
<head>
<script>
function loadNews(url, divId) {
  var xmlhttp=new XMLHttpRequest();
  xmlhttp.onreadystatechange=function() {
    if (xmlhttp.readyState==4 && xmlhttp.status==200) {
      document.getElementById(divId).innerHTML=xmlhttp.
          responseText;
    }
  }
  xmlhttp.open("GET",url,true);
  xmlhttp.send();
}

function news() {
  loadNews('http://microsoft-news.com/','microsoftNews');
  loadNews('http://news.yahoo.com/','yahooNews');
}
function count() {
  setInterval(function() {
    document.getElementById("time").innerHTML = Number(document.
        getElementById("time").innerHTML) + 1
  },1000);
}
</script>
</head>
<body onload="count();">
<h1>Loading the latest news with AJAX</h1>
<div id="microsoftNews" style="height: 400px; overflow-y: scroll;">
    <h2>Microsoft News</h2></div>
<div id="yahooNews" style="height: 400px; overflow-y: scroll;"><h2>
    Let AJAX change this text</h2></div>
<button type="button" onclick="news();">load news</button>
<div>Time you visited this site in seconds: <span id="time">0</
    span></div>

</body>
</html>
```

Listing A5: HTML code of AJAX.html.

```
1 /*************************************************************
      ********************
2  * Copyright 2011 Google Inc. All Rights Reserved.
3  *
4  * All rights reserved. This program and the accompanying
       materials
5  * are made available under the terms of the Eclipse Public
       License v1.0
6  * which accompanies this distribution, and is available at
7  * http://www.eclipse.org/legal/epl-v10.html
8  *
9  * Unless required by applicable law or agreed to in writing,
       software
10 * distributed under the License is distributed on an "AS IS"
       BASIS,
11 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
       implied.
12 * See the License for the specific language governing permissions
       and
13 * limitations under the License.
14 *************************************************************
      ********************/
15 package de.tu_freiberg.informatik.vonwenckstern.client;
16
17 import com.google.gwt.core.client.EntryPoint;
18 import com.google.gwt.event.dom.client.ClickEvent;
19 import com.google.gwt.event.dom.client.ClickHandler;
20 import com.google.gwt.user.client.Window;
21 import com.google.gwt.user.client.ui.Button;
22 import com.google.gwt.user.client.ui.RootPanel;
23
24 /**
25  * Entry point classes define <code>onModuleLoad()</code>.
26  */
27 public class ImageViewer implements EntryPoint {
28   private Button clickMeButton;
29   public void onModuleLoad() {
30     RootPanel rootPanel = RootPanel.get();
31
32     clickMeButton = new Button();
33     rootPanel.add(clickMeButton);
34     clickMeButton.setText("Click␣me!");
35     clickMeButton.addClickHandler(new ClickHandler(){
36       public void onClick(ClickEvent event) {
37         Window.alert("Hello,␣GWT␣World!");
38       }
39     });
40   }
41 }
```

Listing A6: Auto generated example source code by Eclipse

```
  $intern_64 = '.cache.html',
  $intern_57 = '67FDEAF3397887CF1E2008A6BC06B53D',
  $intern_58 = '918B3274E8A61CA938ACB8AB0CB1EAF3',
  $intern_59 = 'D2314070AA3C3D47EA100703B721615D',
  $intern_60 = 'D86C78425EBF3471F02A10176A0C7644',
  $intern_61 = 'E2D72B049FFE71FEB2C8EA0340FE7E6F',
  $intern_62 = 'FA63816BAE5949A61910915A3B26F68A'

  $intern_41 = 'opera',
  $intern_46 = 'safari',
  $intern_48 = 'ie9',
  $intern_52 = 'gecko1_8',
  $intern_49 = 'ie8',
  $intern_50 = 'ie6'
```

Listing A7: Code from de.tu_freiberg.informatik.vonwenckstern.ImageViewer.nocache.js at line 2 (excerption)

```
313 if (!isHostedMode()) {
314     try {
315       unflattenKeylistIntoAnswers([$intern_41], $intern_57);
316       unflattenKeylistIntoAnswers([$intern_46], $intern_58);
317       unflattenKeylistIntoAnswers([$intern_48], $intern_59);
318       unflattenKeylistIntoAnswers([$intern_52], $intern_60);
319       unflattenKeylistIntoAnswers([$intern_49], $intern_61);
320       unflattenKeylistIntoAnswers([$intern_50], $intern_62);
```

Listing A8: Code from de.tu_freiberg.informatik.vonwenckstern.ImageViewer.nocache.js

```
1  <html><head>
2  <script language="JavaScript">
3  function addWithOut(a,b) {
4    var c = a + b;
5    alert('Result without type conversion is: ' + c);
6  }
7  function addWith(a,b) {
8    var c = Number(a) + Number(b);
9    alert('Result with type conversion is: ' + c);
10 }
11 </script></head><body><form name="calc" action="">
12 Zahl 1: <input type="text" name="nb1"><br>
13 Zahl 1: <input type="text" name="nb2"><br>
14 <input type="button" value="add␣without␣conversion" onClick="
       javascript:addWithOut(document.calc.nb1.value,␣document.calc.
       nb2.value)">
15 <input type="button" value="add␣with␣conversion" onClick="
       javascript:addWith(document.calc.nb1.value,␣document.calc.nb2.
       value)">
16 </form></body></html>
```

Listing A9: Code showing typing conversion in JavaScript

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import com.google.gwt.core.client.EntryPoint;
4  import com.google.gwt.event.dom.client.ClickEvent;
5  import com.google.gwt.event.dom.client.ClickHandler;
6  import com.google.gwt.user.client.Window;
7  import com.google.gwt.user.client.ui.Button;
8  import com.google.gwt.user.client.ui.Label;
9  import com.google.gwt.user.client.ui.RootPanel;
10
11 class Person {
12
13   public Person(String name, int age, boolean isSingle,
14   boolean isMale) {
15     this.name = name; this.age = age;
16     this.isSingle = isSingle; this.isMale = isMale;
17   }
18
19   public String name = "";
20   public int age = 0;
21   public boolean isSingle = true;
22   public boolean isMale = true;
23   public String note = "";
24 }
25
26 public class ImageViewer implements EntryPoint {
27
28   // define static variable
29   static boolean printSingles = false;
30
31   // define local variable
32   boolean printMales = false;
33
34   /** this function shows how to read, write to variables and
       return a String */
35   public native String printStatus(Person p1, Person p2, Person p3
       )
36   /*-{
37     var msg = '';
38     // access to static variable printSingles
39     var single = @de.tu_freiberg.informatik.vonwenckstern.client.
         ImageViewer::printSingles;
40     // access to local variable printMales
41     var male = this.@de.tu_freiberg.informatik.vonwenckstern.
         client.ImageViewer::printMales;
42
43     if(single) {
44       msg = 'The following people are single and ';
45     } else {
46       msg = 'The following people are not single anymore and ';
47     }
48
49     if(male) {
50       msg = msg + 'male: ';
51     } else {
```

```
52        msg = msg + 'female: ';
53      }
54
55      if(p1.@de.tu_freiberg.informatik.vonwenckstern.client.Person::
            isSingle == single &&
56        p1.@de.tu_freiberg.informatik.vonwenckstern.client.Person::
            isMale == male) {
57        // access to variables of Object p1
58        msg = msg +
59            p1.@de.tu_freiberg.informatik.vonwenckstern.client.
                Person::name +
60            '(' +
61            p1.@de.tu_freiberg.informatik.vonwenckstern.client.
                Person::age +
62            ') ; ';
63      }
64
65      if(p2.@de.tu_freiberg.informatik.vonwenckstern.client.Person::
            isSingle == single &&
66        p2.@de.tu_freiberg.informatik.vonwenckstern.client.Person::
            isMale == male) {
67        // access to variables of Object p2
68        msg = msg +
69          p2.@de.tu_freiberg.informatik.vonwenckstern.client.Person
                ::name +
70          '(' + p2.@de.tu_freiberg.informatik.vonwenckstern.client.
                Person::age +
71          ') ; ';
72      }
73
74      if(p3.@de.tu_freiberg.informatik.vonwenckstern.client.Person::
            isSingle == single &&
75        p3.@de.tu_freiberg.informatik.vonwenckstern.client.Person::
            isMale == male) {
76        // access to variables of Object p2
77        msg = msg + p3.@de.tu_freiberg.informatik.vonwenckstern.
            client.Person::name +
78          '(' + p3.@de.tu_freiberg.informatik.vonwenckstern.client.
                Person::age +
79          ') ; ';
80      }
81
82      window.alert(msg);
83
84      // write to variable note in object p1
85      p1.@de.tu_freiberg.informatik.vonwenckstern.client.Person::
            note = 'You were person p1';
86      p2.@de.tu_freiberg.informatik.vonwenckstern.client.Person::
            note = 'You were person p2';
87      p3.@de.tu_freiberg.informatik.vonwenckstern.client.Person::
            note = 'You were person p3';
88
89      // now we want to call a java function to change the button,
90      // the type signature Ljava/lang/String;II means that the
            first parameter of changeButton has Java type String,
91      // second and third one have the type Integer
```

```
92      this.@de.tu_freiberg.informatik.vonwenckstern.client.
            ImageViewer::changeButton(Ljava/lang/String;II)
93      ('now we changed the button',500,200);
94
95      // return the string that we are done with JSNI
96      return 'We are done with JSNI, finally!';
97    }-*/;
98
99    public void changeButton(String text, int width, int height) {
100     clickMeButton.setText(text);
101     clickMeButton.setPixelSize(width, height);
102   }
103
104   private Button clickMeButton;
105   private Label label;
106   public void onModuleLoad() {
107      RootPanel rootPanel = RootPanel.get();
108
109      clickMeButton = new Button();
110      label = new Label();
111      rootPanel.add(clickMeButton);
112      rootPanel.add(label);
113      clickMeButton.setText("Click␣to␣call␣JSNI␣function␣
             printStatus!");
114      clickMeButton.addClickHandler(new ClickHandler(){
115       public void onClick(ClickEvent event) {
116         Person michael = new Person("Michael␣von␣Wencksstern", 24,
                false, true);
117         Person julius = new Person("Julius␣Austin", 40, true, true
                );
118         Person anna = new Person("Anna␣Weiss", 17, false, false);
119         // read out Michael's note before printStatus was called
120         label.setText("Michael's␣note␣is:␣" + michael.note);
121         String ret = printStatus(michael, julius, anna);
122         // read out the text which printStatus set
123         label.setText("printStatus␣returns:␣" + ret + "\nMichael's
               ␣note␣is:␣" + michael.note);
124       }
125      });
126   }
127 }
```

Listing A10: Example code showing how to access a Java variable, write to a Java variable and return a value in JavaScript using JSNI (see figure A31 for the screenshot and listing A11 for the compiler output of the JSNI function printStatus)

```
function uf(g,a,b,c){
  var d=vi;
  var e=sf;
  var f=g.d;e?(d='The following people are single and '):(d='The
     following people are not single anymore and ');
  f?(d=d+'male: '):(d=d+'female: ');
  a.d==e&&a.c==f&&(d=d+a.e+wi+a.b+_i);
  b.d==e&&b.c==f&&(d=d+b.e+wi+b.b+_i);
  c.d==e&&c.c==f&&(d=d+c.e+wi+c.b+_i);
  window.alert(d);
  a.f='You were person p1';
  b.f='You were person p2';
  c.f='You were person p3';
  g.y('now we changed the button',500,200);
  return 'We are done with JSNI, finally!'
}
```

Listing A11: This is the code generated by the GWT compiler in obfuscated mode of the JSNI function

```
1  <!doctype html>
2  <html>
3    <head>
4      <meta http-equiv="content-type" content="text/html; charset=
           UTF-8">
5
6
7      <link type="text/css" rel="stylesheet" href="ImageViewer.css">
8
9      <script type="text/javascript" src="syntaxhighlighter_3.0.83/
           scripts/shCore.js"></script>
10     <script type="text/javascript" src="syntaxhighlighter_3.0.83/
           scripts/shBrushJScript.js"></script>
11     <script type="text/javascript" src="syntaxhighlighter_3.0.83/
           scripts/shBrushJava.js"></script>
12     <script type="text/javascript" src="syntaxhighlighter_3.0.83/
           scripts/shBrushSql.js"></script>
13     <link type="text/css" rel="stylesheet" href="
           syntaxhighlighter_3.0.83/styles/shCoreDefault.css"/>
14
15     <title>Wrapper HTML for ImageViewer</title>
16
17     <!--                                                  -->
18     <!-- This script loads your compiled module.          -->
19     <!-- If you add any GWT meta tags, they must          -->
20     <!-- be added before this line.                       -->
21     <!--                                                  -->
22     <script language="javascript" src="de.tu_freiberg.informatik.
           vonwenckstern.ImageViewer/de.tu_freiberg.informatik.
           vonwenckstern.ImageViewer.nocache.js"></script>
23
24     <script type="text/javascript">SyntaxHighlighter.all();</
           script>
25   </head>
26
27
28   <body>
29     <!-- OPTIONAL: include this if you want history support -->
30     <iframe id="__gwt_historyFrame" style="width:0;height:0;border
           :0"></iframe>
31
32   </body>
33 </html>
```

Listing A12: ImageViewer.html in LoadJavaScript library's project

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import com.google.gwt.core.client.JavaScriptObject;
4  import com.google.gwt.dom.client.Element;
5
6  public class SyntaxHighlighter extends JavaScriptObject {
7    // constructor of JavaScriptObject must be protected
8    protected SyntaxHighlighter() {}
9
10   // reference to the implementation class
11   private static SyntaxHighlighterImpl impl = new
         SyntaxHighlighterImpl();
12
13   public static SyntaxHighlighter create(Element el) {
14     return impl.create(el);
15   }
16
17   // Instance methods must be 'final' in non-final subclasses of
         JavaScriptObject
18   public final void setLanguage(String language) {
19     impl.setLanguage(this, language);
20   }
21
22   public final void setCode(String code) {
23     impl.setCode(this, code);
24   }
25 }
```

Listing A13: SyntaxHighlighter.java in LoadJavaScript library's project

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import com.google.gwt.dom.client.Element;
4  import com.google.gwt.user.client.ui.HTML;
5
6  public class SyntaxHighlighterWidget extends HTML {
7
8    private SyntaxHighlighter sh = null;
9
10   public SyntaxHighlighter getSyntaxHighlighter(Element el) {
11     if(sh == null) {
12       sh = SyntaxHighlighter.create(el);
13     }
14     return sh;
15   }
16
17   public SyntaxHighlighterWidget() {
18     getSyntaxHighlighter(getElement());
19   }
20
21   public static final String ActionScript3 = "as3";
22   public static final String Shell = "shell";
23   public static final String ColdFusion = "cf";
24   public static final String C_Sharp = "c-sharp";
25   public static final String C_Plus_Plus = "cpp";
26   public static final String CSS = "css";
27   public static final String Delphi = "delphi";
28   public static final String Diff = "diff";
29   public static final String Erlang = "erl";
30   public static final String Groovy = "groovy";
31   public static final String JavaScript = "js";
32   public static final String Java = "java";
33   public static final String JavaFX = "jfx";
34   public static final String Perl = "perl";
35   public static final String PHP = "php";
36   public static final String Plain_Text = "plain";
37   public static final String PowerShell = "ps";
38   public static final String Python = "py";
39   public static final String Ruby = "rails";
40   public static final String Scala = "scala";
41   public static final String SQL = "sql";
42   public static final String Visual_Basic = "vb";
43   public static final String XML = "xml";
44   public void setLanguage(String language) {
45     sh.setLanguage(language);
46   }
47
48   public void setCode(String code) {
49     sh.setCode(code);
50   }
51 }
```

Listing A14: SyntaxHighlighterWidget.java in LoadJavaScript library's project

```
1  /**...
2   * Copyright 2011 Google Inc. All Rights Reserved.
3   *
4   * All rights reserved. This program and the accompanying
      materials
5   * are made available under the terms of the Eclipse Public
      License v1.0
6   * which accompanies this distribution, and is available at
7   * http://www.eclipse.org/legal/epl-v10.html
8   *
9   * Unless required by applicable law or agreed to in writing,
      software
10  * distributed under the License is distributed on an "AS IS"
      BASIS,
11  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
      implied.
12  * See the License for the specific language governing permissions
      and
13  * limitations under the License.
14  *** ... **/
15 package de.tu_freiberg.informatik.vonwenckstern.client;
16
17 import com.google.gwt.core.client.EntryPoint;
18 import com.google.gwt.core.client.JavaScriptObject;
19 import com.google.gwt.dom.client.Element;
20 import com.google.gwt.event.dom.client.ClickEvent;
21 import com.google.gwt.event.dom.client.ClickHandler;
22 import com.google.gwt.user.client.Window;
23 import com.google.gwt.user.client.ui.Button;
24 import com.google.gwt.user.client.ui.Composite;
25 import com.google.gwt.user.client.ui.FlexTable;
26 import com.google.gwt.user.client.ui.HTML;
27 import com.google.gwt.user.client.ui.RootPanel;
28
29 /**
30  * Entry point classes define <code>onModuleLoad()</code>.
31  */
32 public class ImageViewer implements EntryPoint {
33
34   public void onModuleLoad() {
35     RootPanel rootPanel = RootPanel.get();
36     FlexTable table = new FlexTable();
37     rootPanel.add(table);
38     SyntaxHighlighterWidget shJavaScript = new
         SyntaxHighlighterWidget();
39     table.setText(0, 0, "JavaScript");
40     table.setWidget(0, 1, shJavaScript);
41     shJavaScript.setLanguage(SyntaxHighlighterWidget.JavaScript);
42     shJavaScript.setCode("function␣helloSyntaxHighlighter()␣{\n\
         treturn␣\"hi!\";␣//␣returning␣'hi'\n}\n");
43
44     SyntaxHighlighterWidget shJava = new SyntaxHighlighterWidget()
         ;
45     table.setText(0, 2, "Java");
46     table.setWidget(0, 3, shJava);
```

```
47      shJava.setLanguage(SyntaxHighlighterWidget.Java);
48      shJava.setCode("public␣int␣add(int␣a,␣int␣b)␣{\n\tint␣ret␣=␣a␣
            +␣b;␣//␣adding␣a␣and␣b\n\treturn␣ret;\n}");
49
50      SyntaxHighlighterWidget shSQL = new SyntaxHighlighterWidget();
51      table.setText(1, 0, "SQL");
52      table.setWidget(1, 1, shSQL);
53      table.getFlexCellFormatter().setColSpan(1, 1, 3);
54      shSQL.setLanguage(SyntaxHighlighterWidget.SQL);
55      shSQL.setCode("SELECT␣`id`␣FROM␣`cols`␣WHERE␣`deleted`=␣'0'␣
            ORDER␣BY␣`order`␣ASC");
56    }
57  }
```

Listing A15: ImageViewer.java in LoadJavaScript library's project

```
36  public class ImageViewer implements EntryPoint {
37    public void onModuleLoad() {
38      RootPanel rootPanel = RootPanel.get();
39      // if you want to use deferred binding, you have to use GWT.
            create
40      // and not only status = new Status();
41      Status status = GWT.create(Status.class);
42      rootPanel.add(status.asWidget());
43      status.getStatus();
44    }
45  }
```

Listing A16: ImageViewer.java in deferred binding with replacement project (license comment omitted)

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import com.google.gwt.user.client.ui.Label;
4
5  public class StatusLabel extends Label implements Status {
6    @Override
7    public void getStatus() {
8      this.setText("you␣are␣not␣using␣Firefox␣or␣Internet␣Explorer␣9
           ");
9    }
10 }
```

Listing A17: StatusLabel.java in deferred binding with replacement project

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3  import java.io.PrintWriter;
4
5  import com.google.gwt.core.ext.BadPropertyValueException;
6  import com.google.gwt.core.ext.Generator;
7  import com.google.gwt.core.ext.GeneratorContext;
8  import com.google.gwt.core.ext.TreeLogger;
9  import com.google.gwt.core.ext.UnableToCompleteException;
10 import com.google.gwt.core.ext.typeinfo.JClassType;
11 import com.google.gwt.core.ext.typeinfo.NotFoundException;
12 import com.google.gwt.user.rebind.ClassSourceFileComposerFactory;
13 import com.google.gwt.user.rebind.SourceWriter;
14
15 public class StatusGenerator extends Generator {
16
17   @Override
18   /** function will build a new class that will implement
         getStatus() method
19   * @return name of the generated class in the given package
         structure,
20   * so that the compiler can use the generated class
21   * */
22   public String generate(TreeLogger logger, GeneratorContext
         context,
23     String typeName) throws UnableToCompleteException {
24       String userAgent = "";
25       try {
26         // getting the user.agent property for generating this
             file
27         userAgent = context.getPropertyOracle().
             getSelectionProperty(logger, "user.agent").
             getCurrentValue();
28       } catch (BadPropertyValueException e) {
29         e.printStackTrace();
30         return null;
31       }
32       try {
33         SourceWriter sw = getSourceWriter(typeName, context,
             logger, userAgent);
34         assert(sw != null);
35         String browser= getBrowser(userAgent);
36
37         // after we have all the information we need, we write our
              function in the already created class
38         sw.println("@Override");
39         sw.println("public␣void␣getStatus()␣{");
40         sw.println("this.setText(\"You␣are␣using␣" + browser + "
             .\");");
41         sw.println("}");
42
43         // if you forget it then the compiler cannot find the
             generated classes and you get
44         // errors like: Rebind result 'de.tu_freiberg.informatik.
             vonwenckstern.client.StatussafariGenerated' could not
```

```
            be found
45          sw.commit(logger);
46          System.out.println("class '" + typeName + userAgent + "
                Generated' was created succesfully");
47          return typeName + userAgent + "Generated";
48        } catch(NotFoundException e) {
49          e.printStackTrace();
50          return null;
51        }
52      }
53
54      /** function which will return a readable browser version for
            the given userAgent*/
55      public String getBrowser(String userAgent) {
56        String browser = userAgent;
57
58        if(browser.equalsIgnoreCase("gecko") || browser.
              equalsIgnoreCase("gecko1_8")) {
59          browser = "Firefox";
60        } else if(browser.startsWith("ie")) {
61          browser = "Internet Explorer" + browser.substring(2);
62        } else if(browser.equalsIgnoreCase("safari")) {
63          browser = "Safari or Chrome";
64        } else if(browser.equalsIgnoreCase("opera")) {
65          browser = "Opera";
66        } else {
67          browser = "unknown browser";
68        }
69        return browser;
70      }
71
72      /** function returns the source writer where you can add the
            inner classes functions*/
73      public SourceWriter getSourceWriter(String typeName,
            GeneratorContext context,
74        TreeLogger logger, String userAgent) throws
              NotFoundException {
75          // gets the type given by the String typeName
76          JClassType classType = context.getTypeOracle().getType(
                typeName);
77          // gets the package in which the new class should get
                created
78          String packageName = classType.getPackage().getName();
79          // gets the name of the class without the package name
80          String simpleName = classType.getSimpleSourceName();
81          // for us to see what classes were generated by this
                generator
82          simpleName = simpleName + userAgent + "Generated";
83          // a composer factory which will create a new class in the
                given package with the given name
84          ClassSourceFileComposerFactory composer = new
                ClassSourceFileComposerFactory(packageName, simpleName)
                ;
85          // the class which we extend
86          composer.setSuperclass("com.google.gwt.user.client.ui.
                Label");
```

```
87          // now we are adding the implemented interface Status
88          composer.addImplementedInterface("de.tu_freiberg.
                informatik.vonwenckstern.client.Status");
89          // now we are adding the imports we need
90          composer.addImport("com.google.gwt.user.client.ui.Label");
91          // creates the source file in the given package with the
                given name
92          PrintWriter printWriter = context.tryCreate(logger,
                packageName, simpleName);
93          if(printWriter == null) {
94            System.out.println("printWriter is null");
95            return null;
96          } else {
97            // will create the class with all the given imports,
                  extends and so and will write
98            // it to the source file created by printWriter
99            SourceWriter sw=composer.createSourceWriter(context,
                  printWriter);
100           return sw;
101         }
102
103     }
104 }
```

Listing A18: StatusGenerator.java in deferred binding with generator project (code ideas from [Sha09] and [Gooa])

```
1 Compiling module de.tu_freiberg.informatik.vonwenckstern.
    ImageViewer
2    Validating newly compiled units
3 Compile with -strict or with -logLevel set to TRACE or DEBUG to
    see all errors.
4 class 'de.tu_freiberg.informatik.vonwenckstern.client.
    Statusgecko1_8Generated' was created succesfully
5 class 'de.tu_freiberg.informatik.vonwenckstern.client.
    Statusie6Generated' was created succesfully
6 class 'de.tu_freiberg.informatik.vonwenckstern.client.
    Statusie8Generated' was created succesfully
7 class 'de.tu_freiberg.informatik.vonwenckstern.client.
    Statusie9Generated' was created succesfully
8 class 'de.tu_freiberg.informatik.vonwenckstern.client.
    StatusoperaGenerated' was created succesfully
9 class 'de.tu_freiberg.informatik.vonwenckstern.client.
    StatussafariGenerated' was created succesfully
10    Compiling 6 permutations
11       Compiling permutation 0...
12       Compiling permutation 1...
13       Compiling permutation 2...
14       Compiling permutation 3...
15       Compiling permutation 4...
16       Compiling permutation 5...
17    Compile of permutations succeeded
18 Linking into C:\GWT\workspace\DA_DeferredBinding_Generator\war\de.
    tu_freiberg.informatik.vonwenckstern.ImageViewer
19    Link succeeded
20    Compilation succeeded -- 10,406s
```

Listing A19: Compiler output in deferred binding with generator project

```
1 package de.tu_freiberg.informatik.vonwenckstern.client;
2 import com.google.gwt.core.client.EntryPoint;
3 import com.google.gwt.user.client.Window;
4
5 public class ImageViewer implements EntryPoint {
6         /** entry point*/
7         public void onModuleLoad() {
8                 int i = 2 + 3 + 4;
9                 int j = 2*i;
10                String s = i + "␣;␣" + j;
11                Window.alert(s);
12        }
13 }
```

Listing A20: ImageViewer.java in compiler optimization project

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3  class Shape {
4    public String meeting(Shape s) { return "Shape meets Shape"; }
5  }
6
7  class Polygon extends Shape {
8    @Override
9    public String meeting(Shape s) { return "Polygon meets Shape";
       }
10   public String meeting(Polygon p) { return "Polygon meets Polygon
       "; }
11   public String meeting(Quadrilateral q) { return "Polygon meets
       Quadrilateral"; }
12   public String meeting(Square s) { return "Polygon meets Square";
        }
13   public String meeting(Triangle t) { return "Polygon meets
       Triangle"; }
14   public String meeting(Rectangle r) { return "Polygon meets
       Rectangle"; }
15   public String meeting(Parallelogram p) { return "Polygon meets
       Parallelogram"; }
16   public String meeting(Pentagon p) { return "Polygon meets
       Pentagon"; }
17   public String meeting(Hexagon h) { return "Polygon meets Hexagon
       "; }
18   public String meeting(Heptagon h) { return "Polygon meets
       Heptagon"; }
19   public String meeting(Octagon o) { return "Polygon meets Octagon
       "; }
20 }
142 class Octagon extends Shape {
143   @Override
144   public String meeting(Shape s) { return "Octagon meets Shape";
       }
145   public String meeting(Polygon p) { return "Octagon meets Polygon
       "; }
146   public String meeting(Quadrilateral q) { return "Octagon meets
       Quadrilateral"; }
147   public String meeting(Square s) { return "Octagon meets Square";
        }
148   public String meeting(Triangle t) { return "Octagon meets
       Triangle"; }
149   public String meeting(Rectangle r) { return "Octagon meets
       Rectangle"; }
150   public String meeting(Parallelogram p) { return "Octagon meets
       Parallelogram"; }
151   public String meeting(Pentagon p) { return "Octagon meets
       Pentagon"; }
152   public String meeting(Hexagon h) { return "Octagon meets Hexagon
       "; }
153   public String meeting(Heptagon h) { return "Octagon meets
       Heptagon"; }
154   public String meeting(Octagon o) { return "Octagon meets Octagon
       "; }
```

```
155 }
156
157 class TriangleDoubleDispatchEmulator extends Triangle {
158   @Override
159   public String meeting(Shape s) {
160     if(s instanceof Polygon) return super.meeting((Polygon)s);
161     if(s instanceof Quadrilateral) return super.meeting((
          Quadrilateral)s);
162     if(s instanceof Square) return super.meeting((Square)s);
163     if(s instanceof Triangle) return super.meeting((Triangle)s);
164     if(s instanceof Rectangle) return super.meeting((Rectangle)s);
165     if(s instanceof Parallelogram) return super.meeting((
          Parallelogram)s);
166     if(s instanceof Pentagon) return super.meeting((Pentagon)s);
167     if(s instanceof Hexagon) return super.meeting((Hexagon)s);
168     if(s instanceof Heptagon) return super.meeting((Heptagon)s);
169     if(s instanceof Octagon) return super.meeting((Octagon)s);
170     return super.meeting(s);
171   }
172 }
173
174 public class Main {
175
176   public static void main(String[] args) {
177     Shape shape1 = new Triangle();
178     Shape shape2 = new Heptagon();
179     // supports only single dispatch, it is calling Triangle::
          meeting(Shape)
180     //    --> only the object on which the method is invoked will
          be dispatched at runtime
181     // no dispatch would be calling Shape::meeting(Shape)
182     // double dispatch would be calling Triangle::meeting(Heptagon
          ),
183     //    --> all two object would be dispatched at runtime
184     System.out.println("Java␣only␣supports␣single␣dispatch:");
185     System.out.println(shape1.meeting(shape2));
186
187     shape1 = new TriangleDoubleDispatchEmulator();
188     System.out.println();
189     System.out.println("TriangleDoubleDispatchEmulator␣emulates␣
          double␣dispatch:");
190     System.out.println(shape1.meeting(shape2));
191   }
192 }
```

```
Java only supports single dispatch:
Triangle meets Shape

TriangleDoubleDispatchEmulator emulates double dispatch:
Triangle meets Heptagon
```

Listing A21: Main.java (top code with line numbers) and console output (bottom display wihtout line numbers) in dispatch project (shape example).

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3  interface Visitable {
4    public String accept(ShapeVisitor v);
5  }
6
7  class ShapeVisitor implements Visitable {
8    public String visit(ShapeVisitor s) { return "Shape visits Shape
       "; }
9    public String visit(Polygon p) { return "Shape visits Polygon";
       }
10   public String visit(Quadrilateral q) { return "Shape visits
       Quadrilateral"; }
11   public String visit(Square s) { return "Shape visits Square"; }
12   public String visit(Triangle t) { return "Shape visits Triangle"
       ; }
13   public String visit(Rectangle r) { return "Shape visits
       Rectangle"; }
14   public String visit(Parallelogram p) { return "Shape visits
       Parallelogram"; }
15   public String visit(Pentagon p) { return "Shape visits Pentagon"
       ; }
16   public String visit(Hexagon h) { return "Shape visits Hexagon";
       }
17   public String visit(Heptagon h) { return "Shape visits Heptagon"
       ; }
18   public String visit(Octagon o) { return "Shape visits Octagon";
       }
19   @Override
20   public String accept(ShapeVisitor visitor) {
21     return visitor.visit(this);
22   }
23 }
24
25 class Polygon extends ShapeVisitor {
26   @Override
27   public String visit(ShapeVisitor s) { return "Polygon visits
       Shape"; }
28   public String visit(Polygon p) { return "Polygon visits Polygon"
       ; }
29   public String visit(Quadrilateral q) { return "Polygon visits
       Quadrilateral"; }
30   public String visit(Square s) { return "Polygon visits Square";
       }
31   public String visit(Triangle t) { return "Polygon visits
       Triangle"; }
32   public String visit(Rectangle r) { return "Polygon visits
       Rectangle"; }
33   public String visit(Parallelogram p) { return "Polygon visits
       Parallelogram"; }
34   public String visit(Pentagon p) { return "Polygon visits
       Pentagon"; }
35   public String visit(Hexagon h) { return "Polygon visits Hexagon"
       ; }
36   public String visit(Heptagon h) { return "Polygon visits
```

```
      Heptagon"; }
37    public String visit(Octagon o) { return "Polygon visits Octagon"
        ; }
38    @Override
39    public String accept(ShapeVisitor visitor) {
40      return visitor.visit(this);
41    }
42  }
196 class Octagon extends ShapeVisitor {
197   @Override
198   public String visit(ShapeVisitor s) { return "Octagon visits
        Shape"; }
199   public String visit(Polygon p) { return "Octagon visits Polygon"
        ; }
200   public String visit(Quadrilateral q) { return "Octagon visits
        Quadrilateral"; }
201   public String visit(Square s) { return "Octagon visits Square";
        }
202   public String visit(Triangle t) { return "Octagon visits
        Triangle"; }
203   public String visit(Rectangle r) { return "Octagon visits
        Rectangle"; }
204   public String visit(Parallelogram p) { return "Octagon visits
        Parallelogram"; }
205   public String visit(Pentagon p) { return "Octagon visits
        Pentagon"; }
206   public String visit(Hexagon h) { return "Octagon visits Hexagon"
        ; }
207   public String visit(Heptagon h) { return "Octagon visits
        Heptagon"; }
208   public String visit(Octagon o) { return "Octagon visits Octagon"
        ; }
209   @Override
210   public String accept(ShapeVisitor visitor) {
211     return visitor.visit(this);
212   }
213 }
214
215 public class Main {
216
217   public static void main(String[] args) {
218     ShapeVisitor shape1 = new Triangle();
219     ShapeVisitor shape2 = new Heptagon();
220
221     // double dispatch is achieved by resolving twice single
          dispatch
222     // shape1.accept --> Triangle::accept(ShapeVisitor)
223     //     ==> visitor.visit(this) --> Heptagon::visit(Triangle)
224     System.out.println("Visitor pattern:");
225     System.out.println(shape1.accept(shape2));
226
227
228   }
229 }
```

```
Visitor pattern:
Heptagon visits Triangle
```

Listing A22: Main.java (top code with line numbers) and console output (bottom display without line numbers) in visitor pattern project (shape example).

```
org.eclipse.core.contenttype_3.4.100.v20110423-0524.jar
org.eclipse.core.jobs_3.5.101.v20120113-1953.jar
org.eclipse.core.resources_3.7.101.v20120125-1505.jar
org.eclipse.core.runtime_3.7.0.v20110110.jar
org.eclipse.equinox.common_3.6.0.v20110523.jar
org.eclipse.equinox.preferences_3.4.2.v20120111-2020.jar
org.eclipse.jdt.core_3.7.3.v20120119-1537.jar
org.eclipse.osgi_3.7.2.v20120110-1415.jar
org.eclipse.text_3.5.101.v20110928-1504.jar
(available under ...\eclipse\plugins)
ui-3.2.1-r321_v20060907.jar
(available under http://repo1.maven.org/maven2/org/eclipse/jdt/ui
    /3.2.1-r321_v20060907/ui-3.2.1-r321_v20060907.jar)
.
```

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3  import java.io.File;
4  import java.io.FileInputStream;
5  import java.io.FileOutputStream;
6  import java.io.IOException;
7
8  import org.eclipse.jdt.core.dom.AST;
9  import org.eclipse.jdt.core.dom.ASTNode;
10 import org.eclipse.jdt.core.dom.ASTParser;
11 import org.eclipse.jface.text.BadLocationException;
12 import org.eclipse.text.edits.MalformedTreeException;
13
14 public class Main {
15
16   /** returning source code of the given class
17    * @param className the full class name inclusive package name
18    * */
19   public static String readClassFile(String className) throws
        IOException {
20     String s = new File(".").getAbsolutePath(); // get the running
           directory (we assume you are running this class out of
         Eclipse)
21     s = s.substring(0, s.length() - 1); // remove the . at the end
22     s = s + "src\\" + className.replace(".", "\\") + ".java";
23
24     FileInputStream fis = new FileInputStream(s);
25     byte[] content = new byte[fis.available()];
26     fis.read(content);
27     return new String(content, "UTF-8");
28   }
29
30   /** creating the given java file and writes the content into it
         */
31   public static void writeClassFile(String content, String
        className) throws IOException {
32     String s = new File(".").getAbsolutePath(); // get the running
           directory (we assume you are running this class out of
         Eclipse)
33     s = s.substring(0, s.length() - 1); // remove the . at the end
34     s = s + "src\\" + className.replace(".", "\\") + ".java";
35
36     FileOutputStream fos = new FileOutputStream(s);
37     fos.write(content.getBytes("UTF-8"));
38   }
```

```
39
40   public static void main(String args[]) throws
        MalformedTreeException, BadLocationException{
41     String content = "";
42     try {
43       content = readClassFile("de.tu_freiberg.informatik.
           vonwenckstern.TestClass");
44     } catch(IOException e) {
45       e.printStackTrace();
46       return;
47     }
48
49     /*
50      * getting the abstract source tree of the loaded Java class
51      */
52     ASTParser parser = ASTParser.newParser(AST.JLS3);
53     parser.setKind(ASTParser.K_COMPILATION_UNIT);
54     parser.setSource(content.toCharArray()); // set source
55
56     // normally you would use bindings getting the
57     // complete function names and argument types,
58     // but it does not work with just given a simple file,
59     // for binding you need an entire Eclipse project
60     // parser.setResolveBindings(true);
61
62     ASTNode node = parser.createAST(null);
63
64     /*
65      * doing some operations with the Java AST
66      * the AST gets changed
67      */
68
69     ASTPrinter.exec(node);
70     ASTOptimizer.exec(node);
71     ASTRenamer.exec(node, "de.tu_freiberg.informatik.TestClass2");
72
73     /*
74      * converting the Java AST to normal Java code
75      * and writing it into a file
76      */
77    String s=org.eclipse.jdt.internal.corext.dom.ASTFlattener.
         asString(node);
78     try {
79       writeClassFile(s, "de.tu_freiberg.informatik.TestClass2");
80     } catch (IOException e) {
81       e.printStackTrace();
82     }
83   }
84 }
```

```
package name: de.tu_freiberg.informatik.vonwenckstern
You are using the following imports:
java.util.ArrayList     java.util.Date
You have declared the following types:
Empty(is interface)
  field declarations:
```

```
  method declarations:
TestClass(is class, public, final)
  field declarations: integer1(int)    integer2(int)    cityList
     (ArrayList<String>, private)    date(Date, private)
  method declarations: addingSomeCitiesToTheCityList(return type:
     void, modifiers: private, parameter types: )
     getCitiesOfList(return type: String, modifiers: public,
     parameter types: )    thisFunctionIsUseLess(return type:
     void, modifiers: private, parameter types: int, long, double,
      float)
'thisFunctionIsUseLess' is private and is not used in this class,
   will be deleted now.
'date' this field variable is private and is not used in this
   class, will be deleted now.
```

Listing A23: Included external JAR libraries(top listing without line numbers), Main.java (middle code with line numbers) and console output (bottom display wihtout line numbers) in AST project.
The listings A24, A25 and A26 belongs to this project, too.

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3  import java.util.List;
4
5  import org.eclipse.jdt.core.dom.ASTNode;
6  import org.eclipse.jdt.core.dom.ASTVisitor;
7  import org.eclipse.jdt.core.dom.CompilationUnit;
8  import org.eclipse.jdt.core.dom.SingleVariableDeclaration;
9  import org.eclipse.jdt.core.dom.VariableDeclarationFragment;
10
11
12 public class ASTPrinter extends ASTVisitor {
13
14   private boolean showedImport = false;
15   private boolean typeDecl = false;
16
17   /**
18   * prints some information of the abstract source tree to the
       command line
19   * @param node root node of ASTParser
20   */
21   public static void exec(ASTNode node) {
22     CompilationUnit cu = (CompilationUnit) node;
23     cu.accept(new ASTPrinter());
24   }
25
26   //////////////////////////////////// types of CompilationUnit
       ////////////////////////
27   public boolean visit(org.eclipse.jdt.core.dom.PackageDeclaration
        packageDecl) {
28     System.out.println("package␣name:␣" + packageDecl.getName());
29     return false;
30   }
31
32   public boolean visit(org.eclipse.jdt.core.dom.ImportDeclaration
       importDecl) {
```

```
33      if(!showedImport) {
34        System.out.println("You␣are␣using␣the␣following␣imports:␣");
35        showedImport = true;
36      }
37      System.out.print(importDecl.getName() + "␣␣␣␣␣");
38      return false;
39    }
40
41    public boolean visit(org.eclipse.jdt.core.dom.TypeDeclaration
          typeDeclaration) {
42      if(!typeDecl) {
43        System.out.println();
44        System.out.println("You␣have␣declared␣the␣following␣types:␣"
              );
45        typeDecl = true;
46      }
47      List modifiers = typeDeclaration.modifiers();
48      System.out.print(typeDeclaration.getName());
49      System.out.print( typeDeclaration.isInterface() ? "(is␣
            interface" : "(is␣class");
50      for(int i=0; i<modifiers.size(); i++) {
51        System.out.print(",␣" + modifiers.get(i));
52      }
53      System.out.println(")");
54
55      System.out.print("␣␣field␣declarations:␣");
56      typeDeclaration.accept(new FieldPrinter());
57      System.out.println();
58
59      System.out.print("␣␣method␣declarations:␣");
60      typeDeclaration.accept(new MethodPrinter());
61      System.out.println();
62      return false;
63    }
64    ////////////////////////////////// end types of
          CompilationUnit ///////////////////////
65
66    class FieldPrinter extends ASTVisitor {
67
68      /** displays the field variables of a class or interface */
69      public boolean visit(org.eclipse.jdt.core.dom.FieldDeclaration
            variableDecl) {
70        List<VariableDeclarationFragment> list = variableDecl.
              fragments();
71        for(VariableDeclarationFragment frag: list) {
72          System.out.print(frag.getName());
73
74          System.out.print("(" + variableDecl.getType().toString());
75          List modifiers = variableDecl.modifiers();
76          for(int i=0; i<modifiers.size(); i++) {
77            System.out.print(",␣" + modifiers.get(i));
78          }
79          System.out.print(")␣␣␣␣␣");
80        }
81
82        return false;
```

```
83      }
84
85    }
86
87    /** displays the declared functions of a class or interface */
88    class MethodPrinter extends ASTVisitor {
89      public boolean visit(org.eclipse.jdt.core.dom.
            MethodDeclaration methodDecl) {
90        System.out.print(methodDecl.getName());
91        System.out.print("(return␣type:␣" + methodDecl.
              getReturnType2());
92        System.out.print(",␣modifiers:␣");
93        List modifiers = methodDecl.modifiers();
94        for(int i=0; i<modifiers.size(); i++) {
95          if(i > 0) System.out.print(",␣");
96            System.out.print(modifiers.get(i));
97          }
98          System.out.print(",␣parameter␣types:␣");
99          List<SingleVariableDeclaration> parameters =methodDecl.
                parameters();
100         for(int i=0; i<parameters.size(); i++) {
101           if(i > 0) System.out.print(",␣");
102           System.out.print(parameters.get(i).getType());
103         }
104         System.out.print(")␣␣␣␣␣");
105         return false;
106       }
107     }
108 }
```

Listing A24: ASTPrinter.java in AST project

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  import org.eclipse.jdt.core.dom.ASTNode;
7  import org.eclipse.jdt.core.dom.ASTVisitor;
8  import org.eclipse.jdt.core.dom.CompilationUnit;
9  import org.eclipse.jdt.core.dom.Expression;
10 import org.eclipse.jdt.core.dom.IExtendedModifier;
11 import org.eclipse.jdt.core.dom.IMethodBinding;
12 import org.eclipse.jdt.core.dom.Modifier;
13 import org.eclipse.jdt.core.dom.SimpleName;
14 import org.eclipse.jdt.core.dom.VariableDeclarationFragment;
15
16 /** pseudo optimizes Java code by deleting not used private
       functions and field variables
17  *  this code is neither complete nor will it work for real Java
       files, it implements only
18  *  the visitor methods which are needed to "optimize" the example
        Java file TestClass.java
19  *  */
20 public class ASTOptimizer {
21   public static void exec(ASTNode node) {
22     new ASTOptimizer(node);
```

```
23   }
24
25   public ASTOptimizer(ASTNode node) {
26     CompilationUnit cu = (CompilationUnit) node;
27     cu.accept(new ASTVisitor() {
28       /** saving all invoked functions with its name in an array
             list */
29       public boolean visit(org.eclipse.jdt.core.dom.
             MethodInvocation methodInvocation) {
30         // to get methods solved by argument types you would need
              binding
31         // but it is not available for this example
32         // IMethodBinding binding = methodInvocation.
              resolveMethodBinding();
33
34         String name = methodInvocation.getName().
             getFullyQualifiedName();
35         methodInvocations.add(name);
36         return true;
37       }
38     });
39     cu.accept(new ASTVisitor() {
40       public boolean visit(org.eclipse.jdt.core.dom.
             MethodDeclaration methodDecl) {
41         String name = methodDecl.getName().getFullyQualifiedName()
              ;
42         if(Modifier.isPrivate(methodDecl.getModifiers()) &&
43             !methodInvocations.contains(name) ) {
44           System.out.println("'" + name + "' is private and is not
                used in this class , will be deleted now.");
45           methodDecl.delete();
46           return false;
47         }
48         return true;
49       }
50     });
51     cu.accept(new ASTVisitor() {
52       public boolean visit(org.eclipse.jdt.core.dom.
             MethodInvocation v) {
53         if(v.getExpression() == null)
54           return true;
55         String name = v.getExpression().toString(); //var.getName
              ().getFullyQualifiedName(); //var.getFullyQualifiedName
              ();
56         vars.add(name);
57         return true;
58       }
59       public boolean visit(org.eclipse.jdt.core.dom.
             InfixExpression v) {
60         if(v.getLeftOperand() instanceof SimpleName) {
61           String name = ((SimpleName)v.getLeftOperand()).
                getFullyQualifiedName();
62           vars.add(name);
63         }
64         return true;
65       }
```

```
66      });
67      cu.accept(new ASTVisitor() {
68        public boolean visit(org.eclipse.jdt.core.dom.
              FieldDeclaration variableDecl) {
69          List<VariableDeclarationFragment> list = variableDecl.
                fragments();
70          for(int i=list.size()-1; i>=0; i--) {
71            String name = list.get(i).getName().
                  getFullyQualifiedName();
72            if(vars.contains(name)) {
73              continue; // this field variable is used later
74            }
75            if(Modifier.isPrivate(variableDecl.getModifiers())) {
76              System.out.println("'" + name + "' this field variable
                    is private and is not used in this class, will be
                    deleted now.");
77              if(list.size() > 1)
78                list.get(i).delete();
79              else
80                variableDecl.delete();
81            }
82          }
83          return true;
84        }
85      });
86  //  this.printMethodInvocations();
87  //  this.printUsedVars();
88    }
89
90    private ArrayList<String> methodInvocations = new ArrayList<>();
91    private ArrayList<String> vars = new ArrayList<>();
92
93    public void printMethodInvocations() {
94      System.out.println("Invoked methods:");
95      for(String methodName: methodInvocations) {
96        System.out.println(methodName);
97      }
98    }
99
100   public void printUsedVars() {
101     System.out.println("used vars:");
102     for(String var: vars) {
103       System.out.println(var);
104     }
105   }
106 }
```

Listing A25: ASTOptimizer.java in AST project

```java
package de.tu_freiberg.informatik.vonwenckstern;

import java.lang.reflect.Modifier;

import org.eclipse.jdt.core.dom.ASTNode;
import org.eclipse.jdt.core.dom.ASTVisitor;
import org.eclipse.jdt.core.dom.CompilationUnit;

/**
 * this file changes the package and the class name, so that
 * the new file can be successfully scanned in Eclipse to
 * format the source code
 */
public class ASTRenamer extends ASTVisitor {
  private String className;

  public static void exec(ASTNode node, String newClassName) {
    CompilationUnit cu = (CompilationUnit) node;
    cu.accept(new ASTRenamer(newClassName));
  }

  public ASTRenamer(String newClassName) {
    this.className = newClassName;
  }

  public boolean visit(org.eclipse.jdt.core.dom.PackageDeclaration
      packageDecl) {
    String pName = className.substring(0,className.lastIndexOf("."
        ));
    packageDecl.setName(packageDecl.getAST().newName(pName));
    return false;
  }

  public boolean visit(org.eclipse.jdt.core.dom.TypeDeclaration
      typeDeclaration) {
    if(Modifier.isPublic(typeDeclaration.getModifiers())) {
      String name = className.substring(className.lastIndexOf(".")
          +1);
      typeDeclaration.setName(typeDeclaration.getAST().
          newSimpleName(name));
    }
    return false;
  }
}
```

Listing A26: ASTRenamer.java in AST project

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import com.google.gwt.core.client.EntryPoint;
4  import com.google.gwt.event.dom.client.ClickEvent;
5  import com.google.gwt.event.dom.client.ClickHandler;
6  import com.google.gwt.user.client.Window;
7  import com.google.gwt.user.client.ui.Button;
8  import com.google.gwt.user.client.ui.RootPanel;
9
10 public class ImageViewer implements EntryPoint {
11   public void onModuleLoad() {
12     Button b = new Button("get Today()");
13     b.addClickHandler(new ClickHandler() {
14       @Override
15       public void onClick(ClickEvent event) {
16         /** Today.getToday() uses the java.util.GregorianCalendar
               class which is not part of the JREE, and
17          *  so this project cannot be compiled, but it can be
               tested in the development mode.
18          *  The aim of this project is to show that development
               mode uses the standard JRE and to give an
19          *  example of a project which can be tested but not
               compiled.*/
20         Window.alert(Today.getToday());
21       }
22     });
23     RootPanel.get().add(b);
24   }
25 }
```

Listing A27: ImageViewer.java in JREE calendar project

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import java.util.Calendar;
4  import java.util.GregorianCalendar;
5
6  public class Today {
7    public static String getToday() {
8      GregorianCalendar calendar = new GregorianCalendar();
9      String s = calendar.get(Calendar.DAY_OF_MONTH) + "." +
10                    Integer.toString(calendar.get(Calendar.MONTH) +
                          1) + "." +
11                    calendar.get(Calendar.YEAR);
12     s += "\nactual week:" + calendar.get(Calendar.WEEK_OF_YEAR);
13     return s;
14   }
15 }
```

Listing A28: Today.java in JREE calendar project

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import com.google.gwt.canvas.client.Canvas;
4  import com.google.gwt.canvas.dom.client.Context2d;
5  import com.google.gwt.canvas.dom.client.CssColor;
6  import com.google.gwt.core.client.EntryPoint;
7  import com.google.gwt.dom.client.VideoElement;
8  import com.google.gwt.media.client.Audio;
9  import com.google.gwt.media.client.Video;
10 import com.google.gwt.user.client.ui.Anchor;
11 import com.google.gwt.user.client.ui.Button;
12 import com.google.gwt.user.client.ui.CheckBox;
13 import com.google.gwt.user.client.ui.DoubleBox;
14 import com.google.gwt.user.client.ui.FlexTable;
15 import com.google.gwt.user.client.ui.Image;
16 import com.google.gwt.user.client.ui.ListBox;
17 import com.google.gwt.user.client.ui.LongBox;
18 import com.google.gwt.user.client.ui.PushButton;
19 import com.google.gwt.user.client.ui.RadioButton;
20 import com.google.gwt.user.client.ui.RichTextArea;
21 import com.google.gwt.user.client.ui.RootPanel;
22 import com.google.gwt.user.client.ui.SimpleCheckBox;
23 import com.google.gwt.user.client.ui.SimpleRadioButton;
24 import com.google.gwt.user.client.ui.TextArea;
25 import com.google.gwt.user.client.ui.TextBox;
26 import com.google.gwt.user.client.ui.ToggleButton;
27
28 public class FocusWidgets extends FlexTable implements EntryPoint
      {
29   public void onModuleLoad() {
30     setText(0, 0, "RichTextArea:");
31     RichTextArea rta = new RichTextArea();
32     rta.setHTML("<b><u>GWT</u></b> stands for <i>G</i>oogle <i>W</
         i>eb <i>T</i>oolkit.");
33     setWidget(0,1, rta);
34     setText(1,0, "Anchor:");
35     setWidget(1,1, new Anchor("Jump downwards!"));
36     setText(2,0, "Button");
37     setWidget(2,1, new Button("Please click at me!"));
38     setText(3,0, "CheckBox:");
39     setWidget(3,1, new CheckBox("Tomato"));
40     setText(4,0, "");
41     setWidget(4,1, new CheckBox("Banana"));
42     setText(5,0, "RadioButton:");
43     RadioButton rb = new RadioButton("gender","male");
44     rb.setValue(true); // select this option
45     setWidget(5,1, rb);
46     setText(6,0, "");
47     setWidget(6,1, new RadioButton("gender","female"));
48     setText(7, 0, "PushButton:");
49     Image im = new Image("http://upload.wikimedia.org/wikipedia/en
         /b/ba/Flag_of_Germany.svg");
50     im.setPixelSize(200, 100);
51     setWidget(7,1, new PushButton(im));
52     setText(8, 0, "ToggleButton::");
```

```
53      setWidget(8,1, new ToggleButton("Press␣me␣down!"));
54      setText(9, 0, "ListBox:");
55      ListBox lb = new ListBox();
56        lb.addItem("USA");
57        lb.addItem("Germany");
58        lb.addItem("France");
59        lb.addItem("UK");
60        lb.addItem("Spain");
61        lb.setVisibleItemCount(5);
62      setWidget(9,1, lb);
63      setText(10, 0, "Audio:");
64      Audio audio = Audio.createIfSupported();
65      if(audio != null) {
66        audio.setSrc("http://downloads.bbc.co.uk/doctorwho/sounds/
              everycitizen.mp3");
67        audio.setControls(true);
68        setWidget(10,1, audio);
69      } else {
70        setText(10,1, "Your␣Browser␣does␣not␣support␣audio␣playback!
              ");
71      }
72      setText(11,0, "Video:");
73      Video video = Video.createIfSupported();
74      if(video != null) {
75        video.addSource("http://html5demos.com/assets/dizzy.mp4",
              VideoElement.TYPE_MP4);
76        video.addSource("http://html5demos.com/assets/dizzy.ogv",
              VideoElement.TYPE_OGG);
77        video.setControls(true);
78        setWidget(11,1,video);
79      } else {
80        setText(11,1, "Your␣Browser␣does␣not␣support␣video␣playback!
              ");
81      }
82      setText(12,0, "Canvas");
83      Canvas canvas = Canvas.createIfSupported();
84      if(canvas != null) {
85        Context2d context = canvas.getContext2d();
86        context.setFillStyle(CssColor.make(255, 0, 0).value()); //
              makes the fill color red
87        context.fillRect(10, 10, 30, 30);
88        context.setFillStyle(CssColor.make(0, 0, 255).value()); //
              makes the fill color red
89        context.fillText("Drawing", 20, 70);
90        setWidget(12,1, canvas);
91      } else {
92        setText(12,1, "Your␣Browser␣does␣not␣support␣the␣canvas␣
              element!");
93      }
94
95      setText(13,0, "SimpleCheckBox:");
96      setWidget(13,1, new SimpleCheckBox());
97      setText(14,0, "SimpleRadioButton:");
98      setWidget(14,1, new SimpleRadioButton("p2"));
99      setText(15,0, "");
100     setWidget(15,1, new SimpleRadioButton("p2"));
```

```
101       setText(16,0, "TextArea:");
102       TextArea ta = new TextArea();
103       ta.setValue("The user can insert\nhere mulitple lines.");
104       setWidget(16,1, ta);
105       setText(17,0, "TextBox:");
106       TextBox tb = new TextBox();
107       tb.setValue("The user can insert only one line.");
108       setWidget(17,1, tb);
109       setText(18,0, "DoubleBox:");
110       DoubleBox db = new DoubleBox();
111       db.setValue(2.3);
112       setWidget(18,1, db);
113       setText(19,0, "LongBox:");
114       LongBox longb = new LongBox();
115       longb.setValue(1000L);
116       setWidget(19,1, longb);
117
118       RootPanel.get().add(this);
119
120     }
121 }
```

Listing A29: Example project showing all extended classes of FocusWidget

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3
4  import java.util.ArrayList;
5  import java.util.Arrays;
6  import java.util.Date;
7  import java.util.List;
8
9  import com.google.gwt.cell.client.NumberCell;
10 import com.google.gwt.cell.client.TextCell;
11 import com.google.gwt.core.client.EntryPoint;
12 import com.google.gwt.editor.ui.client.ValueBoxEditorDecorator;
13 import com.google.gwt.text.shared.AbstractRenderer;
14 import com.google.gwt.user.cellview.client.CellList;
15 import com.google.gwt.user.cellview.client.CellTable;
16 import com.google.gwt.user.cellview.client.CellTree;
17 import com.google.gwt.user.cellview.client.Column;
18 import com.google.gwt.user.cellview.client.
       HasKeyboardSelectionPolicy.KeyboardSelectionPolicy;
19 import com.google.gwt.user.cellview.client.SimplePager;
20 import com.google.gwt.user.cellview.client.TextColumn;
21 import com.google.gwt.user.client.ui.CaptionPanel;
22 import com.google.gwt.user.client.ui.DisclosurePanel;
23 import com.google.gwt.user.client.ui.DoubleBox;
24 import com.google.gwt.user.client.ui.FlexTable;
25 import com.google.gwt.user.client.ui.HTML;
26 import com.google.gwt.user.client.ui.Label;
27 import com.google.gwt.user.client.ui.MultiWordSuggestOracle;
28 import com.google.gwt.user.client.ui.NotificationMole;
29 import com.google.gwt.user.client.ui.RootPanel;
30 import com.google.gwt.user.client.ui.StackPanel;
31 import com.google.gwt.user.client.ui.SuggestBox;
32 import com.google.gwt.user.client.ui.TabPanel;
```

```
33 import com.google.gwt.user.client.ui.ValueListBox;
34 import com.google.gwt.user.client.ui.ValuePicker;
35 import com.google.gwt.user.client.ui.VerticalPanel;
36 import com.google.gwt.user.datepicker.client.DateBox;
37 import com.google.gwt.user.datepicker.client.DatePicker;
38 import com.google.gwt.view.client.ListDataProvider;
39 import com.google.gwt.view.client.TreeViewModel;
40
41 public class Composites extends FlexTable implements EntryPoint {
42   private static final List<String> DAYS = Arrays.asList("Sunday",
         "Monday",
43           "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
               );
44   private final static String ImageBase64 =
45       "<img␣alt=\"\"␣src=\"...\"␣/>";
46   public void onModuleLoad() {
47     NotificationMole m = new NotificationMole();
48     m.setMessage("Hallo,␣I␣am␣a␣Notification␣mole");
49     m.setAnimationDuration(5000);
50     RootPanel.get().add(m, 40, 200); // they must get added to the
           DOM before showing up
51     m.show();
52
53     setText(0, 0, "CaptionPanel:");
54     CaptionPanel cp = new CaptionPanel("Even␣<font␣color=\"red\">
           decorated␣<b>headers</b>␣are</font>␣<span␣style=\"
           background-color:green;␣\">possible</span>", true);
55     cp.add(new HTML("I␣am␣the␣only␣widget␣in␣the␣body<br>" +
           ImageBase64));
56     setWidget(0,1, cp);
57
58     setText(1, 0, "DateBox:");
59     DateBox db = new DateBox();
60     @SuppressWarnings("deprecation")
61     Date date = new Date(2008-1900, 10-1, 9); // 9th Oct 2008
62     db.setValue(date);
63     setWidget(1,1, db);
64
65     AbstractRenderer<String> simpleRenderer = new AbstractRenderer
           <String>() {
66       @Override
67       public String render(String object) {
68         return object;
69       }
70     };
71
72     setText(2, 0, "ValuePicker:");
73       ValuePicker<String> vp = new ValuePicker<String>(
             simpleRenderer);
74     vp.setAcceptableValues(DAYS);
75     vp.setValue(DAYS.get(1));
76     setWidget(2,1, vp);
77
78     setText(3, 0, "ValueListBox:");
79     ValueListBox<String> vl = new ValueListBox<String>(
           simpleRenderer);
```

```
80      vl.setAcceptableValues(DAYS);
81      vl.setValue(DAYS.get(3));
82      setWidget(3,1, vl);
83
84      setText(4, 0, "DatePicker:");
85      DatePicker dateP = new DatePicker();
86      dateP.setValue(date);
87      setWidget(4,1, dateP);
88
89      setText(5, 0, "ValueBoxEditorDecorator:");
90      ValueBoxEditorDecorator<Double> vbed = new
            ValueBoxEditorDecorator<Double>();
91      DoubleBox doubleBox = new DoubleBox();
92      doubleBox.setValue(3.45);
93      vbed.setValueBox(doubleBox);
94      setWidget(5,1, vbed);
95
96      setText(6, 0, "DisclosurePanel:");
97      DisclosurePanel disPanel = new DisclosurePanel("I␣am␣the␣
            header␣text!");
98      disPanel.add(new Label("Click␣on␣the␣header␣to␣hide␣me."));
99      disPanel.setOpen(true);
100     setWidget(6,1, disPanel);
101
102     setText(7, 0, "CellTree:");
103     TreeViewModel model = new TreeViewModel() {
104       public <T> NodeInfo<?> getNodeInfo(T value) {
105         ListDataProvider<String> dataProvider = new
                ListDataProvider<String>();
106         for (int i = 0; i < 5; i++) {
107       // creates 1.1; 2.3.4 and so on
108     dataProvider.getList().add(value + String.valueOf(i) + ".");
109         }
110         // Return a node info that pairs the data with a cell.
111         return new DefaultNodeInfo<String>(dataProvider, new
                TextCell());
112       }
113       public boolean isLeaf(Object value) {
114         // We want a maximal depth of three
115         return value.toString().length() - value.toString().
                replace(".", "").length() >= 3;
116       }
117     };
118
119     CellTree tree = new CellTree(model, "");
120     setWidget(7,1, tree);
121
122     setText(8, 0, "CellTable:");
123     CellTable<String[]> table = new CellTable<String[]>();
124     table.setKeyboardSelectionPolicy(KeyboardSelectionPolicy.
            ENABLED);
125     TextColumn<String[]> nameCol = new TextColumn<String[]>() {
126       @Override
127       public String getValue(String[] object) {
128         return object[0];
129       }
```

```
130       };
131       table.addColumn(nameCol, "Name");
132       Column<String[], Number> matCol = new Column<String[], Number
              >(new NumberCell()) {
133         @Override
134         public Integer getValue(String[] object) {
135           return Integer.parseInt(object[1]);
136         }
137       };
138       table.addColumn(matCol, "Matrikel number");
139       List<String[]> data = new ArrayList<String[]>();
140       data.add(new String[]{"Tom Bayer", "12345"});
141       data.add(new String[]{"Max Noway", "87445"});
142       table.setRowCount(2);
143       table.setRowData(data);
144       setWidget(8,1, table);
145
146       setText(8, 0, "CellList / SimplePager:");
147       CellList<String> cellList = new CellList<String>(new TextCell
              ());
148         ListDataProvider<String> dataProvider = new ListDataProvider
              <String>();
149         List<String> pageData = dataProvider.getList();
150         for (int i = 0; i < 200; i++) {
151           pageData.add("data " + i);
152         }
153         dataProvider.addDataDisplay(cellList);
154         cellList.setPageSize(4);
155         SimplePager pager = new SimplePager();
156         pager.setDisplay(cellList);
157         // Add the pager and list to the page.
158         VerticalPanel vPanel = new VerticalPanel();
159         vPanel.add(cellList);
160         vPanel.add(pager);
161         setWidget(9,1, vPanel);
162
163         setText(10, 0, "StackPanel:");
164         StackPanel stack = new StackPanel();
165         Label label = new Label("One One One One One One One One One
              One One One One One One One");
166         stack.add(label, "One", false);
167         label = new Label("Two Two Two Two Two Two Two Two Two Two
              Two Two Two Two Two Two");
168         stack.add(label, "Two", false);
169         label = new Label("Three Three Three Three Three Three Three
              Three Three Three Three");
170         stack.add(label, "Three", false);
171         stack.setSize("400px", "200px");
172         setWidget(10,1, stack);
173
174         setText(11, 0, "TabPanel:");
175         TabPanel tabPanel = new TabPanel();
176         tabPanel.add(new HTML("One<br>One<br>One<br>One"), "One");
177         tabPanel.add(new HTML("Two<br>Two<br>Two<br>Two"), "Two");
178         tabPanel.add(new HTML("Three<br>Three<br>Three<br>Three"), "
              Three");
```

```
179         tabPanel.selectTab(0);
180         tabPanel.setSize("500px", "200px");
181         setWidget(11,1, tabPanel);
182 
183         setText(12, 0, "SuggestBox: (insert an Euro country)");
184         MultiWordSuggestOracle oracle = new MultiWordSuggestOracle()
                ;
185         oracle.addAll(Arrays.asList(new String[] {"Austria", "
                Belgium", "Cyprus", "Estonia", "Finland", "France", "
                Germany", "Greece", "Ireland", "Italy", "Luxembourg", "
                Malta", "Netherlands", "Portugal", "Slovakia", "Slovenia"
                , "Spain"}));
186         SuggestBox suggestbox = new SuggestBox(oracle);
187         setWidget(12,1, suggestbox);
188 
189       RootPanel.get().add(this);
190     }
191 }
```

Listing A30: Example project showing nearly all extended classes of Composite

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2 
3  import com.google.gwt.core.client.EntryPoint;
4  import com.google.gwt.dom.client.Style.Unit;
5  import com.google.gwt.event.dom.client.ClickEvent;
6  import com.google.gwt.event.dom.client.ClickHandler;
7  import com.google.gwt.user.client.DOM;
8  import com.google.gwt.user.client.ui.Button;
9  import com.google.gwt.user.client.ui.DialogBox;
10 import com.google.gwt.user.client.ui.DockLayoutPanel;
11 import com.google.gwt.user.client.ui.FlexTable;
12 import com.google.gwt.user.client.ui.FlowPanel;
13 import com.google.gwt.user.client.ui.FormPanel;
14 import com.google.gwt.user.client.ui.HTML;
15 import com.google.gwt.user.client.ui.HTMLPanel;
16 import com.google.gwt.user.client.ui.HeaderPanel;
17 import com.google.gwt.user.client.ui.HorizontalPanel;
18 import com.google.gwt.user.client.ui.Label;
19 import com.google.gwt.user.client.ui.RootPanel;
20 import com.google.gwt.user.client.ui.ScrollPanel;
21 import com.google.gwt.user.client.ui.TextBox;
22 import com.google.gwt.user.client.ui.VerticalPanel;
23 
24 public class Panels extends FlexTable implements EntryPoint {
25   private final static String Hamlet = "Sir, in my heart there was
         a kind of fighting <br>That would not let me sleep. Methought
         I lay <br>Worse than the mutines in the bilboes. Rashly -<br>
         And prais'd be rashness for it - let us know<br>Our
         indiscretion sometimes serves us well... <br>Hamlet, Act 5,
         Scene 2, 4-8";
26   public void onModuleLoad() {
27     setText(0, 0, "FlexTable:");
28     FlexTable table = new FlexTable();
29     table.setCellPadding(0);
30     table.setCellSpacing(0);
```

```
31      DOM.setStyleAttribute(table.getElement(), "border", "1px␣solid
            ␣black");
32      FlexCellFormatter formatter = table.getFlexCellFormatter();
33      for(int row =0; row < 6; row++) {
34        for(int col=0; col<4; col++) {
35          table.setText(row, col, "(" + row + "␣,␣" + col + ")");
36          DOM.setStyleAttribute(formatter.getElement(row, col), "
                border", "1px␣solid␣black");
37        }
38      }
39      formatter.setRowSpan(1, 1, 3);
40      formatter.setColSpan(1, 1, 2);
41      table.removeCell(1, 3);
42      table.removeCells(2, 2, 2);
43      table.removeCells(3, 2, 2);
44      setWidget(0, 1, table);
45
46      setText(1, 0, "HeaderPanel:");
47      HeaderPanel hp = new HeaderPanel();
48      HTML header = new HTML("I␣am␣the␣header");
49      DOM.setStyleAttribute(header.getElement(), "backgroundColor",
            "lightblue");
50      hp.setHeaderWidget(header);
51      Label footer = new Label("I␣am␣the␣footer,␣or␣the␣statusbar");
52      DOM.setStyleAttribute(footer.getElement(), "backgroundColor",
            "lightgray");
53      hp.setFooterWidget(footer);
54      Button content = new Button("I␣am␣the␣content␣widget");
55      DOM.setStyleAttribute(hp.getElement(), "border", "1px␣dotted␣
            darkgray");
56      hp.setContentWidget(content);
57      hp.setPixelSize(500, 100);
58      setWidget(1, 1, hp);
59
60      setText(2, 0, "HTMLPanel:");
61  //          .lin-grad {
62  //            background:-moz-linear-gradient(left, red, orange,
        yellow, green, blue); /* Firefox */
63  //            background:-webkit-linear-gradient(left, red, orange,
        yellow, green, blue); /* Safari, Chrome */
64  //            background:-o-linear-gradient(left, red, orange,
        yellow, green, blue); /* Opera */
65  //            background:-ms-linear-gradient(left, red, orange,
        yellow, green, blue); /* IE */
66  //            background:linear-gradient(left, red, orange, yellow,
        green, blue); /* W3C Standard */
67  //          }
68  //
69  //          .rad-grad {
70  //            background: -moz-radial-gradient(red, yellow, #1E90FF)
        ;
71  //            background: webkit-radial-gradient(red, yellow, #1
        E90FF);
72  //            background: -o-radial-gradient(red, yellow, #1E90FF);
73  //            background: -ms-radial-gradient(red, yellow, #1E90FF);
74  //            background: radial-gradient(red, yellow, #1E90FF);
```

```
75 //        }
76     String layoutHTML = "<div id='one' class='lin-grad' style='
           width: 200px; height: 30px;'></div>"+
77         "<div id='two' class='rad-grad' style='width: 200px;
               height: 200px;'></div>";
78     HTMLPanel htmlP = new HTMLPanel(layoutHTML);
79     htmlP.setPixelSize(300, 230);
80     htmlP.add(new TextBox(), "one");
81     htmlP.add(new Button("No op!"), "two");
82     setWidget(2, 1, htmlP);
83
84     setText(3, 0, "FlowPanel:");
85     FlowPanel flowP = new FlowPanel();
86     for(int nb = 0; nb < 16; nb++) {
87       flowP.add(new Button("Button " + nb));
88     }
89     flowP.setPixelSize(200, 150);
90     setWidget(3, 1, flowP);
91
92     setText(4, 0, "DockLayoutPanel:");
93     DockLayoutPanel dockLayoutP = new DockLayoutPanel(Unit.EM);
94     header = new HTML("header");
95     DOM.setStyleAttribute(header.getElement(), "border", "5px
           solid lightgray");
96     dockLayoutP.addNorth(header, 2);
97     footer = new HTML("footer");
98     DOM.setStyleAttribute(footer.getElement(), "border", "5px
           solid lightgray");
99     dockLayoutP.addSouth(footer, 2);
100    HTML nav = new HTML("navigation");
101    DOM.setStyleAttribute(nav.getElement(), "borderLeft", "5px
           solid lightgray");
102    DOM.setStyleAttribute(nav.getElement(), "borderRight", "5px
           solid lightgray");
103    dockLayoutP.addWest(nav, 10);
104    HTML shakespeare = new HTML(Hamlet);
105    dockLayoutP.add(shakespeare);
106    DOM.setStyleAttribute(shakespeare.getElement(), "borderRight"
           , "5px solid lightgray");
107    dockLayoutP.setPixelSize(400, 200);
108    setWidget(4, 1, dockLayoutP);
109
110    setText(5, 0, "HorizontalPanel:");
111    HorizontalPanel horP = new HorizontalPanel();
112    for(int nb = 0; nb < 5; nb++) {
113      horP.add(new Button("Button " + nb));
114    }
115    setWidget(5, 1, horP);
116
117    setText(6, 0, "VerticalPanel:");
118    VerticalPanel verP = new VerticalPanel();
119    for(int nb = 0; nb < 3; nb++) {
120      verP.add(new Button("Button " + nb));
121    }
122    setWidget(6, 1, verP);
123
```

```
124     setText(7, 0, "ScrollPanel:");
125     ScrollPanel scrollP = new ScrollPanel(new HTML(Hamlet));
126     scrollP.setPixelSize(200, 100);
127     setWidget(7, 1, scrollP);
128
129     setText(8, 0, "FormPanel:");
130     final FormPanel formP = new FormPanel();
131     formP.setAction("javascript:alert('normally this would be a
            post or a get request to a URL')");
132     formP.setMethod(FormPanel.METHOD_GET);
133     HorizontalPanel hPanel = new HorizontalPanel();
134     hPanel.add(new TextBox());
135     hPanel.add(new Button("Submit", new ClickHandler() {
136           public void onClick(ClickEvent event) {
137         formP.submit();
138       }
139           }));
140     formP.add(hPanel);
141     setWidget(8, 1, formP);
142
143     DialogBox box = new DialogBox();
144     box.setText("I am a DialogBox.");
145     box.setTitle("Title");
146     box.setAnimationEnabled(true);
147     box.setWidget(new Button("OK"));
148     box.show();
149
150     RootPanel.get().add(this);
151   }
152 }
```

Listing A31: Example project showing nearly all extended classes of Panel

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import java.util.logging.Level;
4  import java.util.logging.Logger;
5
6  import com.google.gwt.core.client.EntryPoint;
7  import com.google.gwt.dom.client.Element;
8  import com.google.gwt.event.dom.client.ClickEvent;
9  import com.google.gwt.event.dom.client.ClickHandler;
10 import com.google.gwt.event.dom.client.DoubleClickEvent;
11 import com.google.gwt.event.dom.client.DoubleClickHandler;
12 import com.google.gwt.event.dom.client.DragEndEvent;
13 import com.google.gwt.event.dom.client.DragEndHandler;
14 import com.google.gwt.event.dom.client.DragEnterEvent;
15 import com.google.gwt.event.dom.client.DragEnterHandler;
16 import com.google.gwt.event.dom.client.DragEvent;
17 import com.google.gwt.event.dom.client.DragHandler;
18 import com.google.gwt.event.dom.client.DragLeaveEvent;
19 import com.google.gwt.event.dom.client.DragLeaveHandler;
20 import com.google.gwt.event.dom.client.DragOverEvent;
21 import com.google.gwt.event.dom.client.DragOverHandler;
22 import com.google.gwt.event.dom.client.DragStartEvent;
23 import com.google.gwt.event.dom.client.DragStartHandler;
24 import com.google.gwt.event.dom.client.DropEvent;
```

```
25 import com.google.gwt.event.dom.client.DropHandler;
26 import com.google.gwt.event.dom.client.FocusEvent;
27 import com.google.gwt.event.dom.client.FocusHandler;
28 import com.google.gwt.event.dom.client.KeyDownEvent;
29 import com.google.gwt.event.dom.client.KeyDownHandler;
30 import com.google.gwt.event.dom.client.KeyPressEvent;
31 import com.google.gwt.event.dom.client.KeyPressHandler;
32 import com.google.gwt.event.dom.client.KeyUpEvent;
33 import com.google.gwt.event.dom.client.KeyUpHandler;
34 import com.google.gwt.event.dom.client.MouseDownEvent;
35 import com.google.gwt.event.dom.client.MouseDownHandler;
36 import com.google.gwt.event.dom.client.MouseMoveEvent;
37 import com.google.gwt.event.dom.client.MouseMoveHandler;
38 import com.google.gwt.event.dom.client.MouseOutEvent;
39 import com.google.gwt.event.dom.client.MouseOutHandler;
40 import com.google.gwt.event.dom.client.MouseOverEvent;
41 import com.google.gwt.event.dom.client.MouseOverHandler;
42 import com.google.gwt.event.dom.client.MouseUpEvent;
43 import com.google.gwt.event.dom.client.MouseUpHandler;
44 import com.google.gwt.event.dom.client.MouseWheelEvent;
45 import com.google.gwt.event.dom.client.MouseWheelHandler;
46 import com.google.gwt.event.logical.shared.InitializeEvent;
47 import com.google.gwt.event.logical.shared.InitializeHandler;
48 import com.google.gwt.user.client.ui.CheckBox;
49 import com.google.gwt.user.client.ui.FlowPanel;
50 import com.google.gwt.user.client.ui.Label;
51 import com.google.gwt.user.client.ui.RichTextArea;
52 import com.google.gwt.user.client.ui.RootPanel;
53 import com.google.gwt.user.client.ui.Widget;
54
55 public class Handlers extends FlowPanel implements EntryPoint,
       ClickHandler, DoubleClickHandler, DragStartHandler, DropHandler
       , DragEndHandler, DragEnterHandler, DragHandler,
       DragLeaveHandler,
56      DragOverHandler, FocusHandler, InitializeHandler,
            KeyDownHandler, KeyPressHandler, KeyUpHandler,
            MouseDownHandler, MouseMoveHandler, MouseOutHandler,
            MouseOverHandler, MouseUpHandler,
57      MouseWheelHandler {
58   RichTextArea rt1 = new RichTextArea();
59   RichTextArea rt2 = new RichTextArea();
60   CheckBox checkClick = new CheckBox("onClick");
61   CheckBox checkDoubleClick = new CheckBox("onDoubleClick");
62   CheckBox checkDragStart = new CheckBox("onDragStart");
63   CheckBox checkDrop = new CheckBox("onDrop");
64   CheckBox checkDragEnd = new CheckBox("onDragEnd");
65   CheckBox checkDragEnter = new CheckBox("onDragEnter");
66   CheckBox checkDrag = new CheckBox("onDrag");
67   CheckBox checkDragLeave = new CheckBox("onDragLeave");
68   CheckBox checkDragOver = new CheckBox("onDragOver");
69   CheckBox checkFocus = new CheckBox("onFocus");
70   CheckBox checkInitialize = new CheckBox("onInitialize");
71   CheckBox checkKeyDown = new CheckBox("onKeyDown");
72   CheckBox checkKeyPress = new CheckBox("onKeyPress");
73   CheckBox checkKeyUp = new CheckBox("onKeyUp");
74   CheckBox checkMouseDown = new CheckBox("onMouseDown");
```

```
75    CheckBox checkMouseMove = new CheckBox("onMouseMove");
76    CheckBox checkMouseOut = new CheckBox("onMouseOut");
77    CheckBox checkMouseOver = new CheckBox("onMouseOver");
78    CheckBox checkMouseUp = new CheckBox("onMouseUp");
79    CheckBox checkMouseWheel = new CheckBox("onMouseWheel");
80
81    // to enable logging, add the following to Handlers.gwt.xml:
82    // <inherits name="com.google.gwt.logging.Logging"/>
83    Logger logger = Logger.getLogger("HandlerLogger");
84
85    public void onModuleLoad() {
86      this.addDomHandler(this, DropEvent.getType());
87
88      setPixelSize(600, 600);
89      rt1.setText("Type and select any text here!");
90      rt1.setPixelSize(600, 300);
91      rt1.setTitle("top RichTextArea");
92      rt1.addDropHandler(this);
93      rt1.addFocusHandler(this);
94      rt1.addInitializeHandler(this);
95      rt1.addKeyDownHandler(this);
96      rt1.addKeyPressHandler(this);
97      rt1.addKeyUpHandler(this);
98      rt1.addMouseDownHandler(this);
99      rt1.addMouseMoveHandler(this);
100     rt1.addMouseOutHandler(this);
101     rt1.addMouseOverHandler(this);
102     rt1.addMouseUpHandler(this);
103     rt1.addMouseWheelHandler(this);
104     rt1.addClickHandler(this);
105     rt1.addDoubleClickHandler(this);
106     rt1.addDragStartHandler(this);
107
108     rt2.setText("Drag and drop text between these text boxes");
109     rt2.setPixelSize(600, 300);
110     rt2.setTitle("bottom RichTextArea");
111
112     rt2.addDropHandler(this);
113     rt2.addFocusHandler(this);
114     rt2.addInitializeHandler(this);
115     rt2.addKeyDownHandler(this);
116     rt2.addKeyPressHandler(this);
117     rt2.addKeyUpHandler(this);
118     rt2.addMouseDownHandler(this);
119     rt2.addMouseMoveHandler(this);
120     rt2.addMouseOutHandler(this);
121     rt2.addMouseOverHandler(this);
122     rt2.addMouseUpHandler(this);
123     rt2.addMouseWheelHandler(this);
124     rt2.addClickHandler(this);
125     rt2.addDoubleClickHandler(this);
126     rt2.addDragStartHandler(this);
127
128     Label label = new Label("test double click und drag me around"
          );
129     label.setTitle("Label");
```

```
130     label.addClickHandler(this);
131     label.addDoubleClickHandler(this);
132     label.addDragStartHandler(this);
133     label.addDropHandler(this);
134     label.addDragEndHandler(this);
135     label.addDragEnterHandler(this);
136     label.addDragHandler(this);
137     label.addDragLeaveHandler(this);
138     label.addDragOverHandler(this);
139     label.getElement().setDraggable(Element.DRAGGABLE_TRUE);
140
141     Label label2 = new Label("label2");
142     label2.setTitle("label2");
143     label2.addClickHandler(this);
144     label2.addDoubleClickHandler(this);
145     label2.addDragStartHandler(this);
146     label2.addDropHandler(this);
147     label2.addDragEndHandler(this);
148     label2.addDragEnterHandler(this);
149     label2.addDragHandler(this);
150     label2.addDragLeaveHandler(this);
151     label2.addDragOverHandler(this);
152     label2.getElement().setDraggable(Element.DRAGGABLE_TRUE);
153
154     add(rt1);
155     add(rt2);
156     add(label);
157     add(label2);
158
159     checkInitialize.setValue(true);
160
161     FlowPanel fp = new FlowPanel();
162     fp.add(checkClick);
163     fp.add(checkDoubleClick);
164     fp.add(checkDrag);
165     fp.add(checkDragEnd);
166     fp.add(checkDragEnter);
167     fp.add(checkDragLeave);
168     fp.add(checkDragOver);
169     fp.add(checkDragStart);
170     fp.add(checkDrop);
171     fp.add(checkFocus);
172     fp.add(checkInitialize);
173     fp.add(checkKeyDown);
174     fp.add(checkKeyPress);
175     fp.add(checkKeyUp);
176     fp.add(checkMouseDown);
177     fp.add(checkMouseMove);
178     fp.add(checkMouseOut);
179     fp.add(checkMouseOver);
180     fp.add(checkMouseUp);
181     fp.add(checkMouseWheel);
182
183     add(fp);
184
185     RootPanel.get().add(this);
```

```
186     logger.setLevel(Level.ALL);
187       log("the application started");
188   }
189
190   public void onDoubleClick(DoubleClickEvent e) {
191     if(checkDoubleClick.getValue())
192       log("DoubleClickEvent[" + ((Widget)e.getSource()).getTitle()
              + "]: " +
193         e.getRelativeX(getElement()) + "px; " + e.getRelativeY(
              getElement()));
194   }
195
196   public void onClick(ClickEvent e) {
197     if(checkClick.getValue())
198       log("ClickEvent[" + ((Widget)e.getSource()).getTitle() + "]:
            " +
199         e.getRelativeX(getElement()) + "px; " + e.getRelativeY(
              getElement()));
200   }
201
202   private void log(String text) {
203     logger.log(Level.INFO, text);
204   }
205
206   @Override
207   public void onDragStart(DragStartEvent e) {
208     if(checkDragStart.getValue())
209     log("DragStartEvent[" + ((Widget)e.getSource()).getTitle() + "
          ]: ");
210   }
211
212   @Override
213   public void onDrop(DropEvent e) {
214     if(checkDrop.getValue())
215     log("DropEvent[" + ((Widget)e.getSource()).getTitle() + "]: ")
          ;
216   }
217
218   @Override
219   public void onDragOver(DragOverEvent e) {
220     if(checkDragOver.getValue())
221     log("DragOverEvent[" + ((Widget)e.getSource()).getTitle() + "
          ]: ");
222   }
223
224   @Override
225   public void onDragLeave(DragLeaveEvent e) {
226     if(checkDragLeave.getValue())
227     log("DragLeaveEvent[" + ((Widget)e.getSource()).getTitle() + "
          ]: ");
228   }
229
230   @Override
231   public void onDrag(DragEvent e) {
232     if(checkDrag.getValue())
233     log("DragEvent[" + ((Widget)e.getSource()).getTitle() + "]: ")
```

```
        ;
234   }
235
236   @Override
237   public void onDragEnter(DragEnterEvent e) {
238     if(checkDragEnter.getValue())
239     log("DragEnterEvent[" + ((Widget)e.getSource()).getTitle() + "
          ]:␣");
240   }
241
242   @Override
243   public void onDragEnd(DragEndEvent e) {
244     if(checkDragEnd.getValue())
245     log("DragEndEvent[" + ((Widget)e.getSource()).getTitle() + "]:
          ␣");
246   }
247
248   @Override
249   public void onMouseWheel(MouseWheelEvent e) {
250     if(checkMouseWheel.getValue())
251       log("MouseWheelEvent[" + ((Widget)e.getSource()).getTitle()
            + "]:␣"+ e.getDeltaY());
252   }
253
254   @Override
255   public void onMouseUp(MouseUpEvent e) {
256     if(checkMouseUp.getValue())
257       log("MouseUpEvent[" + ((Widget)e.getSource()).getTitle() + "
            ]:␣" + e.getClientX() + ",␣" + e.getClientY());
258   }
259
260   @Override
261   public void onMouseOver(MouseOverEvent e) {
262     if(checkMouseOver.getValue())
263       log("onMouseOver[" + ((Widget)e.getSource()).getTitle() + "
            ]:␣" + e.getClientX() + ",␣" + e.getClientY());
264   }
265
266   @Override
267   public void onMouseOut(MouseOutEvent e) {
268     if(checkMouseOut.getValue())
269       log("MouseOutEvent[" + ((Widget)e.getSource()).getTitle() +
            "]:␣" + e.getClientX() + ",␣" + e.getClientY());
270   }
271
272   @Override
273   public void onMouseMove(MouseMoveEvent e) {
274     if(checkMouseMove.getValue())
275       log("MouseMoveEvent[" + ((Widget)e.getSource()).getTitle() +
             "]:␣" + e.getClientX() + ",␣" + e.getClientY());
276   }
277
278   @Override
279   public void onMouseDown(MouseDownEvent e) {
280     if(checkMouseDown.getValue())
281       log("MouseDownEvent[" + ((Widget)e.getSource()).getTitle() +
```

```
                "]:␣" + e.getClientX() + ",␣" + e.getClientY());
282   }
283
284   @Override
285   public void onKeyUp(KeyUpEvent e) {
286     if(checkKeyUp.getValue())
287       log("KeyUpEvent[" + ((Widget)e.getSource()).getTitle() + "]:
              ␣" + e.getNativeKeyCode());
288   }
289
290   @Override
291   public void onKeyPress(KeyPressEvent e) {
292     if(checkKeyPress.getValue())
293       log("KeyPressEvent[" + ((Widget)e.getSource()).getTitle() +
              "]:␣" + e.getUnicodeCharCode());
294   }
295
296   @Override
297   public void onKeyDown(KeyDownEvent e) {
298     if(checkKeyDown.getValue())
299       log("KeyDownEvent[" + ((Widget)e.getSource()).getTitle() + "
              ]:␣" + e.getNativeKeyCode());
300   }
301
302   @Override
303   public void onInitialize(InitializeEvent e) {
304     if(checkInitialize.getValue())
305       log("InitializeEvent[" + ((Widget)e.getSource()).getTitle()
              + "]:␣");
306   }
307
308   @Override
309   public void onFocus(FocusEvent e) {
310     if(checkFocus.getValue())
311       log("FocusEvent[" + ((Widget)e.getSource()).getTitle() + "]:
              ␣");
312   }
313 }
```

Listing A32: Example project showing different EventHandler implementations

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import com.google.gwt.core.client.EntryPoint;
4  import com.google.gwt.dom.client.Element;
5  import com.google.gwt.dom.client.NativeEvent;
6  import com.google.gwt.event.dom.client.ContextMenuEvent;
7  import com.google.gwt.event.dom.client.ContextMenuHandler;
8  import com.google.gwt.user.client.DOM;
9  import com.google.gwt.user.client.Event;
10 import com.google.gwt.user.client.Event.NativePreviewEvent;
11 import com.google.gwt.user.client.Event.NativePreviewHandler;
12 import com.google.gwt.user.client.Window;
13 import com.google.gwt.user.client.ui.FlexTable;
14 import com.google.gwt.user.client.ui.HTML;
15 import com.google.gwt.user.client.ui.HorizontalPanel;
16 import com.google.gwt.user.client.ui.InlineLabel;
17 import com.google.gwt.user.client.ui.RootPanel;
18
19 public class DOMManipulation extends HorizontalPanel implements
       EntryPoint, NativePreviewHandler {
20   public void onModuleLoad() {
21     FlexTable tbl = new FlexTable();
22     tbl.setCellPadding(0);
23     tbl.setCellSpacing(0);
24     DOM.setElementPropertyInt(tbl.getElement(), "border", 1); //
           set a border to the table
25     for(int row=0; row < 5; row++) {
26       for(int col=0; col < 5; col++) {
27         tbl.setWidget(row, col, new HTML("("+row+",␣"+col+")"));
28       }
29     }
30     add(tbl);
31     DOM.setElementPropertyInt(getElement(), "cellSpacing", 10); //
           to have some space between the table and the label
32     add(new InlineLabel("Left␣click␣on␣a␣cell␣to␣select␣it.␣Right␣
           click␣to␣select␣the␣entire␣row."));
33     Event.addNativePreviewHandler(this);
34     RootPanel.get().add(this);
35     /** to prevent the default browser context menu to pop up */
36     RootPanel.get().addDomHandler(new ContextMenuHandler() {
37     @Override public void onContextMenu(ContextMenuEvent event) {
38       event.preventDefault();
39       event.stopPropagation();
40     }
41       }, ContextMenuEvent.getType());
42   }
43
44   @Override
45   public void onPreviewNativeEvent(NativePreviewEvent event) {
46     NativeEvent e = event.getNativeEvent();
47     if("mousedown".equalsIgnoreCase(e.getType())) {
48       // now we are processing the click event,
49       // using mousedown insead of click to get also the right
           click
50       Element el = e.getEventTarget().cast();
```

```
51        String tag = el.getTagName().toLowerCase();
52        if("span".equals(tag)) {
53          Window.alert("You should click into a cell and not at the
              label.");
54        } else if("div".equals(tag)) {
55          if(e.getButton() == NativeEvent.BUTTON_LEFT) {
56            DOM.setStyleAttribute((com.google.gwt.user.client.
                Element) el, "backgroundColor", "yellow");
57          } else if(e.getButton() == NativeEvent.BUTTON_RIGHT) {
58            Element parent = el;
59            do {
60              parent = parent.getParentElement();
61            } while(!"tr".equalsIgnoreCase(parent.getTagName()));
62            for(int i=0; i<parent.getChildCount(); i++) {
63              Element child = (Element) parent.getChild(i);
64              while(!"div".equalsIgnoreCase(child.getTagName())) {
65                child = child.getFirstChildElement();
66              }
67              DOM.setStyleAttribute((com.google.gwt.user.client.
                    Element) child, "backgroundColor", "red");
68            }
69          }
70        } else {
71          Window.alert("You missed the cell!");
72        }
73      }
74    }
75  }
```

Listing A33: DOMManipulation.java

```
1  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.
       ent">
2  <ui:UiBinder xmlns:ui="urn:ui:com.google.gwt.uibinder"
3    xmlns:g="urn:import:com.google.gwt.user.client.ui">
4    <ui:style>
5      .sizeDockLayoutPanel { width: 800px; height: 300px;  }
6      .top { letter-spacing: 10px; }
7      .can { background-color: navy; font: italic; color: white; }
8      .styled { border: dotted 3px lime; padding: 15px; }
9      .west { padding: 10px; background-color: lightgray; }
10   </ui:style>
11   <g:DockLayoutPanel unit='PX' styleName="{style.
         sizeDockLayoutPanel}">
12   <g:north size="50">
13     <g:HTML styleName="{style.top}">To<b>p</b> <span class="{style
           .can}">can</span> get <span class="{style.styled}">styled</
           span></g:HTML>
14   </g:north>
15   <g:center>
16     <g:Grid>
17       <g:row>
18         <g:cell>Some <b>widgets</b></g:cell>
19         <g:customCell><g:ListBox visibleItemCount="3">
20           <g:item value="1">first item</g:item>
21           <g:item value="2">second item</g:item>
22           <g:item value="3">third item</g:item>
```

```
23          </g:ListBox></g:customCell>
24          <g:customCell><g:CheckBox checked="true" text="I␣like␣GWT"
                /></g:customCell>
25          <g:customCell><g:CheckBox text="I␣eat␣only␣pizza" /></
                g:customCell>
26          <g:customCell><g:Button text="Click␣me." /></g:customCell>
27        </g:row>
28        <g:row>
29          <g:customCell><g:RadioButton name="radio" value="true"
                text="option␣1" /></g:customCell>
30          <g:customCell><g:RadioButton name="radio" text="option␣2"
                /></g:customCell>
31          <g:customCell><g:TextBox value="Hi" /></g:customCell>
32          <g:customCell><g:RichTextArea width="150px">Hi<br/>I am <u
                >Michael</u></g:RichTextArea></g:customCell>
33          <g:customCell><g:ListBox visibleItemCount="1">
34            <g:item value="1">first item</g:item>
35            <g:item value="2">second item</g:item>
36            <g:item value="3">third item</g:item>
37          </g:ListBox></g:customCell>
38        </g:row>
39      </g:Grid>
40    </g:center>
41    <g:west size="200">
42      <g:HTML styleName="{style.west}">
43        Important links:
44        <ul>
45          <li><a href="http:\\tu-freiberg.de">tu-freiberg.de</a></li
                >
46          <li><a href="http\\google.de">google.de</a></li>
47          <li><a href="javascript:history.back()">back</a></li>
48        </ul>
49      </g:HTML>
50    </g:west>
51 </g:DockLayoutPanel>
52 </ui:UiBinder>
```

Listing A34: UiBinderExample.ui.xml

```
1 package de.tu_freiberg.informatik.vonwenckstern;
2
3 import java.rmi.Remote;
4 import java.rmi.RemoteException;
5
6 public interface RmiServerIntf extends Remote {
7     public String getMessage(String msg) throws RemoteException;
8 }
```

```
1 package de.tu_freiberg.informatik.vonwenckstern;
2
3 import java.rmi.Naming;
4 import java.rmi.RemoteException;
5 import java.rmi.RMISecurityManager;
6 import java.rmi.server.UnicastRemoteObject;
7 import java.rmi.registry.*;
8
9 /**
```

```
10  * create a <code>no.policy</code> file with the following content
        <br>
11  * <code>
12  * grant {<br>
13  *   permission java.security.AllPermission;<br>
14  * };
15  * </code> and then add the following line <br>
16  * <code>-Djava.security.policy=C:\GWT\workspace\DA_RMI\no.policy
        </code><br>
17  * to the VM arguments in the Eclipse's run configuration under
        the arguments tab
18  * <br><br>
19  * <b>server class nearly complete copied from wikipedia:<br>
20  * http://en.wikipedia.org/wiki/Java_remote_method_invocation#
        Example</b>
21  * */
22 public class RmiServer extends UnicastRemoteObject implements
      RmiServerIntf {
23    private static final long serialVersionUID = 1L;
24    public static final String MESSAGE = "Hello world";
25
26   public RmiServer() throws RemoteException {
27   }
28
29   public String getMessage(String msg) {
30     try { Thread.sleep(3000); // I am very busy
31         } catch (InterruptedException e) {}
32     return MESSAGE + "\nYou said: " + msg + "\n";
33   }
34
35   public static void main(String args[]) {
36     System.out.println("RMI server started");
37
38     System.setProperty("java.rmi.server.codebase", "file:/C:/GWT/
         workspace/DA_RMI/bin/");
39
40     // Create and install a security manager
41     if (System.getSecurityManager() == null) {
42       System.setSecurityManager(new RMISecurityManager());
43       System.out.println("Security manager installed.");
44     } else {
45       System.out.println("Security manager already exists.");
46     }
47
48     try { //special exception handler for registry creation
49       LocateRegistry.createRegistry(1099);
50       System.out.println("java RMI registry created.");
51     } catch (RemoteException e) {
52       //do nothing, error means registry already exists
53       System.out.println("java RMI registry already exists.");
54     }
55
56     try {
57       //Instantiate RmiServer
58       RmiServer obj = new RmiServer();
59
```

```
60        // Bind this object instance to the name "RmiServer"
61        Naming.rebind("//localhost/RmiServer", obj);
62
63        System.out.println("PeerServer bound in registry");
64      } catch (Exception e) {
65        System.err.println("RMI server exception:" + e);
66        e.printStackTrace();
67      }
68    }
69 }
```

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3  import java.awt.FlowLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import java.rmi.Naming;
7  import java.rmi.RMISecurityManager;
8
9  import javax.swing.JButton;
10 import javax.swing.JFrame;
11 import javax.swing.JPanel;
12 import javax.swing.JProgressBar;
13 import javax.swing.JTextArea;
14 import javax.swing.JTextField;
15 import javax.swing.Timer;
16
17 /**
18  * create a <code>no.policy</code> file with the following content
        <br>
19  * <code>
20  * grant {<br>
21  *   permission java.security.AllPermission;<br>
22  * };
23  * </code> and then add the following line <br>
24  * <code>-Djava.security.policy=C:\GWT\workspace\DA_RMI\no.policy
        </code><br>
25  * to the VM arguments in the Eclipse's run configuration under
        the arguments tab
26  * */
27 public class RmiClient {
28   // "obj" is the reference of the remote object
29   RmiServerIntf obj = null;
30
31   /** wrapper making synchronous RMI asynchronously */
32   public void getMessage(final String msg, final ActionListener
         listener) {
33     if(listener != null) {
34       Runnable run = new Runnable() {
35         @Override
36         public void run() {
37           String result = getMessage(msg);
38           listener.actionPerformed(new ActionEvent(this, 0, result
                ));
39         }
40       };
41       Thread t = new Thread(run);
```

```
42        t.start();
43      }
44    }
45
46    /** sending a synchronous RMI message to the server */
47    public String getMessage(final String msg) {
48      try {
49        obj = (RmiServerIntf)Naming.lookup("//localhost/RmiServer");
50        return obj.getMessage(msg);
51      } catch (Exception e) {
52        System.err.println("RmiClient exception: " + e);
53        e.printStackTrace();
54
55        return e.getMessage();
56      }
57    }
58
59
60    private JButton sendSync = new JButton("send synchronously");
61    private JButton sendAsync = new JButton("send asynchronously");
62    private JTextField text = new JTextField("Hi I am Michael", 30);
63    private JTextArea  label = new JTextArea("Servers response:\n");
64    private JProgressBar pbar = new JProgressBar(0, 100);
65    public RmiClient() {
66      JFrame frame = new JFrame();
67      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
68      JPanel panel = new JPanel();
69      panel.setLayout(new FlowLayout());
70      frame.add(panel);
71      text.setSize(200, 50);
72      panel.add(text);
73      panel.add(sendSync);
74      panel.add(sendAsync);
75      panel.add(label);
76      label.setWrapStyleWord(true);
77      label.setLineWrap(true);
78      label.setEditable(false);
79      label.setSize(200, 200);
80      sendSync.addActionListener(new ActionListener() {
81        @Override
82        public void actionPerformed(ActionEvent arg0) {
83          String answer = getMessage(text.getText());
84          label.setText(label.getText() + answer);
85        }
86      });
87      sendAsync.addActionListener(new ActionListener() {
88        @Override
89        public void actionPerformed(ActionEvent arg0) {
90          getMessage(text.getText(), new ActionListener() {
91            @Override
92            public void actionPerformed(ActionEvent e) {
93              String answer = e.getActionCommand();
94              label.setText(label.getText() + answer);
95            }
96          });
97        }
```

```
98      });
99      pbar.setValue(0);
100     pbar.setSize(200, 20);
101     panel.add(pbar);
102     Timer t = new Timer(500, new ActionListener() {
103       @Override
104       public void actionPerformed(ActionEvent arg0) {
105         pbar.setValue(pbar.getValue() + 1);
106         if(pbar.getValue() >= 100) {
107           pbar.setValue(0);
108         }
109       }
110     });
111     t.start();
112     frame.setSize(400, 400);
113     frame.setVisible(true);
114   }
115
116   public static void main(String args[]) {
117     // Create and install a security manager
118     if (System.getSecurityManager() == null) {
119       System.setSecurityManager(new RMISecurityManager());
120     }
121     new RmiClient();
122   }
123 }
```

Listing A35: RMI example.
Top: RmiServerIntf.java, Middle: RmiServer.java, Bottom: RmiClient.java

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import java.util.Arrays;
4  import java.util.HashMap;
5  import java.util.List;
6
7  import com.google.gwt.core.client.EntryPoint;
8  import com.google.gwt.event.dom.client.ClickEvent;
9  import com.google.gwt.event.dom.client.ClickHandler;
10 import com.google.gwt.event.logical.shared.ValueChangeEvent;
11 import com.google.gwt.event.logical.shared.ValueChangeHandler;
12 import com.google.gwt.http.client.URL;
13 import com.google.gwt.user.client.History;
14 import com.google.gwt.user.client.Window;
15 import com.google.gwt.user.client.ui.Button;
16 import com.google.gwt.user.client.ui.CheckBox;
17 import com.google.gwt.user.client.ui.FlexTable;
18 import com.google.gwt.user.client.ui.HTML;
19 import com.google.gwt.user.client.ui.HorizontalPanel;
20 import com.google.gwt.user.client.ui.ListBox;
21 import com.google.gwt.user.client.ui.RootPanel;
22 import com.google.gwt.user.client.ui.TextBox;
23 import com.google.gwt.user.client.ui.VerticalPanel;
24
25 public class Survey implements EntryPoint, ClickHandler,
      ValueChangeHandler<String> {
```

```
26   FlexTable page1 = new FlexTable();
27   FlexTable page2 = new FlexTable();
28   FlexTable page3 = new FlexTable();
29   TextBox name = new TextBox();
30   TextBox firstName = new TextBox();
31   ListBox sex = new ListBox();
32   HorizontalPanel hobbies = new HorizontalPanel();
33   TextBox friendsName = new TextBox();
34   ListBox allFriends = new ListBox();
35   FlexTable summery = new FlexTable();
36   HashMap<String, String> actualState = new HashMap<String, String
         >();
37   public void onModuleLoad() {
38     createPage1();
39     createPage2();
40     createPage3();
41
42     RootPanel.get().add(page1);
43     History.addValueChangeHandler(this);
44   }
45
46   // normally you would create the view with an XML file, in order
         to have only one java file we do not do it so in this
        example
47   public void createPage1() {
48     page1.setWidget(0,0,new HTML("<h1>Page 1</h1><h2>general
          information</h2>"));
49     page1.setText(1, 0, "Name:");
50     page1.setWidget(1, 1, name);
51
52     page1.setText(2, 0, "First name:");
53     page1.setWidget(2, 1, firstName);
54
55     page1.setText(3, 0, "Sex:");
56     sex.addItem("female");
57     sex.addItem("male");
58     sex.setVisibleItemCount(2);
59     sex.setSelectedIndex(-1);
60     page1.setWidget(3, 1, sex);
61
62     Button next = new Button("next >>");
63     next.setTitle("next1");
64     next.addClickHandler(this);
65     page1.setWidget(4, 1, next);
66   }
67
68   public void createPage2() {
69     page2.setWidget(0,0,new HTML("<h1>Page 2</h1><h2>hobbies and
          friends</h2>"));
70     page2.setText(1, 0, "Hobbies:");
71     String[] sHobbies = new String[] {"soccer","tennis","
          basketball","baseball","volleyball","football"};
72     for(String sHobby : sHobbies) {
73       CheckBox check = new CheckBox(sHobby);
74       hobbies.add(check);
75     }
```

```
76      page2.setWidget(1, 1, hobbies);
77
78      page2.setText(2, 0, "Friends:");
79      VerticalPanel friends = new VerticalPanel();
80      HorizontalPanel addName = new HorizontalPanel();
81      Button add = new Button("add");
82      add.setTitle("add");
83      add.addClickHandler(this);
84      addName.add(friendsName);
85      addName.add(add);
86      allFriends.setVisibleItemCount(10);
87      Button delete = new Button("delete");
88      delete.setTitle("delete");
89      delete.addClickHandler(this);
90      friends.add(addName);
91      friends.add(allFriends);
92      friends.add(delete);
93      page2.setWidget(2, 1, friends);
94
95      Button prev = new Button("<< previous");
96      prev.setTitle("prev2");
97      prev.addClickHandler(this);
98      page2.setWidget(3, 0, prev);
99
100     Button next = new Button("next >>");
101     next.setTitle("next2");
102     next.addClickHandler(this);
103     page2.setWidget(3, 1, next);
104
105   }
106
107   public void createPage3() {
108     page3.setWidget(0,0,new HTML("<h1>Page 3</h1><h2>Summery of
          your input data</h2>"));
109     page3.setWidget(1, 0, summery);
110
111     Button prev = new Button("<< previous");
112     prev.setTitle("prev3");
113     prev.addClickHandler(this);
114     page3.setWidget(3, 0, prev);
115   }
116
117   @Override
118   public void onClick(ClickEvent event) {
119     String title = ((Button) event.getSource()).getTitle();
120     if(title.equals("add")) {
121       if(!friendsName.getValue().trim().isEmpty()) {
122         allFriends.addItem(friendsName.getValue().trim());
123         friendsName.setValue("");
124       }
125     } else if(title.equals("delete")) {
126       if(allFriends.getSelectedIndex() != -1) {
127         allFriends.removeItem(allFriends.getSelectedIndex());
128       }
129     } else if(title.equals("next1")) {
130       String n = name.getValue().trim();
```

```
131         String fn = firstName.getValue().trim();
132         if(n.isEmpty() || fn.isEmpty()  || sex.getSelectedIndex() ==
                -1) {
133           Window.alert("You did not insert all necessary data!");
134         } else {
135           actualState.put("page", "page2");
136           actualState.put("name", n);
137           actualState.put("firstname", fn);
138           actualState.put("sex", sex.getValue(sex.getSelectedIndex()
                ));
139           setHistoryURL();
140         }
141       } else if(title.equals("next2") || title.equals("prev2")) {
142         String hobbies = "";
143         for(int i=0; i<this.hobbies.getWidgetCount(); i++) {
144           if(this.hobbies.getWidget(i) instanceof CheckBox) {
145             CheckBox check = (CheckBox)this.hobbies.getWidget(i);
146             if(check.getValue()) {
147               hobbies += check.getText() + ",";
148             }
149           }
150         }
151         if(!hobbies.isEmpty()) {
152           hobbies = hobbies.substring(0, hobbies.length()-1);
153         }
154         actualState.put("hobbies", hobbies);
155         String friends = "";
156         for(int i=0; i<allFriends.getItemCount(); i++) {
157           friends += allFriends.getValue(i) + ",";
158         }
159         if(!friends.isEmpty()) {
160           friends = friends.substring(0, friends.length()-1);
161         }
162         actualState.put("friends", friends);
163         if(title.equals("next2")) {
164           actualState.put("page", "page3");
165         } else {
166           actualState.put("page", "page1");
167         }
168         setHistoryURL();
169       } else if(title.equals("prev3")) {
170         actualState.put("page", "page2");
171         setHistoryURL();
172       }
173     }
174
175     public void setHistoryURL() {
176       String state = "";
177       for(String key : actualState.keySet()) {
178         state += key + "=" + actualState.get(key) + ";";
179       }
180       History.newItem(URL.encode(state));
181     }
182
183     @Override
184     public void onValueChange(ValueChangeEvent<String> event) {
```

```
185       String state = URL.decode(event.getValue());
186       if(state == null || state.length() == 0) {
187         History.newItem(URL.encode("page=page1"));
188       }
189       String keyValues[] = state.split(";");
190       for(String keyValue : keyValues) {
191         String key = keyValue.split("=")[0];
192         String value = keyValue.split("=")[1];
193         actualState.put(key, value);
194       }
195       RootPanel.get().clear();
196       if(actualState.get("page").equals("page1")) {
197         firstName.setValue(actualState.get("firstname"));
198         name.setValue(actualState.get("name"));
199         if(actualState.containsKey("sex")) {
200           sex.setSelectedIndex(actualState.get("sex").equals("female
                ") ? 0 : 1);
201         } else {
202           sex.setSelectedIndex(-1);
203         }
204         RootPanel.get().add(page1);
205       } else if(actualState.get("page").equals("page2")) {
206         if(actualState.containsKey("hobbies")) {
207           List<String> hobbies = Arrays.asList(actualState.get("
                hobbies").split(","));
208           for(int i=0; i<this.hobbies.getWidgetCount(); i++) {
209             if(this.hobbies.getWidget(i) instanceof CheckBox) {
210               CheckBox check = (CheckBox)this.hobbies.getWidget(i);
211               check.setValue(hobbies.contains(check.getText()));
212             }
213           }
214         }
215         allFriends.clear();
216         if(actualState.containsKey("friends")) {
217           String[] friends = actualState.get("friends").split(",");
218           for(String friend: friends) {
219             allFriends.addItem(friend);
220           }
221         }
222         RootPanel.get().add(page2);
223       } else if(actualState.get("page").equals("page3")) {
224         summery.clear();
225         int i=0;
226         for(String key : actualState.keySet()) {
227           summery.setText(i, 0, key);
228           summery.setText(i, 1, actualState.get(key));
229           i++;
230         }
231         RootPanel.get().add(page3);
232       }
233     }
234 }
```

Listing A36: Source code of Survey.java

```
1
2  package de.tu_freiberg.informatik.vonwenckstern;
3
4  import com.google.gwt.core.client.EntryPoint;
5  import com.google.gwt.core.shared.GWT;
6  import com.google.gwt.event.dom.client.ClickEvent;
7  import com.google.gwt.event.dom.client.ClickHandler;
8  import com.google.gwt.event.logical.shared.ValueChangeEvent;
9  import com.google.gwt.event.logical.shared.ValueChangeHandler;
10 import com.google.gwt.user.client.History;
11 import com.google.gwt.user.client.Window;
12 import com.google.gwt.user.client.Window.Location;
13 import com.google.gwt.user.client.ui.Button;
14 import com.google.gwt.user.client.ui.Label;
15 import com.google.gwt.user.client.ui.RootPanel;
16 import com.google.gwt.user.client.ui.TextBox;
17
18 public class URLlength implements EntryPoint, ValueChangeHandler<
       String> {
19   private Button btnGenerateURL = new Button("generate␣URL␣and␣
       test");
20   private TextBox tbox = new TextBox();
21   private static final String referenceString = "1
       a2a3a4a5a6a7a8a9a"; // length 18, sum is 45
22   private int sum;
23   public void onModuleLoad() {
24     RootPanel rootPanel = RootPanel.get();
25     rootPanel.add(new Label("URL␣length"));
26     tbox.setValue("180");
27     rootPanel.add(tbox);
28     rootPanel.add(btnGenerateURL);
29     History.addValueChangeHandler(this);
30     btnGenerateURL.addClickHandler(new ClickHandler() {
31       @Override
32       public void onClick(ClickEvent event) {
33         int length = 0;
34         try {
35           length = Integer.parseInt(tbox.getValue());
36           if(length < 0) throw new NumberFormatException();
37         } catch(NumberFormatException e) { Window.alert("the␣
             textbox␣should␣contain␣a␣positive␣integer␣number"); }
38         String _url = Location.getHref();
39         int hostLength = _url.contains("#") ? _url.indexOf("#") :
             _url.length();
40         int loopIt = (length - hostLength) / 18 + 1;
41         String url = "";
42         for(int i=0; i<loopIt; i++) {
43           url += referenceString;
44         }
45         sum = loopIt * 45;
46         History.newItem(url);
47       }
48     });
49   }
50   @Override
```

```
51   public void onValueChange(ValueChangeEvent<String> event) {
52     String url = event.getValue();
53     String pattern = GWT.isScript() ? "^[\\da]+$" : "[\\da]+";
54     if(url != null && !url.isEmpty() && url.matches(pattern)) {
55       String[] s = url.split("a");
56       int rSum = 0;
57       for(String nb : s) {
58         rSum += Integer.parseInt(nb);
59       }
60       if(rSum == sum) {
61         Window.alert("The browser could successful decode the " +
               Location.getHref().length() + " character long URL.");
62       } else {
63         Window.alert("The URL is too long and could not get
               successful decoded.");
64       }
65     }
66   }
67 }
```

Listing A37: Source code of URLlength class

A 3.1 Source code of the Agricola board game

de.tu_freiberg.informatik.vonwenckstern.client package

```
1 package de.tu_freiberg.informatik.vonwenckstern.client;
2
3 import com.google.gwt.core.client.EntryPoint;
4 import com.google.gwt.user.client.ui.RootPanel;
5
6 import de.tu_freiberg.informatik.vonwenckstern.client.model.Player;
7 import de.tu_freiberg.informatik.vonwenckstern.client.view.ViewFactory;
8
9 public class Agricola implements EntryPoint {
10   public void onModuleLoad() {
11     RootPanel.get().setPixelSize(1920, 950);
12     AppController app = new AppController(ViewFactory.Util.getViewFactory
           ().getAppView(), Player.BLUE);
13     RootPanel.get().add(app.getView());
14   }
15 }
```

Listing A38: Agricola.java file

```
71 public class AppController implements Presenter, AddResourceHandler,
       BuildHouseHandler, PlayerFieldDoneHandler, PlowFieldHandler,
72   BuildFenceHandler, FamilyAdditionHandler, RestaurateHandler, SeedHandler
         , GetSheepHandler, GetBoarHandler, GetCowHandler,
         PlowFieldSeedHandler,
73   FamilyAdditionWithoutHouseHandler, RestaurateAndFenceHandler,
         NextRoundHandler, GetBigAcquisitionHandler,
         EnableBigAcquisitionHandler {
74
75   public interface Display {
76     public void setLeftWidget(Widget w);
```

```
77      public void setMiddleWidget(Widget w);
78      public void setTopWidget(Widget w);
79      public void setBottomWidget(Widget w);
80      public void setRightWidget(Widget w);
81      public void setInfoWidget(Widget w);
82      public void setForcePlayerField(boolean b);
83      public void setForceBigAcquisitationField(boolean b);
84      public boolean isBigAcquisitationFieldEnabled();
85      public Widget asWidget();
86    }
87
88    private Display display = null;
89    private Presenter leftPresenter = null;
90    private Presenter infoPresenter = null;
91    private Presenter middlePresenter = null;
92    private Presenter bottomPresenter = null;
93    private Presenter rightPresenter = null;
94    private Presenter topPresenter = null;
95
96    private Player player = Player.BLUE;
97    private PlayerFieldModel playerModel = new PlayerFieldModel(player);
98    private PlayerResourceModel resourceModel = new PlayerResourceModel();
99
100   public AppController(Display display, Player player) {
101     this.display = display;
102     this.player = player;
103     ViewFactory vf = ViewFactory.Util.getViewFactory();
104     leftPresenter = new CardFieldPresenter(vf.getCardFieldView(), player,
          new CardFieldModel());
105     middlePresenter = new Rounds1To7Presenter(vf.getRounds1To7View(),
          player, new Rounds1To7Model());
106     infoPresenter = new InfoViewPresenter(vf.getInfoView(), resourceModel)
          ;
107     rightPresenter = new PlayerFieldPresenter(vf.getPlayerFieldView(),
          playerModel, resourceModel);
108     bottomPresenter = new Rounds8To14Presenter(vf.getRounds8To14View(),
          player, new Rounds8To14Model());
109     topPresenter = new BigAcquisitionsPresenter(vf.getAcquisitionsView(),
          new BigAcquisitationsModel());
110     updateView();
111     bind();
112   }
113
114   private void bind() {
115     EventBus.getEventBus().addAddResourceHandler(this);
116     EventBus.getEventBus().addBuildHouseHandler(this);
117     EventBus.getEventBus().addPlayerFieldDoneHandler(this);
118     EventBus.getEventBus().addPlowFieldHandler(this);
119     EventBus.getEventBus().addBuildFenceHandler(this);
120     EventBus.getEventBus().addFamilyAdditionHandler(this);
121     EventBus.getEventBus().addRestaurateHandler(this);
122     EventBus.getEventBus().addSeedHandler(this);
123     EventBus.getEventBus().addGetSheepHandler(this);
124     EventBus.getEventBus().addGetBoarHandler(this);
125     EventBus.getEventBus().addGetCowHandler(this);
126     EventBus.getEventBus().addPlowFieldSeedHandler(this);
127     EventBus.getEventBus().addFamilyAdditionWithoutHouseHandler(this);
128     EventBus.getEventBus().addRestaurateAndFenceHandler(this);
129     EventBus.getEventBus().addNextRoundHandler(this);
130     EventBus.getEventBus().addGetBigAcquisitionHandler(this);
131     EventBus.getEventBus().addEnableBigAcquisitionHandler(this);
```

```
132   }
133
134   private void updateView() {
135     if(leftPresenter != null) {
136       display.setLeftWidget(leftPresenter.getView());
137     }
138     if(middlePresenter != null) {
139       display.setMiddleWidget(middlePresenter.getView());
140     }
141     if(infoPresenter != null) {
142       display.setInfoWidget(infoPresenter.getView());
143     }
144     if(bottomPresenter != null) {
145       display.setBottomWidget(bottomPresenter.getView());
146     }
147     if(rightPresenter != null) {
148       display.setRightWidget(rightPresenter.getView());
149     }
150     if(topPresenter != null) {
151       display.setTopWidget(topPresenter.getView());
152     }
153   }
154
155   @Override
156   public Widget getView() {
157     return display.asWidget();
158   }
159
160   @Override
161   public void onAddResource(AddResourceEvent event) {
162     if(infoPresenter instanceof InfoViewPresenter) {
163       for(RessourceItem item : event.getItems()) {
164         resourceModel.addRessource(item.res, item.resCount);
165       }
166     }
167   }
168
169   @Override
170   public void onBuildHouse(BuildHouseEvent event) {
171     if(rightPresenter instanceof PlayerFieldPresenter) {
172       final DialogBox dlg = new DialogBox(false, true);
173       Grid grid = new Grid(4, 2);
174       grid.setHTML(0, 0, "Make your choice:");
175       grid.setHTML(1, 0, "Do you want to build a house?");
176       final CheckBox buildHouse = new CheckBox("house");
177       buildHouse.setValue(true);
178       grid.setWidget(1, 1, buildHouse);
179       grid.setHTML(2, 0, "How many stables do you want?");
180       final IntegerBox stables = new IntegerBox();
181       stables.setValue(0);
182       grid.setWidget(2, 1, stables);
183       Button ok = new Button("OK");
184       ok.addClickHandler(new ClickHandler() {
185         @Override
186         public void onClick(ClickEvent event) {
187           dlg.hide();
188           EventBus.fire(new ShowingDialogEvent(false));
189           display.setForcePlayerField(true);
190           if(buildHouse.getValue()) {
191             ((PlayerFieldPresenter) rightPresenter).addState(State.
                  BUILD_HOUSE);
```

```
192             }
193             for(int i=0; i<stables.getValue() && i<10; i++) {
194               ((PlayerFieldPresenter) rightPresenter).addState(State.
                      BUILD_STABLE);
195             }
196             ((PlayerFieldPresenter) rightPresenter).executeStates();
197           }
198         });
199         grid.setWidget(3, 1, ok);
200         dlg.setWidget(grid);
201         EventBus.fire(new ShowingDialogEvent(true));
202         dlg.show();
203       }
204     }
205
206     @Override
207     public void onPlayerFieldDone(PlayerFieldDoneEvent event) {
208       display.setForcePlayerField(false);
209     }
210
211     @Override
212     public void onPlowField(PlowFieldEvent event) {
213       display.setForcePlayerField(true);
214       ((PlayerFieldPresenter) rightPresenter).addState(State.PLOW_FIELD);
215       ((PlayerFieldPresenter) rightPresenter).executeStates();
216     }
217
218     @Override
219     public void onSeeding(SeedEvent event) {
220       if(rightPresenter instanceof PlayerFieldPresenter) {
221         final DialogBox dlg = new DialogBox(false, true);
222         Grid grid = new Grid(5, 2);
223         grid.setHTML(0, 0, "Make␣your␣choice:");
224         grid.setHTML(1, 0, "Do␣you␣want␣to␣back␣bread?");
225         final CheckBox backBread = new CheckBox("back␣bread");
226         backBread.setValue(false);
227         grid.setWidget(1, 1, backBread);
228         grid.setHTML(2, 0, "How␣many␣<b>grain</b>␣fields␣should␣get␣seeded?"
              );
229         final IntegerBox grains = new IntegerBox();
230         grains.setValue(1);
231         grid.setWidget(2, 1, grains);
232         grid.setHTML(3, 0, "How␣many␣<b>vegetable</b>␣fields␣should␣get␣
              seeded?");
233         final IntegerBox vegetables = new IntegerBox();
234         vegetables.setValue(0);
235         grid.setWidget(3, 1, vegetables);
236         Button ok = new Button("OK");
237         ok.addClickHandler(new ClickHandler() {
238           @Override
239           public void onClick(ClickEvent event) {
240             dlg.hide();
241             EventBus.fire(new ShowingDialogEvent(false));
242             display.setForcePlayerField(true);
243             if(backBread.getValue()) {
244               ((PlayerFieldPresenter) rightPresenter).addState(State.
                      BACK_BREAD);
245             }
246             for(int i=0; i<grains.getValue() && i<10; i++) {
247               ((PlayerFieldPresenter) rightPresenter).addState(State.
                      SEED_GRAIN);
```

```
248             }
249             for(int i=0; i<vegetables.getValue() && i<10; i++) {
250               ((PlayerFieldPresenter) rightPresenter).addState(State.
                      SEED_VEGETABLE);
251             }
252             ((PlayerFieldPresenter) rightPresenter).executeStates();
253           }
254         });
255         grid.setWidget(4, 1, ok);
256         dlg.setWidget(grid);
257         EventBus.fire(new ShowingDialogEvent(true));
258         dlg.show();
259       }
260     }
261
262     @Override
263     public void onRestaurate(RestaurateEvent event) {
264       ((PlayerFieldPresenter) rightPresenter).addState(State.RESTAURATE);
265       if(Window.confirm("Do␣you␣want␣to␣make␣a␣big␣acquisitation␣after␣
              restaurating␣your␣rooms?")) {
266         onEnablingBigAcquisition(null);
267       }
268       display.setForcePlayerField(true);
269       ((PlayerFieldPresenter) rightPresenter).executeStates();
270     }
271
272     @Override
273     public void onFamilyAddition(FamilyAdditionEvent event) {
274       display.setForcePlayerField(true);
275       ((PlayerFieldPresenter) rightPresenter).addState(State.FAMILY_ADDITION
              );
276       ((PlayerFieldPresenter) rightPresenter).executeStates();
277     }
278
279     @Override
280     public void onBuildFence(BuildFenceEvent event) {
281       String res = Window.prompt("How␣many␣fields␣do␣you␣want␣to␣fence?", "1
              ");
282       try {
283         int amount = Integer.parseInt(res);
284         for(int i=0; i<amount && i<20; i++) {
285           ((PlayerFieldPresenter) rightPresenter).addState(State.BUILD_FENCE
                  );
286         }
287         display.setForcePlayerField(true);
288         ((PlayerFieldPresenter) rightPresenter).executeStates();
289       } catch(Exception e) {
290         Window.alert("You␣have␣to␣enter␣a␣number");
291         onBuildFence(event);
292       }
293     }
294
295     @Override
296     public void onGettingSheep(GetSheepEvent event) {
297       for(int i=0; i<event.getSheepCount(); i++) {
298         ((PlayerFieldPresenter) rightPresenter).addState(State.GET_SHEEP);
299       }
300       display.setForcePlayerField(true);
301       ((PlayerFieldPresenter) rightPresenter).executeStates();
302     }
303
```

```
304   @Override
305   public void onRestaurateAndBuildFence(RestaurateAndFenceEvent event) {
306     ((PlayerFieldPresenter) rightPresenter).addState(State.RESTAURATE);
307     String res = Window.prompt("How many fields do you want to fence after
          restaurating your rooms?", "0");
308     try {
309       int amount = Integer.parseInt(res);
310       for(int i=0; i<amount && i<20; i++) {
311         ((PlayerFieldPresenter) rightPresenter).addState(State.BUILD_FENCE
              );
312       }
313       display.setForcePlayerField(true);
314       ((PlayerFieldPresenter) rightPresenter).executeStates();
315     } catch(Exception e) {
316       Window.alert("You did not enter a number, so you will build no fence
            .");
317     }
318     display.setForcePlayerField(true);
319     ((PlayerFieldPresenter) rightPresenter).executeStates();
320   }
321
322   @Override
323   public void onFamilyAdditionWithoutHouse(FamilyAdditionWithoutHouseEvent
          event) {
324     ((PlayerFieldPresenter) rightPresenter).addState(State.
          FAMILIY_ADDITION_NO_HOUSE);
325     display.setForcePlayerField(true);
326     ((PlayerFieldPresenter) rightPresenter).executeStates();
327   }
328
329   @Override
330   public void onPlowFieldAndSeed(PlowFieldSeedEvent event) {
331     if(rightPresenter instanceof PlayerFieldPresenter) {
332       final DialogBox dlg = new DialogBox(false, true);
333       Grid grid = new Grid(5, 2);
334       grid.setHTML(0, 0, "Make your choice:");
335       grid.setHTML(1, 0, "Do you want to plow one field?");
336       final CheckBox plowField = new CheckBox("plow field");
337       plowField.setValue(true);
338       grid.setWidget(1, 1, plowField);
339       grid.setHTML(2, 0, "How many <b>grain </b> fields should get seeded?"
            );
340       final IntegerBox grains = new IntegerBox();
341       grains.setValue(0);
342       grid.setWidget(2, 1, grains);
343       grid.setHTML(3, 0, "How many <b>vegetable </b> fields should get
            seeded?");
344       final IntegerBox vegetables = new IntegerBox();
345       vegetables.setValue(0);
346       grid.setWidget(3, 1, vegetables);
347       Button ok = new Button("OK");
348       ok.addClickHandler(new ClickHandler() {
349         @Override
350         public void onClick(ClickEvent event) {
351           dlg.hide();
352           EventBus.fire(new ShowingDialogEvent(false));
353           display.setForcePlayerField(true);
354           if(plowField.getValue()) {
355             ((PlayerFieldPresenter) rightPresenter).addState(State.
                  PLOW_FIELD);
356           }
```

```
357           for(int i=0; i<grains.getValue() && i<10; i++) {
358             ((PlayerFieldPresenter) rightPresenter).addState(State.
                  SEED_GRAIN);
359           }
360           for(int i=0; i<vegetables.getValue() && i<10; i++) {
361             ((PlayerFieldPresenter) rightPresenter).addState(State.
                  SEED_VEGETABLE);
362           }
363           ((PlayerFieldPresenter) rightPresenter).executeStates();
364         }
365       });
366       grid.setWidget(4, 1, ok);
367       dlg.setWidget(grid);
368       EventBus.fire(new ShowingDialogEvent(true));
369       dlg.show();
370     }
371   }
372
373   @Override
374   public void onGettingCow(GetCowEvent event) {
375     for(int i=0; i<event.getCowCount(); i++) {
376       ((PlayerFieldPresenter) rightPresenter).addState(State.GET_COW);
377     }
378     display.setForcePlayerField(true);
379     ((PlayerFieldPresenter) rightPresenter).executeStates();
380   }
381
382   @Override
383   public void onGettingBoar(GetBoarEvent event) {
384     for(int i=0; i<event.getBoarCount(); i++) {
385       ((PlayerFieldPresenter) rightPresenter).addState(State.GET_BOAR);
386     }
387     display.setForcePlayerField(true);
388     ((PlayerFieldPresenter) rightPresenter).executeStates();
389   }
390
391   @Override
392   public void onNextRound(NextRoundEvent event) {
393     int rnd = event.getRound();
394     System.out.println("round:␣" + rnd);
395     if(rnd > 14) {
396       Grid g = new Grid(16, 2);
397       g.setHTML(0, 0, "Fields:␣");
398       int pointsFields = playerModel.countFields() - 1;
399       if(pointsFields == 0) pointsFields = -1;
400       if(pointsFields > 4) pointsFields = 4;
401       g.setHTML(0, 1, pointsFields+"");
402
403       g.setHTML(1, 0, "pastures:␣");
404       int pointsPastures = playerModel.countPastures();
405       if(pointsPastures == 0) pointsPastures = -1;
406       if(pointsPastures > 4) pointsPastures = 4;
407       g.setHTML(1, 1, pointsPastures+"");
408
409       g.setHTML(2, 0, "Grains:␣");
410       int grains = resourceModel.getGrainCount() + playerModel.
              countGrainsOnFields();
411       int pointsGrains = grains < 1 ? -1 : grains < 4 ? 1 : grains < 6 ? 2
              : grains < 8 ? 3 : 4;
412       g.setHTML(2, 1, pointsGrains+"");
413
```

```
414         g.setHTML(3, 0, "Vegetables: ");
415         int pointsVegetables = resourceModel.getVegetableCount() +
                playerModel.countVegetablesOnFields();
416         if(pointsVegetables == 0) pointsVegetables = -1;
417         if(pointsVegetables > 4) pointsVegetables = 4;
418         g.setHTML(3, 1, pointsVegetables+"");
419
420         g.setHTML(4, 0, "Sheep: ");
421         int sheep = playerModel.countSheep();
422         int pointsSheep = sheep < 1 ? -1 : sheep < 4 ? 1 : sheep < 6 ? 2 :
                sheep < 8 ? 3 : 4;
423         g.setHTML(4, 1, pointsSheep+"");
424
425         g.setHTML(5, 0, "Boars: ");
426         int boars = playerModel.countBoars();
427         int pointsBoars = boars < 1 ? -1 : boars < 3 ? 1 : boars < 5 ? 2 :
                boars < 7 ? 3 : 4;
428         g.setHTML(5, 1, pointsBoars+"");
429
430         g.setHTML(6, 0, "Cows: ");
431         int cows = playerModel.countCows();
432         int pointsCows = cows < 1 ? -1 : cows < 2 ? 1 : cows < 4 ? 2 : cows
                < 6 ? 3 : 4;
433         g.setHTML(6, 1, pointsCows+"");
434
435         g.setHTML(7, 0, "Unused fields: ");
436         int pointsUnusedFields = -1*playerModel.countUnusedFields();
437         g.setHTML(7, 1, pointsUnusedFields+"");
438
439         g.setHTML(8, 0, "Fenced stables: ");
440         int pointsFencedStables = playerModel.countFencedStables();
441         g.setHTML(8, 1, pointsFencedStables+"");
442
443         g.setHTML(9, 0, "Clay houses: ");
444         int pointsClayHouses = playerModel.countClayHouses();
445         g.setHTML(9, 1, pointsClayHouses+"");
446
447         g.setHTML(10, 0, "Stone houses: ");
448         int pointsStoneHouses = 2*playerModel.countStoneHouses();
449         g.setHTML(10, 1, pointsStoneHouses+"");
450
451         g.setHTML(11, 0, "Family members: ");
452         int pointsFamily = 3*playerModel.countPersons();
453         g.setHTML(11, 1, pointsFamily+"");
454
455         g.setHTML(12, 0, "Card points: ");
456         int pointsCards = playerModel.countCardPoints();
457         g.setHTML(12, 1, pointsCards+"");
458
459         int pointsExtra = 0;
460         for(int i=0; i<10; i++) {
461           AcquisitionCardModel a = playerModel.getAcquisitation(i);
462           if(a == null)
463             break;
464           BigAcquisitions ba = a.getAcquisition();
465           if(ba == BigAcquisitions.BA_POTTERY) {
466             int c = resourceModel.getClayCount();
467             pointsExtra += c>6 ? 3 : c>4 ? 2 : c>2 ? 1 : 0;
468           } else if(ba == BigAcquisitions.BA_BASKET_MAKER) {
469             int r = resourceModel.getReedCount();
470             pointsExtra += r>4 ? 3 : r>3 ? 2 : r>1 ? 1 : 0;
```

```
471         } else if(ba == BigAcquisitions.BA_JOINERY) {
472           int w = resourceModel.getWoodCount();
473           pointsExtra += w>6 ? 3 : w>4 ? 2 : w>2 ? 1 : 0;
474         }
475       }
476       g.setHTML(13, 0, "Extra points: ");
477       g.setHTML(13, 1, pointsExtra+"");
478
479       g.setHTML(14, 0, "Beggar points: ");
480       int pointsBeggar = -3*resourceModel.getBeggerCards();
481       g.setHTML(14, 1, pointsBeggar+"");
482
483       g.setHTML(15, 0, "Total number of points: ");
484       int total = pointsFields + pointsPastures + pointsGrains +
              pointsVegetables + pointsSheep + pointsBoars + pointsCows +
              pointsUnusedFields +
485           pointsFencedStables + pointsClayHouses + pointsStoneHouses +
                  pointsFamily + pointsCards + pointsExtra + pointsBeggar;
486       g.setHTML(15, 1, "<b>"+total+"</b>");
487       DialogBox dlg = new DialogBox(false, true);
488       VerticalPanel panel = new VerticalPanel();
489       panel.add(new HTML("<h1>Your score: </h1>"));
490       panel.add(g);
491       Button btn = new Button("restart game");
492       panel.add(btn);
493       btn.addClickHandler(new ClickHandler() {
494         @Override
495         public void onClick(ClickEvent event) {
496           Window.Location.reload();
497         }
498       });
499       dlg.setWidget(panel);
500       dlg.show();
501     }
502   }
503
504   @Override
505   public void onGetBigAcquisition(GetBigAcquisitionEvent event) {
506     if(!display.isBigAcquisitationFieldEnabled()) {
507       Window.alert("The big acquisitation field is disabled.\nEnable this
              field by using the field card 1 big acquisitation - it is coming
              in round 1,2,3 or 4.");
508       return;
509     }
510     BigAcquisitions ba = event.getAcquisition().getAcquisition();
511     if(
512         (ba == BigAcquisitions.BA_FIRE_PLACE) && (resourceModel.
                getClayCount() < 2) ||
513         (ba == BigAcquisitions.BA_FIRE_PLACE2) && (resourceModel.
                getClayCount() < 3) ||
514         (ba == BigAcquisitions.BA_COOKERY) && (resourceModel.getClayCount
                () < 4) ||
515         (ba == BigAcquisitions.BA_COOKERY2) && (resourceModel.getClayCount
                () < 5) ||
516         (ba == BigAcquisitions.BA_FOUNTAIN) && (resourceModel.getWoodCount
                () < 1 || resourceModel.getStoneCount() < 3) ||
517
518         (ba == BigAcquisitions.BA_CLAY_OVEN) && (resourceModel.
                getClayCount() < 3 || resourceModel.getStoneCount() < 1) ||
519         (ba == BigAcquisitions.BA_STONE_OVEN) && (resourceModel.
                getClayCount() < 1 || resourceModel.getStoneCount() < 3) ||
```

```
520         (ba == BigAcquisitions.BA_JOINERY) && (resourceModel.getStoneCount
                () < 2 || resourceModel.getStoneCount() < 2) ||
521         (ba == BigAcquisitions.BA_POTTERY) && (resourceModel.getClayCount
                () < 2 || resourceModel.getStoneCount() < 2) ||
522         (ba == BigAcquisitions.BA_BASKET_MAKER) && (resourceModel.
                getReedCount() < 2 || resourceModel.getStoneCount() < 2)
523       ) {
524         Window.alert("You␣have␣not␣enough␣resources␣to␣buy␣the␣big␣
              acquisitation.\nYour␣turn␣is␣over␣now.");
525       } else {
526         switch(ba) {
527         case BA_FIRE_PLACE: resourceModel.addRessource(Resource.R_CLAY, -2);
               break;
528         case BA_FIRE_PLACE2: resourceModel.addRessource(Resource.R_CLAY, -3)
              ; break;
529         case BA_COOKERY: resourceModel.addRessource(Resource.R_CLAY, -4);
              break;
530         case BA_COOKERY2: resourceModel.addRessource(Resource.R_CLAY, -5);
              break;
531         case BA_FOUNTAIN: resourceModel.addRessource(Resource.R_WOOD, -1);
              resourceModel.addRessource(Resource.R_STONE, -3); resourceModel.
              addRessource(Resource.R_FOOD, 5); break;
532
533         case BA_CLAY_OVEN: resourceModel.addRessource(Resource.R_CLAY, -3);
              resourceModel.addRessource(Resource.R_STONE, -1); break;
534         case BA_STONE_OVEN: resourceModel.addRessource(Resource.R_CLAY, -1);
               resourceModel.addRessource(Resource.R_STONE, -3); break;
535         case BA_JOINERY: resourceModel.addRessource(Resource.R_WOOD, -2);
              resourceModel.addRessource(Resource.R_STONE, -2); break;
536         case BA_POTTERY: resourceModel.addRessource(Resource.R_CLAY, -2);
              resourceModel.addRessource(Resource.R_STONE, -2); break;
537         case BA_BASKET_MAKER: resourceModel.addRessource(Resource.R_REED,
              -2); resourceModel.addRessource(Resource.R_STONE, -2); break;
538         case BA_NONE: break;
539         }
540         ((PlayerFieldPresenter) rightPresenter).addBigAcquisitation(event.
              getAcquisition());
541         ((BigAcquisitionsPresenter) topPresenter).hideAcquisition(ba);
542       }
543       display.setForceBigAcquisitationField(false);
544       ((PlayerFieldPresenter) rightPresenter).setGettingBigAcquisitation(
              false);
545     }
546
547     @Override
548     public void onEnablingBigAcquisition(EnableBigAcquisitionEvent event) {
549       ((PlayerFieldPresenter) rightPresenter).setGettingBigAcquisitation(
              true);
550       display.setForceBigAcquisitationField(true);
551     }
552 }
```

Listing A39: AppController.java file (imports omitted)

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.uibinder.client.UiBinder;
5  import com.google.gwt.uibinder.client.UiField;
6  import com.google.gwt.user.client.ui.Composite;
7  import com.google.gwt.user.client.ui.Label;
8  import com.google.gwt.user.client.ui.SimplePanel;
9  import com.google.gwt.user.client.ui.Widget;
10
11 public class AppView extends Composite implements AppController.Display {
12
13   private static final Binder binder = GWT.create(Binder.class);
14   @UiField
15   SimplePanel leftLayout;
16   @UiField
17   SimplePanel middleLayout;
18   @UiField
19   SimplePanel topLayout;
20   @UiField
21   SimplePanel bottomLayout;
22   @UiField
23   SimplePanel rightLayout;
24   @UiField
25   SimplePanel infoLayout;
26   @UiField
27   Label disabledLabel;
28   @UiField
29   Label disabledLabel2;
30   @UiField
31   Label disabledLabel3;
32   @UiField
33   Label disabledLabel4;
34
35   interface Binder extends UiBinder<Widget, AppView> {
36   }
37
38   public AppView() {
39     initWidget(binder.createAndBindUi(this));
40   }
41
42   @Override
43   public void setLeftWidget(Widget w) {
44     leftLayout.setWidget(w);
45   }
46
47   @Override
48   public void setMiddleWidget(Widget w) {
49     middleLayout.setWidget(w);
50   }
51
52   @Override
53   public void setTopWidget(Widget w) {
54     topLayout.setWidget(w);
55   }
56
57   @Override
58   public void setBottomWidget(Widget w) {
59     bottomLayout.setWidget(w);
60   }
61
```

```
62    @Override
63    public void setRightWidget(Widget w) {
64      rightLayout.setWidget(w);
65    }
66
67    @Override
68    public void setInfoWidget(Widget w) {
69      infoLayout.setWidget(w);
70    }
71
72    @Override
73    public void setForcePlayerField(boolean b) {
74      disabledLabel.setVisible(b);
75      disabledLabel2.setVisible(b);
76    }
77
78    @Override
79    public void setForceBigAcquisitationField(boolean b) {
80      disabledLabel.setVisible(b);
81      disabledLabel2.setVisible(b);
82      disabledLabel4.setVisible(b);
83      disabledLabel3.setVisible(!b);
84    }
85
86    @Override
87    public boolean isBigAcquisitationFieldEnabled() {
88      return !disabledLabel3.isVisible();
89    }
90
91  }
```

Listing A40: AppView.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='
       urn:import:com.google.gwt.user.client.ui'>
4    <ui:style>
5      .disabled {
6        background-color: rgba(0,0,0,0.6);
7        z-index: 101;
8        }
9        .disabled2 {
10       background-color: rgba(0,0,0,0.6);
11       z-index: 50;
12       pointer-events: none
13       }
14       .info {
15       color: darkgray;
16       font-size: medium;
17     }
18     .info a:link {color:darkgray; text-decoration: none; font-style:
           italic; }      /* unvisited link */
19      .info a:visited {color:darkgray; text-decoration: none; font-style:
           italic; }  /* visited link */
20      .info a:hover {color:darkgray; text-decoration: none; font-style:
           normal; }  /* mouse over link */
21     .info a:active {color:darkgray; text-decoration: none; font-style:
           italic; }  /* selected link */
22   </ui:style>
23   <g:AbsolutePanel width="1920px" height="950px">
```

```
24      <g:at left="300" top="20">
25        <g:HTML>
26        <h1>MVP example</h1><h2>Agricola board game</h2>
27        </g:HTML>
28      </g:at>
29      <g:at left="0" top="134">
30        <g:SimplePanel ui:field="leftLayout"/>
31      </g:at>
32      <g:at left="405" top="131">
33        <g:SimplePanel ui:field="middleLayout"/>
34      </g:at>
35      <g:at left="810" top="0">
36        <g:SimplePanel ui:field="topLayout"/>
37      </g:at>
38      <g:at left="810" top="295">
39        <g:SimplePanel ui:field="bottomLayout"/>
40      </g:at>
41      <g:at left="1375" top="295">
42        <g:SimplePanel ui:field="rightLayout"/>
43      </g:at>
44      <g:at left="1312" top="0">
45        <g:SimplePanel ui:field="infoLayout"/>
46      </g:at>
47      <g:at left="0" top="130">
48        <g:Label width="810px" height="570px" styleName="{style.disabled}"
              visible="false" ui:field="disabledLabel"/>
49      </g:at>
50      <g:at left="810" top="290">
51        <g:Label width="560px" height="410px" styleName="{style.disabled}"
              visible="false" ui:field="disabledLabel2"/>
52      </g:at>
53      <g:at left="810" top="0">
54        <g:Label width="400px" height="290px" styleName="{style.disabled2}"
              visible="true" ui:field="disabledLabel3"/>
55      </g:at>
56      <g:at left="1370" top="290">
57        <g:Label width="550px" height="410px" styleName="{style.disabled}"
              visible="false" ui:field="disabledLabel4"/>
58      </g:at>
59
60      <g:at left="10" top="720">
61        <g:HTML styleName="{style.info}">
62          Images are photos of the original Agricola board game.<br/>
63          For more information see <a href="http://www.brettspiele-report.de
                /agricola/">this link</a>. <br/><br/>
64          The GWT source code of this game is written by Michael von
                Wenckstern. <br/>
65          The source code of this game is used in his diploma thesis to show
                 the Model-View-Presenter pattern.
66        </g:HTML>
67      </g:at>
68    </g:AbsolutePanel>
69  </ui:UiBinder>
```

Listing A41: AppView.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.uibinder.client.UiBinder;
5  import com.google.gwt.uibinder.client.UiField;
6  import com.google.gwt.user.client.ui.Composite;
7  import com.google.gwt.user.client.ui.Label;
8  import com.google.gwt.user.client.ui.SimplePanel;
9  import com.google.gwt.user.client.ui.Widget;
10
11 public class AppViewMobile extends Composite implements AppController.
       Display {
12
13   private static final Binder binder = GWT.create(Binder.class);
14   @UiField
15   SimplePanel leftLayout;
16   @UiField
17   SimplePanel middleLayout;
18   @UiField
19   SimplePanel topLayout;
20   @UiField
21   SimplePanel bottomLayout;
22   @UiField
23   SimplePanel rightLayout;
24   @UiField
25   SimplePanel infoLayout;
26   @UiField
27   Label disabledLabel;
28   @UiField
29   Label disabledLabel3;
30   @UiField
31   Label disabledLabel4;
32
33   interface Binder extends UiBinder<Widget, AppViewMobile> {
34   }
35
36   public AppViewMobile() {
37     initWidget(binder.createAndBindUi(this));
38   }
39
40   @Override
41   public void setLeftWidget(Widget w) {
42     leftLayout.setWidget(w);
43   }
44
45   @Override
46   public void setMiddleWidget(Widget w) {
47     middleLayout.setWidget(w);
48   }
49
50   @Override
51   public void setTopWidget(Widget w) {
52     topLayout.setWidget(w);
53   }
54
55   @Override
56   public void setBottomWidget(Widget w) {
57     bottomLayout.setWidget(w);
58   }
59
60   @Override
```

```
61    public void setRightWidget(Widget w) {
62      rightLayout.setWidget(w);
63    }
64
65    @Override
66    public void setInfoWidget(Widget w) {
67      infoLayout.setWidget(w);
68    }
69
70    @Override
71    public void setForcePlayerField(boolean b) {
72      disabledLabel.setVisible(b);
73    }
74
75    @Override
76    public void setForceBigAcquisitationField(boolean b) {
77      disabledLabel.setVisible(b);
78      disabledLabel4.setVisible(b);
79      disabledLabel3.setVisible(!b);
80    }
81
82    @Override
83    public boolean isBigAcquisitationFieldEnabled() {
84      return !disabledLabel3.isVisible();
85    }
86
87 }
```

Listing A42: AppViewMobile.java file

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3 <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='
      urn:import:com.google.gwt.user.client.ui'>
4   <ui:style>
5     .disabled {
6       background-color: rgba(0,0,0,0.6);
7       z-index: 101;
8       }
9       .disabled2 {
10      background-color: rgba(0,0,0,0.6);
11      z-index: 50;
12      pointer-events: none
13      }
14      .info {
15      color: darkgray;
16      font-size: medium;
17    }
18    .info a:link {color:darkgray; text-decoration: none; font-style:
          italic; }      /* unvisited link */
19     .info a:visited {color:darkgray; text-decoration: none; font-style:
          italic; }  /* visited link */
20     .info a:hover {color:darkgray; text-decoration: none; font-style:
          normal; }  /* mouse over link */
21    .info a:active {color:darkgray; text-decoration: none; font-style:
          italic; }  /* selected link */
22  </ui:style>
23  <g:AbsolutePanel width="1920px" height="950px">
24    <g:at left="20" top="20">
25      <g:HTML>
26      <h1>MVP example</h1><h2>Agricola board game</h2>
```

```
27        </g:HTML>
28      </g:at>
29      <g:at left="0" top="125">
30        <g:SimplePanel ui:field="leftLayout"/>
31      </g:at>
32      <g:at left="315" top="130">
33        <g:SimplePanel ui:field="middleLayout"/>
34      </g:at>
35      <g:at left="280" top="0">
36        <g:SimplePanel ui:field="topLayout"/>
37      </g:at>
38      <g:at left="630" top="235">
39        <g:SimplePanel ui:field="bottomLayout"/>
40      </g:at>
41      <g:at left="1050" top="295">
42        <g:SimplePanel ui:field="rightLayout"/>
43      </g:at>
44      <g:at left="700" top="0">
45        <g:SimplePanel ui:field="infoLayout"/>
46      </g:at>
47      <g:at left="0" top="130">
48        <g:Label width="1050px" height="570px" styleName="{style.disabled}"
              visible="false" ui:field="disabledLabel"/>
49      </g:at>
50      <g:at left="270" top="0">
51        <g:Label width="400px" height="130px" styleName="{style.disabled2}"
              visible="true" ui:field="disabledLabel3"/>
52      </g:at>
53      <g:at left="1050" top="290">
54        <g:Label width="400px" height="400px" styleName="{style.disabled}"
              visible="false" ui:field="disabledLabel4"/>
55      </g:at>
56
57      <g:at left="10" top="520">
58        <g:HTML styleName="{style.info}">
59          You are using the mobile version which has no images. <br/>
60          The GWT source code of this game is written by Michael von
                Wenckstern. <br/>
61          The source code of this game is used in his diploma thesis to show
                the Model-View-Presenter pattern.
62        </g:HTML>
63      </g:at>
64    </g:AbsolutePanel>
65 </ui:UiBinder>
```

Listing A43: AppViewMobile.ui.xml file

```
74 public class EventBus extends HandlerManager implements
       HasAddResourceHandler, HasBuildHouseHandler, HasPlayerFieldDoneHandler
       , HasPlowFieldHandler,
75     HasBuildFenceHandler, HasFamilyAdditionHandler, HasRestaurateHandler,
         HasSeedHandler, HasGetSheepHandler, HasGetBoarHandler,
         HasGetCowHandler,
76     HasPlowFieldSeedHandler, HasFamilyAdditionWithoutHouseHandler,
         HasRestaurateAndFenceHandler, HasChildStartsWorkingHandler,
         HasNextRoundHandler,
77     HasGetBigAcquisitionHandler, HasEnableBigAcquisitionHandler,
         HasShowingDialogHandler, HasHistoryChangedHandler,
         HasRequestHistoryHandler, HasSaveHistoryToURLHandler {
78
79     private EventBus() {
```

```
80      super(null);
81    }
82    private static EventBus eventBus = new EventBus();
83    public static EventBus getEventBus() {
84      return eventBus;
85    }
86
87    public static void fire(GwtEvent<?> event) {
88      eventBus.fireEvent(event);
89    }
90
91    @Override
92    public HandlerRegistration addAddResourceHandler(AddResourceHandler
          handler) {
93      return addHandler(AddResourceEvent.getType(), handler);
94    }
95
96    @Override
97    public HandlerRegistration addBuildHouseHandler(BuildHouseHandler
          handler) {
98      return addHandler(BuildHouseEvent.getType(), handler);
99    }
100
101   @Override
102   public HandlerRegistration addPlayerFieldDoneHandler(
          PlayerFieldDoneHandler handler) {
103     return addHandler(PlayerFieldDoneEvent.getType(), handler);
104   }
105
106   @Override
107   public HandlerRegistration addPlowFieldHandler(PlowFieldHandler handler)
          {
108     return addHandler(PlowFieldEvent.getType(), handler);
109   }
110
111   @Override
112   public HandlerRegistration addSeedHandler(SeedHandler handler) {
113     return addHandler(SeedEvent.getType(), handler);
114   }
115
116   @Override
117   public HandlerRegistration addRestaurateHandler(RestaurateHandler
          handler) {
118     return addHandler(RestaurateEvent.getType(), handler);
119   }
120
121   @Override
122   public HandlerRegistration addFamilyAdditionHandler(
          FamilyAdditionHandler handler) {
123     return addHandler(FamilyAdditionEvent.getType(), handler);
124   }
125
126   @Override
127   public HandlerRegistration addBuildFenceHandler(BuildFenceHandler
          handler) {
128     return addHandler(BuildFenceEvent.getType(), handler);
129   }
130
131   @Override
132   public HandlerRegistration addGetSheepHandler(GetSheepHandler handler) {
133     return addHandler(GetSheepEvent.getType(), handler);
```

```
134   }
135
136   @Override
137   public HandlerRegistration addRestaurateAndFenceHandler(
          RestaurateAndFenceHandler handler) {
138     return addHandler(RestaurateAndFenceEvent.getType(), handler);
139   }
140
141   @Override
142   public HandlerRegistration addFamilyAdditionWithoutHouseHandler(
          FamilyAdditionWithoutHouseHandler handler) {
143     return addHandler(FamilyAdditionWithoutHouseEvent.getType(), handler);
144   }
145
146   @Override
147   public HandlerRegistration addPlowFieldSeedHandler(PlowFieldSeedHandler
          handler) {
148     return addHandler(PlowFieldSeedEvent.getType(), handler);
149   }
150
151   @Override
152   public HandlerRegistration addGetCowHandler(GetCowHandler handler) {
153     return addHandler(GetCowEvent.getType(), handler);
154   }
155
156   @Override
157   public HandlerRegistration addGetBoarHandler(GetBoarHandler handler) {
158     return addHandler(GetBoarEvent.getType(), handler);
159   }
160
161   @Override
162   public HandlerRegistration addChildStartsWorkingHandler(
          ChildStartsWorkingHandler handler) {
163     return addHandler(ChildStartsWorkingEvent.getType(), handler);
164   }
165
166   @Override
167   public HandlerRegistration addNextRoundHandler(NextRoundHandler handler)
          {
168     return addHandler(NextRoundEvent.getType(), handler);
169   }
170
171   @Override
172   public HandlerRegistration addGetBigAcquisitionHandler(
          GetBigAcquisitionHandler handler) {
173     return addHandler(GetBigAcquisitionEvent.getType(), handler);
174   }
175
176   @Override
177   public HandlerRegistration addEnableBigAcquisitionHandler(
          EnableBigAcquisitionHandler handler) {
178     return addHandler(EnableBigAcquisitionEvent.getType(), handler);
179   }
180
181   @Override
182   public HandlerRegistration addShwoingDialogHandler(ShowingDialogHandler
          handler) {
183     return addHandler(ShowingDialogEvent.getType(), handler);
184   }
185
186   @Override
```

```
187   public HandlerRegistration addRequestHistoryHandler(
          RequestHistoryHandler handler) {
188     return addHandler(RequestHistoryEvent.getType(), handler);
189   }
190
191   @Override
192   public HandlerRegistration addHistoryChangedHandler(
          HistoryChangedHandler handler) {
193     return addHandler(HistoryChangedEvent.getType(), handler);
194   }
195
196   @Override
197   public HandlerRegistration addSaveHistoryToURLHandler(
          SaveHistoryToURLHandler handler) {
198     return addHandler(SaveHistoryToURLEvent.getType(), handler);
199   }
200 }
```

Listing A44: EventBus.java file (imports omitted)

```
38 /**
39  * manages the history event, when the user changed the url by pressing
        the back or forward button.
40  * it also creates the url string which represents a history state of the
        complete application and deserializes the string to the state again.
41  * @author wencky
42  *
43  */
44
45 @SuppressWarnings("rawtypes")
46 public class HistoryController implements RequestHistoryHandler,
      HistoryChangedHandler, ValueChangeHandler<String>,
      SaveHistoryToURLHandler{
47   private static HistoryController histController;
48   public static HistoryController getInstance(){
49     if (histController == null) {
50       histController = new HistoryController();
51       if ("".equals(History.getToken())) {
52             History.newItem("start", false);
53       } else {
54             History.fireCurrentHistoryState();
55       }
56     }
57     return histController;
58   }
59
60   private final Class[] modelClasses = {AcquisitionCardModel.class,
         BackgroundCard.class, BaseFieldModel.class, BigAcquisitationsModel.
         class, BigAcquisitions.class, BigFieldModel.class,
61     CardFieldModel.class, Child.class, FieldCard.class, Player.class,
          PlayerFieldModel.class, PlayerResourceModel.class, Resource.class,
           Rounds1To7Model.class, Rounds8To14Model.class, SmallFieldModel.
          class,
62     SmallFieldModel[].class, AcquisitionCardModel[].class};
63   private String[] modelNames = null;
64   private String[] shortNames = null;
65
66   private HistoryMap historyMap = new HistoryMap();
67   private String oldHistoryToken;
68
```

```
69  /** this map contains all activity presenters who registered using the
        addActivityPresenter method*/
70  private HashMap<Integer, Activity> activityPresenterMap = new HashMap<
        Integer, Activity>();
71  /** this list contains all presenters whose history has changed */
72  private ArrayList<Integer> activityHistoryChanged = new ArrayList<
        Integer>();
73
74  private HistoryController() {
75    shortNames = new String[modelClasses.length];
76    modelNames = new String[modelClasses.length];
77    for(int i=0; i<modelNames.length; i++) {
78      modelNames[i] = Serializer.getSerializationSignature(modelClasses[i
          ]) + "\\!";
79      shortNames[i] = "S" + i + "M\\!";
80    }
81    bind();
82  }
83
84  private void bind() {
85    // Event is fired when the URL history changed
86    History.addValueChangeHandler(this);
87    // Event is fired when a component changed their history, e.g. when a
        user loads a table
88    EventBus.getEventBus().addHistoryChangedHandler(this);
89    // Event is fired when a presenter becomes visible and wants to know
        its actual history to update the view
90    EventBus.getEventBus().addRequestHistoryHandler(this);
91    // Event is fired when the changed history should get stored in the
        URL
92    EventBus.getEventBus().addSaveHistoryToURLHandler(this);
93  }
94
95  public void addActivityPresenter(Activity activity) {
96    activityPresenterMap.put(activity.getActivityKey().hashCode(),
        activity);
97  }
98  public void removeActivityPresenter(Activity activity) {
99    activityPresenterMap.remove(activity.getActivityKey());
100 }
101
102 @SuppressWarnings("unchecked")
103 @Override
104 public void onValueChange(ValueChangeEvent<String> event) { /* 1 */
105   String token = event.getValue();
106
107     if (token != null && !token.equals(oldHistoryToken)) {
108
109       oldHistoryToken = token;
110
111       if (token.equals("start")) {
112         Window.Location.reload(); // reload the app
113       } else {
114         String deserilized = URL.decode(token).replace("yy", "\\!");
115         deserilized = deserilized.replace("\\!T\\!", "\\!true\\!");
116         deserilized = deserilized.replace("\\!F\\!", "\\!false\\!");
117         deserilized = (Serializer.getSerializationSignature(HistoryMap.
              class) + "\\!") + deserilized;
118         for(int i=0; i<modelNames.length; i++) {
119           deserilized = deserilized.replace(shortNames[i], modelNames[i
                ]);
120         }
```

```
121           HistoryMap newHistoryMap = Serializer.deserialize(deserilized);
122           for(int key : newHistoryMap.keySet()) {
123             if(historyMap.containsKey(key)) {
124               Serializable newHistory = newHistoryMap.get(key);
125               if(!historyMap.get(key).equals(newHistory)) {
126                 // history of this activity presenter changed
127                 Activity activity = activityPresenterMap.get(key);
128                 if(activity != null) {
129                   activity.setActualHistory(newHistory);
130                 }
131               }
132             } else if(activityPresenterMap.containsKey(key)) {
133               // history is new, because the key is not part of historyMap
134               Serializable newHistory = newHistoryMap.get(key);
135               Activity activity = activityPresenterMap.get(key);
136             if(activity != null) {
137               activity.setActualHistory(newHistory);
138             }
139             }
140           }
141           for(int key : historyMap.keySet()) {
142             if(!newHistoryMap.containsKey(key)) {
143               Activity activity = activityPresenterMap.get(key);
144             if(activity != null) {
145               activity.setActualHistory(null); // the old state is not
                      anymore in the null state, so we have to reset the state
146             }
147             }
148           }
149           historyMap = newHistoryMap;
150         }
151     }
152
153   }
154
155
156   @Override
157   public void onHistoryChanged(HistoryChangedEvent event) {
158     if(!activityHistoryChanged.contains(event.getActivity().getActivityKey
          ().hashCode())) {
159       activityHistoryChanged.add(event.getActivity().getActivityKey().
            hashCode());
160     }
161   }
162
163
164
165   @SuppressWarnings("unchecked")
166   @Override
167   public void onRequestHistory(RequestHistoryEvent event) {
168     Serializable s = historyMap.get(event.getActivity().getActivityKey().
          hashCode());
169     event.getActivity().setActualHistory(s);
170   }
171
172   @Override
173   public void onSaveHistoryToURL(SaveHistoryToURLEvent event) {
174     boolean changed = false;
175     for(int historyKey: activityHistoryChanged) {
176       Activity a = activityPresenterMap.get(historyKey);
177       if(a != null) {
178         Serializable oldHistory = historyMap.get(historyKey);
```

```
179           Serializable history = a.getActualHistory();
180           changed |= (oldHistory == null || !oldHistory.equals(history));
181           historyMap.put(historyKey, history);
182         }
183       }
184       activityHistoryChanged.clear();
185       if(changed) {
186         // the history changed
187         String serialized = Serializer.serialize(historyMap);
188         // removing useless values, because it starts everytime with the
              same
189         serialized = serialized.substring((Serializer.
              getSerializationSignature(HistoryMap.class) + "\\!").length());
190         for(int i=0; i<modelNames.length; i++) {
191               serialized = serialized.replace(modelNames[i], shortNames[i]);
192           }
193         serialized = serialized.replace("\\!true\\!", "\\!T\\!");
194         serialized = serialized.replace("\\!false\\!", "\\!F\\!");
195         oldHistoryToken = URL.encode(serialized.replace("\\!", "yy"));
196         History.newItem(oldHistoryToken, false);
197       }
198     }
199 }
```

Listing A45: HistoryController.java file (imports omitted)

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  public class Utils {
4    /**
5    * copied from Apache ObjectUtils library
6     * <p>Compares two objects for equality, where either one or both
7     * objects may be <code>null</code>.</p>
8     *
9     * <pre>
10    * ObjectUtils.equals(null, null)                  = true
11    * ObjectUtils.equals(null, "")                    = false
12    * ObjectUtils.equals("", null)                    = false
13    * ObjectUtils.equals("", "")                      = true
14    * ObjectUtils.equals(Boolean.TRUE, null)          = false
15    * ObjectUtils.equals(Boolean.TRUE, "true")        = false
16    * ObjectUtils.equals(Boolean.TRUE, Boolean.TRUE)  = true
17    * ObjectUtils.equals(Boolean.TRUE, Boolean.FALSE) = false
18    * </pre>
19    *
20    * @param object1  the first object, may be <code>null</code>
21    * @param object2  the second object, may be <code>null</code>
22    * @return <code>true</code> if the values of both objects are the same
23    */
24   public static boolean equals(Object object1, Object object2) {
25     if (object1 == object2) {
26       return true;
27     }
28     if ((object1 == null) || (object2 == null)) {
29       return false;
30     }
31     return object1.equals(object2);
32   }
33 }
```

Listing A46: Utils.java file

de.tu_freiberg.informatik.vonwenckstern.client.event package

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.event;
2
3  import com.google.gwt.event.shared.EventHandler;
4  import com.google.gwt.event.shared.GwtEvent;
5  import com.google.gwt.event.shared.HandlerRegistration;
6
7  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       AddResourceEvent.AddResourceHandler;
8  import de.tu_freiberg.informatik.vonwenckstern.client.model.Resource;
9
10 /**
11  * Fires after the user selected a card field containing resources
12  */
13 public class AddResourceEvent extends GwtEvent<AddResourceHandler> {
14
15   /**
16    * Handler type.
17    */
18   private static Type<AddResourceHandler> TYPE;
19
20   /**
21    * Gets the type associated with this event.
22    *
23    * @return returns the handler type
24    */
25   public static Type<AddResourceHandler> getType() {
26     if (TYPE == null) {
27       TYPE = new Type<AddResourceHandler>();
28     }
29     return TYPE;
30   }
31
32   public static class RessourceItem {
33     public RessourceItem(Resource res, int resCount) {
34       this.res = res;
35       this.resCount = resCount;
36     }
37     public Resource res = Resource.R_NONE;
38     public int resCount = 0;
39   }
40
41   private RessourceItem[] items;
42
43   public AddResourceEvent(RessourceItem[] items) {
44     this.items = items;
45   }
46
47   @SuppressWarnings({"unchecked", "rawtypes"})
48   @Override
49   public Type<AddResourceHandler> getAssociatedType() {
50     return (Type) TYPE;
51   }
52
53   public RessourceItem[] getItems() {
54     return items;
55   }
56
```

```
57    @Override
58    protected void dispatch(AddResourceHandler handler) {
59      handler.onAddResource(this);
60    }
61
62    /**
63     * Handler class for {@link AddResourceEvent} events.
64     */
65    public interface AddResourceHandler extends EventHandler {
66
67      /**
68       * Called when a player selected resources
69       */
70      void onAddResource(AddResourceEvent event);
71    }
72
73    /**
74     * A widget that implements this interface is a public source of
75     * {@link AddResourceEvent} events.
76     */
77    public interface HasAddResourceHandler {
78
79      /**
80       * Adds a {@link AddResourceHandler} handler for {@link
             AddResourceEvent} events.
81       *
82       * @param handler the handler
83       * @return the registration for the event
84       */
85      HandlerRegistration addAddResourceHandler(AddResourceHandler handler);
86    }
87
88  }
```

Listing A47: AddResourceEvent.java file

The other files are too similar to get listed here.

de.tu_freiberg.informatik.vonwenckstern.client.model package

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  import java.io.Serializable;
4
5  import de.tu_freiberg.informatik.vonwenckstern.client.Utils;
6
7  public class AcquisitionCardModel implements Serializable {
8    private static final long serialVersionUID = 1L;
9    private BigAcquisitions acquisition = BigAcquisitions.BA_NONE;
10   private boolean selectable = false;
11   private transient String description = "";
12   private boolean visible = true;
13
14   public boolean equals(Object o) {
15     if(!(o instanceof AcquisitionCardModel))
16       return false;
17     AcquisitionCardModel am = (AcquisitionCardModel)o;
18     return acquisition == am.acquisition && selectable == am.selectable &&
           Utils.equals(description, am.description) && visible == am.
           visible;
19   }
20
21   public boolean isVisible() {
22     return visible;
23   }
24   public void setVisible(boolean visible) {
25     this.visible = visible;
26   }
27   public AcquisitionCardModel() {}
28   public AcquisitionCardModel(BigAcquisitions acquisition) {
29     this.acquisition = acquisition;
30   }
31
32   public AcquisitionCardModel(BigAcquisitions acquisition, boolean visible
       ) {
33     this.acquisition = acquisition;
34     this.visible = visible;
35   }
36
37   public AcquisitionCardModel(BigAcquisitions acquisition, boolean visible
       , String description) {
38     this.acquisition = acquisition;
39     this.visible = visible;
40     this.description = description;
41   }
42
43   public String getDescription() {
44     return description;
45   }
46   public void setDescription(String description) {
47     this.description = description;
48   }
49   public BigAcquisitions getAcquisition() {
50     return acquisition;
51   }
52   public void setAcquisition(BigAcquisitions acquisition) {
53     this.acquisition = acquisition;
```

```
54    }
55    public boolean isSelectable() {
56      return selectable;
57    }
58    public void setSelectable(boolean selectable) {
59      this.selectable = selectable;
60    }
61  }
```

Listing A48: AcquisitionCardModel.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  public enum BackgroundCard {
4    NONE, SHEEP, ACQUISITION, FENCE, SEEDING_BACKING, FAMILIY_ADDITION2,
         STONE2,
5    RESTAURATION, BOAR, VEGETABLE, STONE4, COW, PLOWING_SOWING,
         FAMILY_ADDITION5,
6    RESTAURATION_FENCE, ONE_WOOD,  TWO_CLAY, TWO_WOOD, REED_STONE_FOOD,
         CABARET,
7  }
```

Listing A49: BackgroundCard.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  import java.io.Serializable;
4
5  import de.tu_freiberg.informatik.vonwenckstern.client.Utils;
6
7  public class BaseFieldModel implements Serializable {
8    private static final long serialVersionUID = 1L;
9    private Resource ressource = Resource.R_NONE;
10   private int ressourceCount = 0;
11   private Child child = Child.C_NONE;
12   private transient String description = null;
13   private transient String id = null;
14
15   public BaseFieldModel() {}
16
17   public BaseFieldModel(String id) {
18     this.id = id;
19   }
20
21   public boolean equals(Object o) {
22     if( !(o instanceof BaseFieldModel))
23       return false;
24     BaseFieldModel bm = (BaseFieldModel)o;
25     return ressource == bm.getRessource() && ressourceCount == bm.
           getRessourceCount() && child == bm.getChild() && Utils.equals(
           description, bm.getDescription())
26         && Utils.equals(id, bm.getId());
27   }
28
29   public String getId() {
30     return id;
31   }
32   public void setId(String id) {
33     this.id = id;
34   }
35   public String getDescription() {
```

```
36      return description;
37    }
38    public void setDescription(String description) {
39      if(description != null)
40        this.description = description.replace("\\n", "\n");
41    }
42
43    public Child getChild() {
44      return child;
45    }
46    public void setChild(Child child) {
47      this.child = child;
48    }
49    public Resource getRessource() {
50      return ressource;
51    }
52    public void setRessource(Resource ressource) {
53      this.ressource = ressource;
54    }
55    public int getRessourceCount() {
56      return ressourceCount;
57    }
58    public void setRessourceCount(int ressourceCount) {
59      if(ressourceCount < 0) throw new RuntimeException("ressource count
          cannot be negative");
60      this.ressourceCount = ressourceCount;
61    }
62  }
```

Listing A50: BaseFieldModel.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  import java.io.Serializable;
4
5  public class BigAcquisitationsModel implements Serializable {
6    private static final long serialVersionUID = 1L;
7    private AcquisitionCardModel[] models = new AcquisitionCardModel[] { new
         AcquisitionCardModel(BigAcquisitions.BA_FIRE_PLACE), new
       AcquisitionCardModel(BigAcquisitions.BA_FIRE_PLACE2), new
       AcquisitionCardModel(BigAcquisitions.BA_COOKERY), new
       AcquisitionCardModel(BigAcquisitions.BA_COOKERY2), new
       AcquisitionCardModel(BigAcquisitions.BA_FOUNTAIN),
8      new AcquisitionCardModel(BigAcquisitions.BA_CLAY_OVEN), new
           AcquisitionCardModel(BigAcquisitions.BA_STONE_OVEN), new
           AcquisitionCardModel(BigAcquisitions.BA_JOINERY), new
           AcquisitionCardModel(BigAcquisitions.BA_POTTERY), new
           AcquisitionCardModel(BigAcquisitions.BA_BASKET_MAKER)};
9
10   public void update(BigAcquisitationsModel model) {
11     if(model == null) return;
12     for(int i=0; i<10; i++) {
13       models[i].setAcquisition(model.models[i].getAcquisition());
14       models[i].setVisible(model.models[i].isVisible());
15     }
16   }
17
18   public void setModel(int id, AcquisitionCardModel m) {
19     models[id] = m;
20   }
21
```

```
22    public AcquisitionCardModel getModel(int id) {
23      return models[id];
24    }
25
26    public boolean equals(Object o) {
27      if(!(o instanceof BigAcquisitationsModel))
28        return false;
29      BigAcquisitationsModel bm = (BigAcquisitationsModel)o;
30      return models[0].equals(bm.getModel(0)) &&models[1].equals(bm.getModel
            (1)) && models[2].equals(bm.getModel(2)) && models[3].equals(bm.
            getModel(3)) && models[4].equals(bm.getModel(4)) && models[5].
            equals(bm.getModel(5)) &&
31          models[6].equals(bm.getModel(6)) && models[7].equals(bm.getModel
                (7)) && models[8].equals(bm.getModel(8)) && models[9].equals(
                bm.getModel(9));
32    }
33
34    public BigAcquisitationsModel clone() {
35      BigAcquisitationsModel model = new BigAcquisitationsModel();
36      for(int i=0; i<10; i++) {
37        model.setModel(i, new AcquisitionCardModel(models[i].getAcquisition
              (), models[i].isVisible()));
38      }
39      return model;
40    }
41  }
```

Listing A51: BigAcquisitationsModel.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  public enum BigAcquisitions {
4    BA_NONE,
5    BA_FIRE_PLACE, BA_FIRE_PLACE2, BA_COOKERY, BA_COOKERY2, BA_FOUNTAIN,
6    BA_CLAY_OVEN, BA_STONE_OVEN, BA_JOINERY, BA_POTTERY, BA_BASKET_MAKER
7  }
```

Listing A52: BigAcquisitions.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  import java.io.Serializable;
4
5  public class BigFieldModel extends BaseFieldModel implements Serializable
       {
6    private static final long serialVersionUID = 1L;
7    private transient BackgroundCard bgCard = BackgroundCard.NONE;
8    private transient int ressourceRoundAddition;
9    private boolean visible;
10
11   public boolean isVisible() {
12     return visible;
13   }
14
15   public void setVisible(boolean visible) {
16     this.visible = visible;
17   }
18
19   public BigFieldModel() { super(); }
20
21   public BigFieldModel(BackgroundCard bgCard, Resource res, int resCount,
         int resRoundAddition, String describtion, boolean visible) {
```

```
22      super();
23      this.bgCard = bgCard;
24      this.setRessource(res);
25      this.setRessourceCount(resCount);
26      this.ressourceRoundAddition = resRoundAddition;
27      this.setDescription(describtion);
28      this.visible = visible;
29    }
30
31    public BigFieldModel(int resCount, boolean visible) {
32      super();
33      this.setRessourceCount(resCount);
34      this.visible = visible;
35    }
36
37    public int getRessourceRoundAddition() {
38      return ressourceRoundAddition;
39    }
40    public void setRessourceRoundAddition(int ressourceRoundAddition) {
41      if(ressourceRoundAddition < 0) throw new RuntimeException("
            ressourceRoundAddition represents the number of new ressources
            which are added to the field in each round and it cannot be
            negative");
42      this.ressourceRoundAddition = ressourceRoundAddition;
43    }
44
45    public BackgroundCard getBgCard() {
46      return bgCard;
47    }
48    public void setBgCard(BackgroundCard bgCard) {
49      this.bgCard = bgCard;
50    }
51
52    @Override
53    public boolean equals(Object o) {
54      if( !(o instanceof BigFieldModel))
55        return false;
56      BigFieldModel bm = (BigFieldModel)o;
57      return super.equals((BaseFieldModel)bm) && bgCard == bm.getBgCard() &&
             ressourceRoundAddition == bm.getRessourceRoundAddition() &&
            visible == bm.isVisible();
58    }
59  }
```

Listing A53: BigFieldModel.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  import java.io.Serializable;
4
5  public class CardFieldModel implements Serializable {
6    private static final long serialVersionUID = 1L;
7    private BigFieldModel modelOneWood = new BigFieldModel(BackgroundCard.
       ONE_WOOD, Resource.R_WOOD, 1, 1, "get woods", true);
8    private BigFieldModel modelTwoWood = new BigFieldModel(BackgroundCard.
       TWO_WOOD, Resource.R_WOOD, 2, 2, "get woods", true);
9    private BigFieldModel modelTwoClay = new BigFieldModel(BackgroundCard.
       TWO_CLAY, Resource.R_CLAY, 2, 2, "get clays", true);
10   private BigFieldModel modelReedStoneFood = new BigFieldModel(
       BackgroundCard.REED_STONE_FOOD, Resource.R_NONE, 0, 0, "get one reed
       , one stone and one food marker", true);;
```

```
11    private BigFieldModel modelCabaret = new BigFieldModel(BackgroundCard.
          CABARET, Resource.R_FOOD, 1, 1, "get␣food␣markers", true);
12
13    private BaseFieldModel modelHouse = new BaseFieldModel("buildHouse");
14    private BaseFieldModel modelStartPlayer = new BaseFieldModel("
          startPlayer");
15    private BaseFieldModel modelGrain = new BaseFieldModel("oneGrain");
16    private BaseFieldModel modelPlowField = new BaseFieldModel("plowField");
17    private BaseFieldModel modelFood = new BaseFieldModel("twoFoodMarkers");
18
19    public BaseFieldModel getModelHouse() {
20      return modelHouse;
21    }
22    public void setModelHouse(BaseFieldModel modelHouse) {
23      this.modelHouse = modelHouse;
24    }
25    public BaseFieldModel getModelStartPlayer() {
26      return modelStartPlayer;
27    }
28    public void setModelStartPlayer(BaseFieldModel modelStartPlayer) {
29      this.modelStartPlayer = modelStartPlayer;
30    }
31    public BaseFieldModel getModelGrain() {
32      return modelGrain;
33    }
34    public void setModelGrain(BaseFieldModel modelGrain) {
35      this.modelGrain = modelGrain;
36    }
37    public BaseFieldModel getModelPlowField() {
38      return modelPlowField;
39    }
40    public void setModelPlowField(BaseFieldModel modelPlowField) {
41      this.modelPlowField = modelPlowField;
42    }
43    public BaseFieldModel getModelFood() {
44      return modelFood;
45    }
46    public void setModelFood(BaseFieldModel modelFood) {
47      this.modelFood = modelFood;
48    }
49    public BigFieldModel getModelOneWood() {
50      return modelOneWood;
51    }
52    public BigFieldModel getModelTwoWood() {
53      return modelTwoWood;
54    }
55    public BigFieldModel getModelTwoClay() {
56      return modelTwoClay;
57    }
58    public BigFieldModel getModelReedStoneFood() {
59      return modelReedStoneFood;
60    }
61    public BigFieldModel getModelCabaret() {
62      return modelCabaret;
63    }
64
65    @Override
66    public boolean equals(Object o) {
67      if( !(o instanceof CardFieldModel) )
68        return false;
69      CardFieldModel cm = (CardFieldModel)o;
70      return modelOneWood.equals(cm.getModelOneWood()) && modelTwoWood.
            equals(cm.getModelTwoWood()) && modelTwoClay.equals(cm.
```

```
            getModelTwoClay())
71          && modelReedStoneFood.equals(cm.getModelReedStoneFood()) &&
                modelCabaret.equals(cm.getModelCabaret()) &&
72          modelHouse.equals(cm.modelHouse) && modelStartPlayer.equals(cm.
                modelStartPlayer) && modelGrain.equals(cm.modelGrain) &&
73          modelPlowField.equals(cm.modelPlowField) && modelFood.equals(cm.
                modelFood);
74    }
75
76    public void update(CardFieldModel model) {
77      if(model == null)
78        return;
79      _cloneBM(this.getModelOneWood(), model.getModelOneWood());
80      _cloneBM(this.getModelTwoWood(), model.getModelTwoWood());
81      _cloneBM(this.getModelTwoClay(), model.getModelTwoClay());
82      _cloneBM(this.getModelReedStoneFood(), model.getModelReedStoneFood());
83      _cloneBM(this.getModelCabaret(), model.getModelCabaret());
84
85      _cloneBaseM(this.modelHouse, model.modelHouse);
86      _cloneBaseM(this.modelStartPlayer, model.modelStartPlayer);
87      _cloneBaseM(this.modelGrain, model.modelGrain);
88      _cloneBaseM(this.modelPlowField, model.modelPlowField);
89      _cloneBaseM(this.modelFood, model.modelFood);
90    }
91
92    public CardFieldModel clone() {
93      CardFieldModel model = new CardFieldModel();
94      _cloneBM(model.getModelOneWood(), this.getModelOneWood());
95      _cloneBM(model.getModelTwoWood(), this.getModelTwoWood());
96      _cloneBM(model.getModelTwoClay(), this.getModelTwoClay());
97      _cloneBM(model.getModelReedStoneFood(), this.getModelReedStoneFood());
98      _cloneBM(model.getModelCabaret(), this.getModelCabaret());
99
100     _cloneBaseM(model.modelHouse, this.modelHouse);
101     _cloneBaseM(model.modelStartPlayer, this.modelStartPlayer);
102     _cloneBaseM(model.modelGrain, this.modelGrain);
103     _cloneBaseM(model.modelPlowField, this.modelPlowField);
104     _cloneBaseM(model.modelFood, this.modelFood);
105     return model;
106   }
107
108   private void _cloneBM(BigFieldModel bm, BigFieldModel org) {
109 //    bm.setBgCard(org.getBgCard());
110     bm.setChild(org.getChild());
111 //    bm.setDescription(org.getDescription());
112 //    bm.setId(org.getId());
113     bm.setRessource(org.getRessource());
114     bm.setRessourceCount(org.getRessourceCount());
115     bm.setVisible(org.isVisible());
116   }
117
118   private void _cloneBaseM(BaseFieldModel bm, BaseFieldModel org) {
119     bm.setChild(org.getChild());
120 //    bm.setDescription(org.getDescription());
121 //    bm.setId(org.getId());
122     bm.setRessource(org.getRessource());
123     bm.setRessourceCount(org.getRessourceCount());
124   }
125 }
```

Listing A54: CardFieldModel.java file

```
1 package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3 public enum Child {
4   C_NONE, C_BLUE, C_GREEN, C_ROSA, C_RED
5 }
```

Listing A55: Child.java file

```
1 package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3 public enum FieldCard {
4   F_NONE, F_FIELD, F_WOOD_HOUSE, F_CLAY_HOUSE, F_STONE_HOUSE, F_STABLE
5 }
```

Listing A56: FieldCard.java file

```
1 package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3 public interface HasAcquisitionCardModel {
4   public AcquisitionCardModel getModel();
5 }
```

Listing A57: HasAcquisitionCardModel.java file

```
1 package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3
4 public interface HasBaseFieldModel {
5   public BaseFieldModel getModel();
6 }
```

Listing A58: HasBaseFieldModel.java file

```
1 package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3 import java.io.Serializable;
4 import java.util.HashMap;
5
6 public class HistoryMap extends HashMap<Integer, Serializable> implements
      Serializable {
7   private static final long serialVersionUID = 1L;
8 }
```

Listing A59: HistoryMap.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  import java.io.Serializable;
4
5  import com.google.gwt.user.client.rpc.SerializationException;
6  import com.google.gwt.user.client.rpc.SerializationStreamReader;
7  import com.google.gwt.user.client.rpc.SerializationStreamWriter;
8
9
10
11
12 public class HistoryMap_CustomFieldSerializer {
13
14   public static void deserialize(SerializationStreamReader streamReader,
         HistoryMap instance) throws SerializationException {
15     int number = streamReader.readInt();
16     for(int i=0; i<number; i++) {
17       int key = streamReader.readInt();
18       instance.put(key, (Serializable) streamReader.readObject());
19     }
20   }
21
22   public static HistoryMap instantiate(SerializationStreamReader
         streamReader) throws SerializationException {
23     return new HistoryMap();
24   }
25
26   public static void serialize(SerializationStreamWriter streamWriter,
         HistoryMap instance) throws SerializationException {
27     streamWriter.writeInt(instance.size());
28     for(int key : instance.keySet()) {
29       streamWriter.writeInt(key);
30       streamWriter.writeObject(instance.get(key));
31     }
32   }
33 }
```

Listing A60: HistoryMap_CustomFieldSerializer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  public enum Player {
4    NONE, ROSA, GREEN, BLUE, RED
5  }
```

Listing A61: Player.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  import java.io.Serializable;
4  import java.util.ArrayList;
5  import java.util.Arrays;
6
7  import de.tu_freiberg.informatik.vonwenckstern.client.model.
       PlayerFieldModel.PastureInfo.Animal;
8
9  public class PlayerFieldModel implements Serializable {
10   private static final long serialVersionUID = 1L;
11   private int fenceId = 1;
12   private SmallFieldModel[] fields = null;
```

```
13  private AcquisitionCardModel[] acquisitations = new AcquisitionCardModel
        [] {null, null, null, null, null, null, null, null, null};
14  private transient Player player;
15  private boolean allowJoinery = true;
16  private boolean allowPottery = true;
17  private boolean allowBasketMaker = true;
18  int round = 1;
19
20  public int getRound() {
21    return round;
22  }
23
24  public void setRound(int round) {
25    this.round = round;
26  }
27
28  public void increaseRound() {
29    round++;
30  }
31
32  public boolean equals(Object o) {
33    if( !(o instanceof PlayerFieldModel)) {
34      return false;
35    }
36    PlayerFieldModel pm = (PlayerFieldModel)o;
37    return fenceId == pm.fenceId && Arrays.equals(fields, pm.fields) &&
          Arrays.equals(acquisitations, pm.acquisitations) && player == pm.
          player &&
38        allowJoinery == pm.allowJoinery && allowPottery == pm.allowPottery
             && allowBasketMaker == pm.allowBasketMaker && round == pm.
            round;
39  }
40
41  public void update(PlayerFieldModel model) {
42    if(model == null) return;
43    fields = new SmallFieldModel[model.fields.length];
44    for(int i=0; i<model.fields.length ; i++) {
45      SmallFieldModel sm = model.fields[i];
46      if(sm == null)
47        fields[i] = null;
48      else
49        fields[i] = new SmallFieldModel(sm.getField(), sm.getLeftFence(),
              sm.getRightFence(), sm.getTopFence(), sm.getBottomFence(), sm.
              isSelectable(), sm.getPersonsCount(), sm.getPersonsAtWork(),
50            sm.getRessource(), sm.getRessourceCount(), sm.getChild(), sm.
                  getDescription(), sm.getId(), player);
51    }
52    acquisitations = new AcquisitionCardModel[model.acquisitations.length
          ];
53    for(int i=0; i<model.acquisitations.length; i++) {
54      AcquisitionCardModel am = model.acquisitations[i];
55      if(am == null)
56        acquisitations[i] = null;
57      else
58        acquisitations[i] = new AcquisitionCardModel(am.getAcquisition(),
              am.isVisible());
59    }
60    allowJoinery = model.allowJoinery;
61    allowPottery = model.allowPottery;
62    allowBasketMaker = model.allowBasketMaker;
63    round = model.round;
```

```
64   }
65
66   public PlayerFieldModel clone() {
67     PlayerFieldModel pm = new PlayerFieldModel(this.player);
68     pm.fields = new SmallFieldModel[fields.length];
69     for(int i=0; i<fields.length ; i++) {
70       SmallFieldModel sm = fields[i];
71       if(sm == null)
72         pm.fields[i] = null;
73       else
74         pm.fields[i] = new SmallFieldModel(sm.getField(), sm.getLeftFence
               (), sm.getRightFence(), sm.getTopFence(), sm.getBottomFence(),
                sm.isSelectable(), sm.getPersonsCount(), sm.getPersonsAtWork
               (),
75               sm.getRessource(), sm.getRessourceCount(), sm.getChild(), sm.
                    getDescription(), sm.getId(), sm.getPlayer());
76     }
77     pm.acquisitations = new AcquisitionCardModel[acquisitations.length];
78     for(int i=0; i<acquisitations.length; i++) {
79       AcquisitionCardModel am = acquisitations[i];
80       if(am == null)
81         pm.acquisitations[i] = null;
82       else
83         pm.acquisitations[i] = new AcquisitionCardModel(am.getAcquisition
               (), am.isVisible());
84     }
85     pm.allowJoinery = allowJoinery;
86     pm.allowPottery = allowPottery;
87     pm.allowBasketMaker = allowBasketMaker;
88     pm.round = round;
89     return pm;
90   }
91
92   public boolean isAllowJoinery() {
93     return allowJoinery;
94   }
95
96   public void setAllowJoinery(boolean allowJoinery) {
97     this.allowJoinery = allowJoinery;
98   }
99
100  public boolean isAllowPottery() {
101    return allowPottery;
102  }
103
104  public void setAllowPottery(boolean allowPottery) {
105    this.allowPottery = allowPottery;
106  }
107
108  public boolean isAllowBasketMaker() {
109    return allowBasketMaker;
110  }
111
112  public void setAllowBasketMaker(boolean allowBasketMaker) {
113    this.allowBasketMaker = allowBasketMaker;
114  }
115
116  public PlayerFieldModel() {
117    this(Player.BLUE);
118  }
119
```

```
120   public PlayerFieldModel(Player player) {
121     this.player = player;
122     fields = new SmallFieldModel[15];
123     for(int i=0; i<15; i++) {
124       fields[i] = new SmallFieldModel(player);
125     }
126     fields[4].setField(FieldCard.F_WOOD_HOUSE);
127     fields[4].setPersonsCount(1);
128     setChild(4);
129     fields[9].setField(FieldCard.F_WOOD_HOUSE);
130     fields[9].setPersonsCount(1);
131     setChild(9);
132   }
133
134   public void addAcquisitation(AcquisitionCardModel acquisition) {
135     for(int i=0; i<10; i++) {
136       if(acquisitations[i] == null) {
137         acquisitations[i] = acquisition;
138         break;
139       }
140     }
141   }
142
143   public static enum AcquisitationType {
144     CHANGE, BACK_BREAD, HARVEST_SEASON
145   }
146
147   /**
148    * @return true, if one more card could get selected; otherwise false
149    * */
150   public boolean selectAcquisitationCards(AcquisitationType type) {
151     boolean ret = false;
152     for(int i=0; i<10; i++) {
153       AcquisitionCardModel m = acquisitations[i];
154       if(m == null)
155         break;
156       BigAcquisitions ba = m.getAcquisition();
157       if(( ba ==  BigAcquisitions.BA_FIRE_PLACE ||
158          ba == BigAcquisitions.BA_FIRE_PLACE2 ||
159          ba == BigAcquisitions.BA_COOKERY ||
160          ba == BigAcquisitions.BA_COOKERY2
161           ) && (type == AcquisitationType.CHANGE || type ==
                  AcquisitationType.BACK_BREAD || type == AcquisitationType.
                  HARVEST_SEASON)) {
162         m.setSelectable(true);
163         ret = true;
164       } else if(( ba == BigAcquisitions.BA_CLAY_OVEN ||
165                ba == BigAcquisitions.BA_STONE_OVEN
166           ) && (type == AcquisitationType.BACK_BREAD)) {
167         m.setSelectable(true);
168         ret = true;
169       } else if(( ba == BigAcquisitions.BA_JOINERY && allowJoinery ||
170                ba == BigAcquisitions.BA_POTTERY && allowPottery ||
171                ba == BigAcquisitions.BA_BASKET_MAKER && allowBasketMaker
172           ) && type == AcquisitationType.HARVEST_SEASON) {
173         m.setSelectable(true);
174         ret = true;
175       } else {
176         m.setSelectable(false);
177       }
178     }
```

```
179     return ret;
180   }
181
182   public AcquisitionCardModel getAcquisitation(int number) {
183     return acquisitations[number];
184   }
185
186   public SmallFieldModel getField(int fieldId) {
187     return fields[fieldId];
188   }
189
190   public void setChild(int fieldId) {
191     switch(player) {
192     case BLUE: fields[fieldId].setChild(Child.C_BLUE); break;
193     case GREEN: fields[fieldId].setChild(Child.C_GREEN); break;
194     case RED: fields[fieldId].setChild(Child.C_RED); break;
195     case ROSA: fields[fieldId].setChild(Child.C_ROSA); break;
196     case NONE: fields[fieldId].setChild(Child.C_NONE); break;
197     }
198   }
199
200   public boolean isHouse(int fieldId) {
201     return fieldId >= 0 && fieldId < 15 &&
202         (fields[fieldId].getField() == FieldCard.F_WOOD_HOUSE || fields[
                fieldId].getField() == FieldCard.F_CLAY_HOUSE || fields[
                fieldId].getField() == FieldCard.F_STONE_HOUSE);
203   }
204
205   public int countFields() {
206     int sum = 0;
207     for(int i=0; i<15; i++) {
208       if(fields[i].getField() == FieldCard.F_FIELD) {
209         sum++;
210       }
211     }
212     return sum;
213   }
214
215   public int countPastures() {
216     int sum = 0;
217     for(int i=0; i<15; i++) {
218       if(isPasture(i)) {
219         sum++;
220       }
221     }
222     return sum;
223   }
224
225   public int countGrainsOnFields() {
226     int sum = 0;
227     for(int i=0; i<15; i++) {
228       if(fields[i].getRessource() == Resource.R_GRAIN && fields[i].
              getField() == FieldCard.F_FIELD) {
229         sum += fields[i].getRessourceCount();
230       }
231     }
232     return sum;
233   }
234
235   public int countVegetablesOnFields() {
236     int sum = 0;
```

```
237     for(int i=0; i<15; i++) {
238       if(fields[i].getRessource() == Resource.R_VEGETABLE && fields[i].
              getField() == FieldCard.F_FIELD) {
239         sum += fields[i].getRessourceCount();
240       }
241     }
242     return sum;
243   }
244
245   public int countSheep() {
246     int sum = 0;
247     for(int i=0; i<15; i++) {
248       if(fields[i].getRessource() == Resource.R_SHEEP && isPasture(i)) {
249         sum += fields[i].getRessourceCount();
250       }
251     }
252     return sum;
253   }
254
255   public int countBoars() {
256     int sum = 0;
257     for(int i=0; i<15; i++) {
258       if(fields[i].getRessource() == Resource.R_BOAR && isPasture(i)) {
259         sum += fields[i].getRessourceCount();
260       }
261     }
262     return sum;
263   }
264
265   public int countCows() {
266     int sum = 0;
267     for(int i=0; i<15; i++) {
268       if(fields[i].getRessource() == Resource.R_COW && isPasture(i)) {
269         sum += fields[i].getRessourceCount();
270       }
271     }
272     return sum;
273   }
274
275   public int countUnusedFields() {
276     int sum = 0;
277     for(int i=0; i<15; i++) {
278       if(fields[i].getField() == FieldCard.F_NONE && fields[i].
              getBottomFence() == 0 && fields[i].getLeftFence() == 0
279           && fields[i].getRightFence() == 0 && fields[i].getTopFence() ==
                  0) {
280         sum++;
281       }
282     }
283     return sum;
284   }
285
286   public int countFencedStables() {
287     int sum = 0;
288     for(int i=0; i<15; i++) {
289       if(fields[i].getField() == FieldCard.F_STABLE && isPasture(i)) {
290         sum++;
291       }
292     }
293     return sum;
294   }
```

```
295
296   public int countClayHouses() {
297     int sum = 0;
298     for(int i=0; i<15; i++) {
299       if(fields[i].getField() == FieldCard.F_CLAY_HOUSE) {
300         sum++;
301       }
302     }
303     return sum;
304   }
305
306   public int countStoneHouses() {
307     int sum = 0;
308     for(int i=0; i<15; i++) {
309       if(fields[i].getField() == FieldCard.F_STONE_HOUSE) {
310         sum++;
311       }
312     }
313     return sum;
314   }
315
316   public int countPersons() {
317     int sum = 0;
318     for(int i=0; i<15; i++) {
319       if(isHouse(i)) {
320         sum+=fields[i].getPersonsCount();
321       }
322     }
323     return sum;
324   }
325
326   public int countCardPoints() {
327     int sum = 0;
328     for(int i=0; i<10; i++) {
329       if(acquisitations[i] == null)
330         break;
331       AcquisitionCardModel a = acquisitations[i];
332       switch(a.getAcquisition()) {
333       case BA_FIRE_PLACE:
334       case BA_FIRE_PLACE2:
335       case BA_COOKERY:
336       case BA_COOKERY2:
337         sum++; break;
338       case BA_CLAY_OVEN:
339       case BA_JOINERY:
340       case BA_POTTERY:
341       case BA_BASKET_MAKER:
342         sum += 2; break;
343       case BA_STONE_OVEN:
344         sum += 3; break;
345       case BA_FOUNTAIN:
346         sum += 4; break;
347       case BA_NONE: break;
348       }
349     }
350     return sum;
351   }
352
353   public void sendPersonToWork() {
354     for(int i=0; i<15; i++) {
355       if(fields[i].getPersonsCount() - fields[i].getPersonsAtWork() > 0) {
```

```
356           fields[i].setPersonsAtWork(fields[i].getPersonsAtWork()+1);
357           break;
358         }
359       }
360     }
361
362     public void sendAllPersonsHome() {
363       for(int i=0; i<15; i++) {
364         fields[i].setPersonsAtWork(0);
365       }
366     }
367
368     public boolean isPersonAvailableForWork() {
369       for(int i=0; i<15; i++) {
370         if(fields[i].getPersonsCount() - fields[i].getPersonsAtWork() > 0) {
371           return true;
372         }
373       }
374       return false;
375     }
376
377     public void nextFence() {
378       fenceId++;
379     }
380
381     public boolean isField(int fieldId) {
382       return fieldId >= 0 && fieldId < 15 && fields[fieldId].getField() ==
              FieldCard.F_FIELD;
383     }
384
385     public boolean hasLeftFence(int fieldId) {
386       return fieldId >= 0 && fieldId < 15 && fields[fieldId].getLeftFence()
              == fenceId;
387     }
388     public boolean hasRightFence(int fieldId) {
389       return fieldId >= 0 && fieldId < 15 && fields[fieldId].getRightFence()
               == fenceId;
390     }
391     public boolean hasTopFence(int fieldId) {
392       return fieldId >= 0 && fieldId < 15 && fields[fieldId].getTopFence()
              == fenceId;
393     }
394     public boolean hasBottomFence(int fieldId) {
395       return fieldId >= 0 && fieldId < 15 && fields[fieldId].getBottomFence
              () == fenceId;
396     }
397
398     public void setLeftFence(int fieldId, boolean value) {
399       if(fieldId >= 0 && fieldId < 15)
400         fields[fieldId].setLeftFence(value?fenceId:0);
401     }
402     public void setRightFence(int fieldId, boolean value) {
403       if(fieldId >= 0 && fieldId < 15)
404         fields[fieldId].setRightFence(value?fenceId:0);
405     }
406     public void setTopFence(int fieldId, boolean value) {
407       if(fieldId >= 0 && fieldId < 15)
408         fields[fieldId].setTopFence(value?fenceId:0);
409     }
410     public void setBottomFence(int fieldId, boolean value) {
411       if(fieldId >= 0 && fieldId < 15)
```

```
412       fields[fieldId].setBottomFence(value?fenceId:0);
413     }
414
415     public static class PastureInfo {
416       public static enum Animal {
417         NONE, SHEEP, COW, BOAR
418       }
419       public Animal animal = Animal.NONE;
420       public int animalCount = 0;
421       public int maxAnimals = 0;
422     }
423
424     private int[] _fillUpEmptyPastureFields(int[] pastureId) {
425       for(int i=0; i<15; i++) {
426         if(pastureId[i] == 0) { // if this field has no pasture id, then we
              look if has no fence to the other pasture field, and give this
              field the same pasture id
427           if(i>0 && !_isLeftFence(i) && pastureId[i-1] > 0 && i/5 == (i-1)
                /5)
428             pastureId[i] = pastureId[i-1];
429           else if(i<14 && !_isRightFence(i) && pastureId[i+1] > 0 && i/5 ==
                (i+1)/5)
430             pastureId[i] = pastureId[i+1];
431           else if(i>4 && !_isTopFence(i) && pastureId[i-5] > 0)
432             pastureId[i] = pastureId[i-5];
433           else if(i<10 && !_isBottomFence(i) && pastureId[i+5] > 0)
434             pastureId[i] = pastureId[i+5];
435         }
436       }
437       return pastureId;
438     }
439
440     public int[] categorizePasture() {
441       int[] pastureId = new int[] {0,0,0,0,0,
442                     0,0,0,0,0,
443                     0,0,0,0,0};
444       for(int i=0; i<15; i++) {
445         if(fields[i].getBottomFence() > 0)
446           pastureId[i] = fields[i].getBottomFence();
447         else if(fields[i].getLeftFence() > 0)
448           pastureId[i] = fields[i].getLeftFence();
449         else if(fields[i].getRightFence() > 0)
450           pastureId[i] = fields[i].getRightFence();
451         else if(fields[i].getTopFence() > 0)
452           pastureId[i] = fields[i].getTopFence();
453       }
454       int pastureMaxId = 100;
455       pastureId = _fillUpEmptyPastureFields(pastureId);
456
457       for(int i=0; i<15; i++) {
458         if(isPasture(i) && pastureId[i] == 0) {
459           pastureId[i] = pastureMaxId; // it is a pasture field, but has no
                fence on its field - it is surrendered by fences of other
                pasture fields
460           pastureMaxId++;
461           // now we need to set the pastureId of the other fields belonging
                to this pasture in the loop
462           for(int k=0; k<15; k++) {
463             pastureId = _fillUpEmptyPastureFields(pastureId);
464           }
465         }
```

```
466     }
467     return pastureId;
468   }
469
470   public void breedAnimals() {
471     int[] pastureId = categorizePasture();
472
473     ArrayList<Integer> doneIds = new ArrayList<Integer>();
474     for(int i=0; i<15; i++) {
475       int pId = pastureId[i];
476       if(pId != 0 && !doneIds.contains(pId)) {
477         doneIds.add(pId);
478         PastureInfo pInfo = getPastureInfo(i, pastureId);
479         int babies = pInfo.animalCount / 2;
480         if(pInfo.animalCount + babies > pInfo.maxAnimals) {
481           babies = pInfo.maxAnimals - pInfo.animalCount;
482         }
483         fields[i].setRessourceCount(fields[i].getRessourceCount() + babies
               );
484         Resource res = Resource.R_NONE;
485         switch(pInfo.animal) {
486         case BOAR: res = Resource.R_BOAR; break;
487         case COW: res = Resource.R_COW; break;
488         case SHEEP: res = Resource.R_SHEEP; break;
489         case NONE: break;
490         }
491         fields[i].setRessource(res);
492       }
493     }
494   }
495
496   public boolean containsAnimals(int fieldId) {
497     return fields[fieldId].getRessourceCount() > 0 && (fields[fieldId].
          getRessource() == Resource.R_BOAR ||
498         fields[fieldId].getRessource() == Resource.R_COW || fields[fieldId
              ].getRessource() == Resource.R_SHEEP);
499   }
500
501   public PastureInfo getPastureInfo(int fieldId, int[] pastureId) {
502     PastureInfo past = new PastureInfo();
503     if(isPasture(fieldId)) {
504       // now we have categorized the pastures with id
505       // next we want to count the animals of the fields with the given
             pastureId
506       // and set also the maximum allowed amount of animals, depending on
             the pasture's size and the amount of stables
507       int stables = 0;
508       int pastureSize = 0;
509       int animals = 0;
510       Animal animal = Animal.NONE;
511       int pId = pastureId[fieldId];
512       for(int i=0; i<15; i++) {
513         if(pastureId[i] == pId) {
514           pastureSize++;
515           if(fields[i].getField() == FieldCard.F_STABLE)
516             stables++;
517           switch(fields[i].getRessource()) {
518           case R_BOAR: animal = Animal.BOAR; break;
519           case R_COW: animal = Animal.COW; break;
520           case R_SHEEP: animal = Animal.SHEEP; break;
521           default: break;
```

```
522             }
523             animals += fields[i].getRessourceCount();
524           }
525         }
526         past.animal = animal;
527         past.animalCount = animals;
528         past.maxAnimals = pastureSize * 2;
529         for(int i=0; i<stables; i++)
530           past.maxAnimals *= 2;
531       } else if(fields[fieldId].getField() == FieldCard.F_STABLE) {
532         past.animalCount = fields[fieldId].getRessourceCount();
533         past.maxAnimals = 1;
534       }
535       return past;
536     }
537
538     // TODO this function does not work for every case
539     public boolean isPasture(int fieldId) {
540       boolean retTop = false; // true if the field has a top fence
541       for(int i=fieldId; i >= 0 && !retTop; i-=5) {
542         retTop = _isTopFence(i);
543       }
544       boolean retBottom = false; // true if the field has a bottom fence
545       for(int i=fieldId; i < 15 && !retBottom; i+=5) {
546         retBottom = _isBottomFence(i);
547       }
548       boolean retLeft = false; // true if the field has a left fence
549       for(int i=fieldId; i >= 0 && !retLeft && i/5 == fieldId /5; i--) {
550         retLeft = _isLeftFence(i);
551       }
552       boolean retRight = false; // true if the field has a right fence
553       for(int i=fieldId; i < 15 && !retRight && i/5 == fieldId /5; i++) {
554         retRight = _isRightFence(i);
555       }
556       return retBottom && retLeft && retRight && retTop;
557     }
558
559     private boolean _isTopFence(int fieldId) {
560       return fieldId >=0 && fieldId < 15 && fields[fieldId].getTopFence() >
             0 || fieldId >= 5 && fieldId < 20 && fields[fieldId -5].
             getBottomFence() > 0;
561     }
562     private boolean _isBottomFence(int fieldId) {
563       return fieldId >=0 && fieldId < 15 && fields[fieldId].getBottomFence()
              > 0 || fieldId >= -5 && fieldId < 10 && fields[fieldId+5].
             getTopFence() > 0;
564     }
565     private boolean _isLeftFence(int fieldId) {
566       return fieldId >=0 && fieldId < 15 && fields[fieldId].getLeftFence() >
              0 || fieldId >= 1 && fieldId < 16 && fieldId/5 == (fieldId -1)/5
             && fields[fieldId -1].getRightFence() > 0;
567     }
568     private boolean _isRightFence(int fieldId) {
569       return fieldId >=0 && fieldId < 15 && fields[fieldId].getRightFence()
             > 0 || fieldId >= -1 && fieldId < 14 && fieldId/5 == (fieldId+1)/5
              && fields[fieldId+1].getLeftFence() > 0;
570     }
571 }
```

Listing A62: PlayerFieldModel.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  import java.io.Serializable;
4
5  import com.google.gwt.event.shared.HandlerManager;
6  import com.google.gwt.event.shared.HandlerRegistration;
7
8  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       ResourceModelChangedEvent;
9  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       ResourceModelChangedEvent.HasResourceModelChangedHandler;
10 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       ResourceModelChangedEvent.ResourceModelChangedHandler;
11
12 public class PlayerResourceModel implements HasResourceModelChangedHandler
       , Serializable {
13   private static final long serialVersionUID = 1L;
14   private int woodCount = 0;
15   private int clayCount = 0;
16   private int stoneCount = 0;
17   private int reedCount = 0;
18   private int foodCount = 3;
19   private int grainCount = 0;
20   private int vegetableCount = 0;
21   private int beggerCards = 0;
22   private int fenceCount = 15;
23   private int personsCount = 3;
24   private int stableCount = 4;
25
26   public void update(PlayerResourceModel model) {
27     if(model == null) return;
28     this.woodCount = model.woodCount;
29     this.clayCount = model.clayCount;
30     this.stoneCount = model.stoneCount;
31     this.reedCount = model.reedCount;
32     this.foodCount = model.foodCount;
33     this.grainCount = model.grainCount;
34     this.vegetableCount = model.vegetableCount;
35     this.beggerCards = model.beggerCards;
36     this.fenceCount = model.fenceCount;
37     this.personsCount = model.personsCount;
38     this.stableCount = model.stableCount;
39   }
40
41   public boolean equals(Object o) {
42     if( !(o instanceof PlayerResourceModel))
43       return false;
44     PlayerResourceModel pm = (PlayerResourceModel)o;
45     return woodCount == pm.woodCount && clayCount == pm.clayCount &&
           stoneCount == pm.stoneCount && reedCount == pm.reedCount &&
           foodCount == pm.foodCount &&
46         grainCount == pm.grainCount && vegetableCount == pm.vegetableCount
               && beggerCards == pm.beggerCards && fenceCount == pm.
               fenceCount &&
47         personsCount == pm.personsCount && stableCount == pm.stableCount;
48   }
49
50   public PlayerResourceModel clone() {
51     PlayerResourceModel pm = new PlayerResourceModel();
52     pm.woodCount = woodCount;
53     pm.clayCount = clayCount;
```

```
54      pm.stoneCount = stoneCount;
55      pm.reedCount = reedCount;
56      pm.foodCount = foodCount;
57      pm.grainCount = grainCount;
58      pm.vegetableCount = vegetableCount;
59      pm.beggerCards = beggerCards;
60      pm.fenceCount = fenceCount;
61      pm.personsCount = personsCount;
62      pm.stableCount = stableCount;
63      return pm;
64    }
65
66    public int getFenceCount() {
67      return fenceCount;
68    }
69
70    public void setFenceCount(int fenceCount) {
71      this.fenceCount = fenceCount;
72      fireModelChanged();
73    }
74
75    public int getPersonsCount() {
76      return personsCount;
77    }
78
79    public void setPersonsCount(int personsCount) {
80      this.personsCount = personsCount;
81      fireModelChanged();
82    }
83
84    public int getStableCount() {
85      return stableCount;
86    }
87
88    public void setStableCount(int stableCount) {
89      this.stableCount = stableCount;
90      fireModelChanged();
91    }
92
93    private static HandlerManager eventBus = new HandlerManager(null);
94
95    public void addRessource(Resource res, int count) {
96      boolean valid = true;
97      switch(res) {
98      case R_WOOD: woodCount += count; break;
99      case R_CLAY: clayCount += count; break;
100     case R_STONE: stoneCount += count; break;
101     case R_REED: reedCount += count; break;
102     case R_FOOD: foodCount += count;
103       if(foodCount < 0) {
104         beggerCards += -1*foodCount;
105         foodCount = 0;
106       }
107       break;
108     case R_GRAIN: grainCount += count; break;
109     case R_VEGETABLE: vegetableCount += count; break;
110     default: valid = false; break;
111     }
112     if(valid)
113       fireModelChanged();
114   }
```

```
115
116   public int getWoodCount() {
117     return woodCount;
118   }
119   public void setWoodCount(int woodCount) {
120     this.woodCount = woodCount;
121     fireModelChanged();
122   }
123   public int getClayCount() {
124     return clayCount;
125   }
126   public void setClayCount(int clayCount) {
127     this.clayCount = clayCount;
128     fireModelChanged();
129   }
130   public int getStoneCount() {
131     return stoneCount;
132   }
133   public void setStoneCount(int stoneCount) {
134     this.stoneCount = stoneCount;
135     fireModelChanged();
136   }
137   public int getReedCount() {
138     return reedCount;
139   }
140   public void setReedCount(int reedCount) {
141     this.reedCount = reedCount;
142     fireModelChanged();
143   }
144   public int getFoodCount() {
145     return foodCount;
146   }
147   public void setFoodCount(int foodCount) {
148     this.foodCount = foodCount;
149     if(this.foodCount < 0) {
150       beggerCards += -1*this.foodCount;
151       this.foodCount = 0;
152     }
153     fireModelChanged();
154   }
155   public int getBeggerCards() {
156     return beggerCards;
157   }
158
159   public int getGrainCount() {
160     return grainCount;
161   }
162   public void setGrainCount(int grainCount) {
163     this.grainCount = grainCount;
164     fireModelChanged();
165   }
166   public int getVegetableCount() {
167     return vegetableCount;
168   }
169   public void setVegetableCount(int vegetableCount) {
170     this.vegetableCount = vegetableCount;
171     fireModelChanged();
172   }
173
174   private void fireModelChanged() {
175     eventBus.fireEvent(new ResourceModelChangedEvent());
```

```
176   }
177
178   @Override
179   public HandlerRegistration addResourceModelChangedHandler(
          ResourceModelChangedHandler handler) {
180     return eventBus.addHandler(ResourceModelChangedEvent.getType(),
            handler);
181   }
182
183
184 }
```

Listing A63: PlayerResourceModel.java file

```
1 package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3 /** added the prefix R_ in order to this enum class with the
      BackgroundCard enum class in the UiBinder xml file */
4 public enum Resource {
5   R_NONE, R_VEGETABLE, R_WOOD, R_CLAY, R_STONE, R_GRAIN, R_SHEEP, R_COW,
        R_BOAR, R_FOOD, R_REED
6 }
```

Listing A64: Resource.java file

```
1 package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3 import java.io.Serializable;
4
5 public class Rounds1To7Model implements Serializable {
6   private static final long serialVersionUID = 1L;
7   private BigFieldModel modelSheep = new BigFieldModel(BackgroundCard.
        SHEEP, Resource.R_SHEEP, 1, 1, "get␣sheep", true);
8   private BigFieldModel modelBigAcquisitation = new BigFieldModel(
        BackgroundCard.ACQUISITION, Resource.R_NONE, 0, 0, "get␣one␣big␣
        acquisitation", false);
9   private BigFieldModel modelFence = new BigFieldModel(BackgroundCard.
        FENCE, Resource.R_NONE, 0, 0, "build␣your␣onw␣fence␣(1␣wood␣for␣each
        ␣fence␣part)", false);
10  private BigFieldModel modelSeedingBacking = new BigFieldModel(
        BackgroundCard.SEEDING_BACKING, Resource.R_NONE, 0, 0, "seed␣grains␣
        or␣vegetables␣on␣your␣fields␣and/or␣back␣bread", false);
11  private BigFieldModel modelFamily = new BigFieldModel(BackgroundCard.
        FAMILIY_ADDITION2, Resource.R_NONE, 0, 0, "make␣a␣baby␣(you␣need␣one
        ␣free␣room)", false);
12  private BigFieldModel modelStone = new BigFieldModel(BackgroundCard.
        STONE2, Resource.R_STONE, 1, 1, "get␣stone", false);
13  private BigFieldModel modelRestauration = new BigFieldModel(
        BackgroundCard.RESTAURATION, Resource.R_NONE, 0, 0, "restaurate␣your
        ␣homes␣and␣if␣wanted␣you␣can␣do␣a␣big␣acquisitation␣later␣on", false
        );
14
15  private BigFieldModel modelWood = new BigFieldModel(BackgroundCard.NONE,
        Resource.R_WOOD, 3, 3, "get␣wood", true);
16  private BigFieldModel modelClay = new BigFieldModel(BackgroundCard.NONE,
        Resource.R_CLAY, 1, 1, "get␣clay", true);
17  private BigFieldModel modelReed = new BigFieldModel(BackgroundCard.NONE,
        Resource.R_REED, 1, 1, "get␣reed", true);
18  private BigFieldModel modelFood = new BigFieldModel(BackgroundCard.NONE,
        Resource.R_FOOD, 1, 1, "get␣food␣markers", true);
19
```

```
20
21   public BigFieldModel getModelSheep() {
22     return modelSheep;
23   }
24
25   public BigFieldModel getModelBigAcquisitation() {
26     return modelBigAcquisitation;
27   }
28
29   public BigFieldModel getModelFence() {
30     return modelFence;
31   }
32
33   public BigFieldModel getModelSeedingBacking() {
34     return modelSeedingBacking;
35   }
36
37   public BigFieldModel getModelFamily() {
38     return modelFamily;
39   }
40
41   public BigFieldModel getModelStone() {
42     return modelStone;
43   }
44
45   public BigFieldModel getModelRestauration() {
46     return modelRestauration;
47   }
48
49   public BigFieldModel getModelWood() {
50     return modelWood;
51   }
52
53   public BigFieldModel getModelClay() {
54     return modelClay;
55   }
56
57   public BigFieldModel getModelReed() {
58     return modelReed;
59   }
60
61   public BigFieldModel getModelFood() {
62     return modelFood;
63   }
64
65   @Override
66   public boolean equals(Object o) {
67     if( !(o instanceof Rounds1To7Model) )
68       return false;
69     Rounds1To7Model cm = (Rounds1To7Model)o;
70     return modelSheep.equals(cm.getModelSheep()) && modelBigAcquisitation.
         equals(cm.getModelBigAcquisitation()) && modelFence.equals(cm.
         getModelFence())
71         && modelSeedingBacking.equals(cm.getModelSeedingBacking()) &&
              modelFamily.equals(cm.getModelFamily()) &&
72         modelStone.equals(cm.getModelStone()) && modelRestauration.equals(
              cm.getModelRestauration()) && modelWood.equals(cm.getModelWood
              ())
73         && modelClay.equals(cm.getModelClay()) && modelReed.equals(cm.
              getModelReed()) && modelFood.equals(cm.getModelFood());
74   }
```

```
75
76    public void update(Rounds1To7Model model) {
77      if(model == null) return;
78      _cloneBM(this.getModelSheep(), model.getModelSheep());
79      _cloneBM(this.getModelBigAcquisitation(), model.
            getModelBigAcquisitation());
80      _cloneBM(this.getModelFence(), model.getModelFence());
81      _cloneBM(this.getModelSeedingBacking(), model.getModelSeedingBacking()
            );
82      _cloneBM(this.getModelFamily(), model.getModelFamily());
83      _cloneBM(this.getModelStone(), model.getModelStone());
84      _cloneBM(this.getModelRestauration(), model.getModelRestauration());
85      _cloneBM(this.getModelWood(), model.getModelWood());
86      _cloneBM(this.getModelClay(), model.getModelClay());
87      _cloneBM(this.getModelReed(), model.getModelReed());
88      _cloneBM(this.getModelFood(), model.getModelFood());
89    }
90
91    public Rounds1To7Model clone() {
92      Rounds1To7Model model = new Rounds1To7Model();
93      _cloneBM(model.getModelSheep(), this.getModelSheep());
94      _cloneBM(model.getModelBigAcquisitation(), this.
            getModelBigAcquisitation());
95      _cloneBM(model.getModelFence(), this.getModelFence());
96      _cloneBM(model.getModelSeedingBacking(), this.getModelSeedingBacking()
            );
97      _cloneBM(model.getModelFamily(), this.getModelFamily());
98      _cloneBM(model.getModelStone(), this.getModelStone());
99      _cloneBM(model.getModelRestauration(), this.getModelRestauration());
100     _cloneBM(model.getModelWood(), this.getModelWood());
101     _cloneBM(model.getModelClay(), this.getModelClay());
102     _cloneBM(model.getModelReed(), this.getModelReed());
103     _cloneBM(model.getModelFood(), this.getModelFood());
104     return model;
105   }
106
107   private void _cloneBM(BigFieldModel bm, BigFieldModel org) {
108 //      bm.setBgCard(org.getBgCard());
109     bm.setChild(org.getChild());
110 //      bm.setDescription(org.getDescription());
111 //      bm.setId(org.getId());
112     bm.setRessource(org.getRessource());
113     bm.setRessourceCount(org.getRessourceCount());
114     bm.setVisible(org.isVisible());
115   }
116 }
```

Listing A65: Rounds1To7Model.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  import java.io.Serializable;
4
5  public class Rounds8To14Model implements Serializable {
6    private static final long serialVersionUID = 1L;
7    private BigFieldModel modelBoar = new BigFieldModel(BackgroundCard.BOAR,
         Resource.R_BOAR, 1, 1, "get␣boar", false);
8    private BigFieldModel modelVegetable = new BigFieldModel(BackgroundCard.
       VEGETABLE, Resource.R_NONE, 0, 0, "get␣one␣vegetable", false);
9    private BigFieldModel modelStone = new BigFieldModel(BackgroundCard.
       STONE4, Resource.R_STONE, 1, 1, "get␣stone", false);
10   private BigFieldModel modelCow = new BigFieldModel(BackgroundCard.COW,
         Resource.R_COW, 1, 1, "get␣cow", false);
11   private BigFieldModel modelPlowSow = new BigFieldModel(BackgroundCard.
       PLOWING_SOWING, Resource.R_NONE, 0, 0, "plow␣one␣field\nand/or\nseed
       ␣grains␣or␣vegetables␣on␣your␣fields", false);
12   private BigFieldModel modelFamily = new BigFieldModel(BackgroundCard.
       FAMILY_ADDITION5, Resource.R_NONE, 0, 0, "make␣a␣baby␣(you␣do␣NOT␣
       need␣a␣free␣room)", false);
13   private BigFieldModel modelRestauration = new BigFieldModel(
         BackgroundCard.RESTAURATION_FENCE, Resource.R_NONE, 0, 0, "
         restaurate␣your␣homes␣\nand␣if␣wanted␣you␣can\nbuild␣your␣onw␣fence␣
         (1␣wood␣for␣each␣fence␣part)", false);
14
15
16   public BigFieldModel getModelBoar() {
17     return modelBoar;
18   }
19
20   public BigFieldModel getModelVegetable() {
21     return modelVegetable;
22   }
23
24   public BigFieldModel getModelStone() {
25     return modelStone;
26   }
27
28   public BigFieldModel getModelCow() {
29     return modelCow;
30   }
31
32   public BigFieldModel getModelPlowSow() {
33     return modelPlowSow;
34   }
35
36   public BigFieldModel getModelFamily() {
37     return modelFamily;
38   }
39
40   public BigFieldModel getModelRestauration() {
41     return modelRestauration;
42   }
43
44   @Override
45   public boolean equals(Object o) {
46     if( !(o instanceof Rounds8To14Model) )
47       return false;
48     Rounds8To14Model cm = (Rounds8To14Model)o;
49     return modelBoar.equals(cm.getModelBoar()) && modelVegetable.equals(cm
         .getModelVegetable()) && modelStone.equals(cm.getModelStone())
```

```
50          && modelCow.equals(cm.getModelCow()) && modelPlowSow.equals(cm.
                getModelPlowSow()) &&
51          modelFamily.equals(cm.getModelFamily()) && modelRestauration.
                equals(cm.getModelRestauration());
52    }
53
54    public void update(Rounds8To14Model model) {
55      if(model == null) return;
56      _cloneBM(this.getModelBoar(), model.getModelBoar());
57      _cloneBM(this.getModelVegetable(), model.getModelVegetable());
58      _cloneBM(this.getModelStone(), model.getModelStone());
59      _cloneBM(this.getModelCow(), model.getModelCow());
60      _cloneBM(this.getModelPlowSow(), model.getModelPlowSow());
61      _cloneBM(this.getModelFamily(), model.getModelFamily());
62      _cloneBM(this.getModelRestauration(), model.getModelRestauration());
63    }
64
65    public Rounds8To14Model clone() {
66      Rounds8To14Model model = new Rounds8To14Model();
67      _cloneBM(model.getModelBoar(), this.getModelBoar());
68      _cloneBM(model.getModelVegetable(), this.getModelVegetable());
69      _cloneBM(model.getModelStone(), this.getModelStone());
70      _cloneBM(model.getModelCow(), this.getModelCow());
71      _cloneBM(model.getModelPlowSow(), this.getModelPlowSow());
72      _cloneBM(model.getModelFamily(), this.getModelFamily());
73      _cloneBM(model.getModelRestauration(), this.getModelRestauration());
74      return model;
75    }
76
77    private void _cloneBM(BigFieldModel bm, BigFieldModel org) {
78  //      bm.setBgCard(org.getBgCard());
79      bm.setChild(org.getChild());
80  //      bm.setDescription(org.getDescription());
81  //      bm.setId(org.getId());
82      bm.setRessource(org.getRessource());
83      bm.setRessourceCount(org.getRessourceCount());
84      bm.setVisible(org.isVisible());
85    }
86  }
```

Listing A66: Rounds8To14Model.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.model;
2
3  import java.io.Serializable;
4
5  public class SmallFieldModel extends BaseFieldModel implements
       Serializable {
6    private static final long serialVersionUID = 1L;
7    private FieldCard field = FieldCard.F_NONE;
8    private int leftFence = 0;
9    private int rightFence = 0;
10   private int topFence = 0;
11   private int bottomFence = 0;
12   private boolean selectable = false;
13   private int personsCount = 0;
14   private int personsAtWork = 0;
15   private transient Player player;
16
17   public SmallFieldModel(FieldCard field, int leftFence, int rightFence,
         int topFence, int bottomFence, boolean selectable, int personsCount,
          int personsAtWork,
18       Resource ressource, int ressourceCount, Child child, String
             description, String id, Player player) {
19     this.field = field; this.leftFence = leftFence; this.rightFence =
           rightFence; this.topFence = topFence; this.bottomFence =
           bottomFence;
20     this.selectable = selectable; this.personsCount = personsCount; this.
           personsAtWork = personsAtWork; this.player = player;
21     setRessource(ressource); setRessourceCount(ressourceCount); setChild(
           child); setDescription(description); setId(id);
22   }
23
24   public boolean equals(Object o) {
25     if( !(o instanceof SmallFieldModel))
26       return false;
27     SmallFieldModel sm = (SmallFieldModel)o;
28     return super.equals((BaseFieldModel)o) && field == sm.field &&
           leftFence == sm.leftFence && rightFence == sm.rightFence &&
           topFence == sm.topFence && bottomFence == sm.bottomFence
29         && selectable == sm.selectable && personsCount == sm.personsCount
               && personsAtWork == sm.personsAtWork && player == sm.player;
30   }
31
32   public int getChildCount() {
33     return childCount;
34   }
35   public void setChildCount(int childCount) {
36     this.childCount = childCount;
37   }
38   private int childCount = 0;
39   public int getPersonsAtWork() {
40     return personsAtWork;
41   }
42   public void setPersonsAtWork(int parentsAtWork) {
43     this.personsAtWork = parentsAtWork;
44   }
45   public int getPersonsCount() {
46     return personsCount;
47   }
48   public void setPersonsCount(int personsCount) {
49     this.personsCount = personsCount;
50   }
```

```
51    public boolean isSelectable() {
52      return selectable;
53    }
54    public void setSelectable(boolean selectable) {
55      this.selectable = selectable;
56    }
57
58    public SmallFieldModel() {}
59    public SmallFieldModel(Player player) {
60      this.player = player;
61    }
62    public FieldCard getField() {
63      return field;
64    }
65    public void setField(FieldCard field) {
66      this.field = field;
67    }
68    public int getLeftFence() {
69      return leftFence;
70    }
71    public void setLeftFence(int leftFence) {
72      this.leftFence = leftFence;
73    }
74    public int getRightFence() {
75      return rightFence;
76    }
77    public void setRightFence(int rightFence) {
78      this.rightFence = rightFence;
79    }
80    public int getTopFence() {
81      return topFence;
82    }
83    public void setTopFence(int topFence) {
84      this.topFence = topFence;
85    }
86    public int getBottomFence() {
87      return bottomFence;
88    }
89    public void setBottomFence(int bottomFence) {
90      this.bottomFence = bottomFence;
91    }
92    public Player getPlayer() {
93      return player;
94    }
95    public void setPlayer(Player player) {
96      this.player = player;
97    }
98  }
```

Listing A67: SmallFieldModel.java file

de.tu_freiberg.informatik.vonwenckstern.client.presenter package

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.presenter;
2
3  import java.io.Serializable;
4  /**
5   * is a presenter which states should get serialized to the URL, so that
        the back/ forward button can restore the older states
6   * @author wencky
7   *
8   * @param <T>
9   */
10 public interface Activity<T extends Serializable> extends Presenter{
11   /**
12    * the activity key to identify the presenter whose data should get
          loaded or updated; e.g. 1 for CardFieldPresenter, 2 for
          ResourcePresenter
13    * @return the activity key, it must be unique for each object, so if
          you have several objects of one class you should add anything like
          System.currentTimeMillis()
14    * to make it unique
15    */
16   public Type<?> getActivityKey();
17   /**
18    * returns an serializable Object which represents the state of this
          history
19    */
20   public T getActualHistory();
21   /**
22    * updates the component to a special history state
23    * @param state the history state which was created by getActualHistory
          ()
24    */
25   public void setActualHistory(T state);
26
27   public static class Type<A> {
28       private static int nextHashCode;
29       private final int index;
30       private final String presenterValue;
31
32       /**
33        * Constructor.
34        */
35       public Type(String presenterValue) {
36         index = ++nextHashCode;
37         this.presenterValue = presenterValue;
38       }
39
40       @Override
41       public final int hashCode() {
42         return index;
43       }
44
45       @Override
46       public String toString() {
47         return "Activity type: " + presenterValue;
48       }
49     }
50 }
```

Listing A68: Activity.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.presenter;
2
3  import com.google.gwt.event.dom.client.ClickEvent;
4  import com.google.gwt.event.dom.client.ClickHandler;
5  import com.google.gwt.user.client.ui.Widget;
6
7  import de.tu_freiberg.informatik.vonwenckstern.client.EventBus;
8  import de.tu_freiberg.informatik.vonwenckstern.client.HistoryController;
9  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       GetBigAcquisitionEvent;
10 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       HistoryChangedEvent;
11 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       RequestHistoryEvent;
12 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       BigAcquisitationsModel;
13 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       BigAcquisitions;
14 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasAcquisitionCardModel;
15
16 public class BigAcquisitionsPresenter implements Activity<
       BigAcquisitationsModel>, ClickHandler {
17
18   public interface Display {
19     public void update(BigAcquisitationsModel model);
20     public void hideAcquisition(BigAcquisitions aquisition);
21     public void registerHandlers(ClickHandler p);
22     public Widget asWidget();
23   }
24
25   private BigAcquisitationsModel model = null;
26
27   private Display display = null;
28
29   public BigAcquisitionsPresenter(Display display, BigAcquisitationsModel
         model) {
30     this.display = display;
31     display.registerHandlers(this);
32     this.model = model;
33     display.update(model);
34     HistoryController.getInstance().addActivityPresenter(this);
35     EventBus.fire(new RequestHistoryEvent(this));
36   }
37
38   @Override
39   public void onClick(ClickEvent event) {
40     if(event.getSource() instanceof HasAcquisitionCardModel) {
41       EventBus.fire(new GetBigAcquisitionEvent(((HasAcquisitionCardModel)
             event.getSource()).getModel()));
42     }
43   }
44
45   @Override
46   public Widget getView() {
47     return display.asWidget();
48   }
49
50   public void hideAcquisition(BigAcquisitions aquisition) {
51     display.hideAcquisition(aquisition);
52     EventBus.fire(new HistoryChangedEvent(this));
```

```
53    }
54
55    private static Type<BigAcquisitionsPresenter> TYPE = new Type<
          BigAcquisitionsPresenter>("BigAcquisitionsPresenter");
56    @Override
57    public Activity.Type<?> getActivityKey() {
58      return TYPE;
59    }
60
61    @Override
62    public BigAcquisitationsModel getActualHistory() {
63      return model.clone();
64    }
65
66    @Override
67    public void setActualHistory(BigAcquisitationsModel state) {
68      model.update(state);
69      display.update(model);
70    }
71
72  }
```

Listing A69: BigAcquisitionsPresenter.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.presenter;
2
3  import com.google.gwt.user.client.Window;
4
5  import de.tu_freiberg.informatik.vonwenckstern.client.EventBus;
6  import de.tu_freiberg.informatik.vonwenckstern.client.HistoryController;
7  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       AddResourceEvent;
8  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       AddResourceEvent.RessourceItem;
9  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       BuildHouseEvent;
10 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       HistoryChangedEvent;
11 import de.tu_freiberg.informatik.vonwenckstern.client.event.NextRoundEvent
       ;
12 import de.tu_freiberg.informatik.vonwenckstern.client.event.PlowFieldEvent
       ;
13 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       RequestHistoryEvent;
14 import de.tu_freiberg.informatik.vonwenckstern.client.model.BaseFieldModel
       ;
15 import de.tu_freiberg.informatik.vonwenckstern.client.model.BigFieldModel;
16 import de.tu_freiberg.informatik.vonwenckstern.client.model.CardFieldModel
       ;
17 import de.tu_freiberg.informatik.vonwenckstern.client.model.Player;
18 import de.tu_freiberg.informatik.vonwenckstern.client.model.Resource;
19
20 public class CardFieldPresenter extends ResourcePresenter implements
       Activity<CardFieldModel>{
21
22   public interface Display extends ResourcePresenter.Display {
23     public void update(CardFieldModel model);
24   }
25
26   private CardFieldModel model = null;
27
```

```
28  public CardFieldPresenter(Display display, Player player, CardFieldModel
        model) {
29    super(display, player);
30    this.model = model;
31    display.update(model);
32    HistoryController.getInstance().addActivityPresenter(this);
33    EventBus.fire(new RequestHistoryEvent(this));
34  }
35
36  public boolean processAction(BaseFieldModel model) {
37    super.processAction(model);
38    String id = model.getId();
39    if(id == null) id = "";
40    if(id.equals("buildHouse")) {
41      EventBus.fire(new BuildHouseEvent());
42    } else if(id.equals("startPlayer")) {
43      Window.alert("Since␣this␣play␣supports␣only␣single␣player␣mode␣until
            ␣now,␣you␣wasted␣one␣turn.");
44    } else if(id.equals("oneGrain")) {
45      RessourceItem[] items = new RessourceItem[] { new RessourceItem(
            Resource.R_GRAIN, 1)};
46      EventBus.fire(new AddResourceEvent(items));
47    } else if(id.equals("plowField")) {
48      EventBus.fire(new PlowFieldEvent());
49    } else if(id.equals("twoFoodMarkers")) {
50      RessourceItem[] items = new RessourceItem[] { new RessourceItem(
            Resource.R_FOOD, 2)};
51      EventBus.fire(new AddResourceEvent(items));
52    } else if(id.equals("noOperation")) {
53      return false;
54    }else if(model instanceof BigFieldModel) {
55      switch(((BigFieldModel) model).getBgCard()) {
56      case REED_STONE_FOOD: {
57        RessourceItem[] items = new RessourceItem[] { new RessourceItem(
              Resource.R_REED, 1),new RessourceItem(Resource.R_STONE, 1),new
               RessourceItem(Resource.R_FOOD, 1) };
58        EventBus.fire(new AddResourceEvent(items));
59        break;
60      }
61      default: break;
62      }
63    }
64    return true;
65  }
66
67  private static Type<CardFieldPresenter> TYPE = new Type<
        CardFieldPresenter>("CardFieldPresenter");
68  @Override
69  public Activity.Type<?> getActivityKey() {
70    return TYPE;
71  }
72
73  @Override
74  public CardFieldModel getActualHistory() {
75    return model.clone(); // need to return a clone, otherwise the equals
          comparison in the history controller is every time true
76  }
77
78  @Override
79  public void setActualHistory(CardFieldModel state) {
80    model.update(state);
```

```
81      ((Display)display).update(model);
82    }
83
84    @Override
85    public void onNextRound(NextRoundEvent event) {
86      super.onNextRound(event);
87      EventBus.fire(new HistoryChangedEvent(this));
88    }
89  }
```

Listing A70: CardFieldPresenter.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.presenter;
2
3  import com.google.gwt.user.client.ui.Widget;
4
5  import de.tu_freiberg.informatik.vonwenckstern.client.EventBus;
6  import de.tu_freiberg.informatik.vonwenckstern.client.HistoryController;
7  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       HistoryChangedEvent;
8  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       RequestHistoryEvent;
9  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       ResourceModelChangedEvent;
10 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       ResourceModelChangedEvent.ResourceModelChangedHandler;
11 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       PlayerResourceModel;
12
13 public class InfoViewPresenter implements Activity<PlayerResourceModel>,
       ResourceModelChangedHandler {
14
15   public interface Display {
16     public void updateView(PlayerResourceModel model);
17     public Widget asWidget();
18   }
19
20   private Display display = null;
21   private PlayerResourceModel model = null;
22
23   public InfoViewPresenter(Display display, PlayerResourceModel model) {
24     this.display = display;
25     this.model = model;
26     model.addResourceModelChangedHandler(this);
27     HistoryController.getInstance().addActivityPresenter(this);
28     EventBus.fire(new RequestHistoryEvent(this));
29   }
30
31   @Override
32   public Widget getView() {
33     return display.asWidget();
34   }
35
36   @Override
37   public void onResourceChanged(ResourceModelChangedEvent event) {
38     display.updateView(model);
39     EventBus.fire(new HistoryChangedEvent(this));
40   }
41
42   private static Type<InfoViewPresenter> TYPE = new Type<InfoViewPresenter
         >("InfoViewPresenter");
```

```
43    @Override
44    public Activity.Type<?> getActivityKey() {
45      return TYPE;
46    }
47
48    @Override
49    public PlayerResourceModel getActualHistory() {
50      return model.clone();
51    }
52
53    @Override
54    public void setActualHistory(PlayerResourceModel state) {
55      model.update(state); // never reassign the model by doing this.model =
            state;
56      // because then you changed the reference, and if anywhere else this
            model was earlier referenced
57      // the view is not synchrony anymore
58      display.updateView(model);
59    }
60
61  }
```

Listing A71: InfoViewPresenter.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.presenter;
2
3  import java.util.ArrayList;
4
5  import com.google.gwt.event.dom.client.ClickEvent;
6  import com.google.gwt.event.dom.client.ClickHandler;
7  import com.google.gwt.user.client.Window;
8  import com.google.gwt.user.client.ui.Button;
9  import com.google.gwt.user.client.ui.DialogBox;
10 import com.google.gwt.user.client.ui.Grid;
11 import com.google.gwt.user.client.ui.IntegerBox;
12 import com.google.gwt.user.client.ui.PushButton;
13 import com.google.gwt.user.client.ui.Widget;
14
15 import de.tu_freiberg.informatik.vonwenckstern.client.EventBus;
16 import de.tu_freiberg.informatik.vonwenckstern.client.HistoryController;
17 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       ChildStartsWorkingEvent;
18 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       ChildStartsWorkingEvent.ChildStartsWorkingHandler;
19 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       HistoryChangedEvent;
20 import de.tu_freiberg.informatik.vonwenckstern.client.event.NextRoundEvent
       ;
21 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       PlayerFieldDoneEvent;
22 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       RequestHistoryEvent;
23 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       SaveHistoryToURLEvent;
24 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       ShowingDialogEvent;
25 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       ShowingDialogEvent.ShowingDialogHandler;
26 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       AcquisitionCardModel;
```

```
27 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       BigAcquisitions;
28 import de.tu_freiberg.informatik.vonwenckstern.client.model.Child;
29 import de.tu_freiberg.informatik.vonwenckstern.client.model.FieldCard;
30 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasAcquisitionCardModel;
31 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasBaseFieldModel;
32 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       PlayerFieldModel;
33 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       PlayerFieldModel.AcquisitationType;
34 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       PlayerFieldModel.PastureInfo;
35 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       PlayerFieldModel.PastureInfo.Animal;
36 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       PlayerResourceModel;
37 import de.tu_freiberg.informatik.vonwenckstern.client.model.Resource;
38 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       SmallFieldModel;
39 import de.tu_freiberg.informatik.vonwenckstern.client.view.HasPosition;
40
41
42 public class PlayerFieldPresenter implements Activity<PlayerFieldModel>,
       ClickHandler, ChildStartsWorkingHandler, ShowingDialogHandler {
43
44   public interface Display {
45     public void setInformation(String text);
46     public void setInformationVisible(boolean visible);
47     public void setFeedingFamilyBtnVisible(boolean visible);
48     public void setEnclosureBtnVisible(boolean visible);
49     public void update(PlayerFieldModel model);
50     public void registerHandlers(ClickHandler p);
51     public Widget asWidget();
52   }
53
54   public static enum State {
55     NONE, BUILD_HOUSE, BUILD_STABLE, PLOW_FIELD, BUILD_FENCE, GET_SHEEP,
         BACK_BREAD, SEED_GRAIN, SEED_VEGETABLE,
56     FAMILY_ADDITION, RESTAURATE, BIG_ACQUISITATION, GET_BOAR, GET_COW,
         FAMILIY_ADDITION_NO_HOUSE, KILL_ANIMALS_FIREPLACE,
57     KILL_ANIMALS_COOKERY
58   }
59
60   private Display display = null;
61   private PlayerFieldModel model = null;
62   private ArrayList<State> states = new ArrayList<State>();
63   private State state = State.NONE;
64   PlayerResourceModel resourceModel = null;
65   /** true if the harvest season already started and the user can do
         special events on the big acquisitation cards */
66   private boolean isHarvestSeason = false;
67   /** true if the user has to pick a big acquisitation and so the harvest
         season cannot start now*/
68   private boolean isGettingBigAcquisitation = false;
69   /** true if the harvest season should start when the user finished
         getting its big acquisitation */
70   private boolean souldHarvestSeasonStart = false;
71   private boolean isShowingDialog = false;
72
```

```
73   public boolean isGettingBigAcquisitation() {
74     return isGettingBigAcquisitation;
75   }
76
77   public void setGettingBigAcquisitation(boolean isGettingBigAcquisitation
         ) {
78     this.isGettingBigAcquisitation = isGettingBigAcquisitation;
79     if(!isGettingBigAcquisitation && souldHarvestSeasonStart) {
80       souldHarvestSeasonStart = false;
81       childFinishedWorking();
82     }
83   }
84
85   public void addState(State state) {
86     states.add(state);
87   }
88
89   private void updateModel() {
90     display.update(model);
91     EventBus.fire(new HistoryChangedEvent(this));
92   }
93
94   @Override
95   public void onChildStartsWorking(ChildStartsWorkingEvent event) {
96     model.sendPersonToWork();
97     if(states.isEmpty() && state == State.NONE && !
           isGettingBigAcquisitation && !isShowingDialog)
98       childFinishedWorking();
99     else {
100      if(isGettingBigAcquisitation || isShowingDialog)
101        souldHarvestSeasonStart = true;
102      updateModel();
103    }
104  }
105
106  private void childFinishedWorking() {
107    if(!model.isPersonAvailableForWork()) {
108      model.sendAllPersonsHome();
109      int round = model.getRound();
110      if(round == 4 || round == 7 || round == 9 || round == 11 || round ==
             13 || round == 14) {
111        // it is harvest season, now
112
113        // (1) pick grains and/or vegetables
114        for(int i=0; i<15; i++) {
115          SmallFieldModel m = model.getField(i);
116          if(m.getField() == FieldCard.F_FIELD && m.getRessourceCount() >
                 0) {
117            resourceModel.addRessource(m.getRessource(), 1);
118            m.setRessourceCount(m.getRessourceCount() - 1);
119            if(m.getRessourceCount() == 0) {
120              m.setRessource(Resource.R_NONE);
121            }
122          }
123        }
124        // TODO let people kill some animals for food
125        if( (model.selectAcquisitationCards(AcquisitationType.
               HARVEST_SEASON)) &&
126            Window.confirm("Do you want to do some special events on your
                   big acquisitations?")) {
127          clearSelection();
```

```
128             display.setFeedingFamilyBtnVisible(true);
129             isHarvestSeason = true;
130             executeStates();
131           } else {
132             clearSelection();
133             feedingFamily();
134           }
135         } else {
136           announceNextRound();
137         }
138       } else {
139         updateModel();
140         EventBus.fire(new SaveHistoryToURLEvent());
141       }
142     }
143
144     private void announceNextRound() {
145       model.increaseRound();
146       EventBus.fire(new NextRoundEvent(model.getRound()));
147       // children will become persons
148       for(int i=0; i<15; i++) {
149         SmallFieldModel m = model.getField(i);
150         m.setPersonsCount(m.getPersonsCount() + m.getChildCount());
151         m.setChildCount(0);
152       }
153       updateModel();
154       EventBus.fire(new SaveHistoryToURLEvent());
155     }
156
157     private void feedingFamily() {
158       // (2) feed your family
159       int food = 0;
160       for(int i=0; i<15; i++) {
161         food += 2*model.getField(i).getPersonsCount() + model.getField(i).
                 getChildCount();
162       }
163       resourceModel.addRessource(Resource.R_FOOD, -1*food);
164
165       // (3) let the animals get babies
166       model.breedAnimals();
167
168       announceNextRound();
169     }
170
171     public void addBigAcquisitation(AcquisitionCardModel acquisition) {
172       model.addAcquisitation(new AcquisitionCardModel(acquisition.
              getAcquisition(), acquisition.isVisible(), acquisition.
              getDescription()));
173       updateModel();
174     }
175
176     public void executeStates() {
177       if(!states.isEmpty()) {
178         State s = states.get(0);
179         states.remove(0);
180         switch(s) {
181         case BUILD_HOUSE: buildHouse(); break;
182         case BUILD_STABLE: buildStable(); break;
183         case PLOW_FIELD: plowField(); break;
184         case BUILD_FENCE: buildFence(); break;
185         case GET_SHEEP: getSheep(); break;
```

```
186         case BACK_BREAD: backBread(); break;
187         case SEED_GRAIN: seedGrain(); break;
188         case SEED_VEGETABLE: seedVegetable(); break;
189         case FAMILY_ADDITION: familyAddition(); break;
190         case RESTAURATE: restaurate(); break;
191         case BIG_ACQUISITATION: break;
192         case GET_BOAR: getBoar(); break;
193         case GET_COW: getCow(); break;
194         case FAMILIY_ADDITION_NO_HOUSE: familyAdditonNoHouse(); break;
195         case KILL_ANIMALS_FIREPLACE: killAnimals(State.
                KILL_ANIMALS_FIREPLACE); break;
196         case KILL_ANIMALS_COOKERY: killAnimals(State.KILL_ANIMALS_COOKERY);
                break;
197         case NONE: break;
198         }
199       } else {
200         if(isHarvestSeason) {
201           harvestSeason();
202         } else {
203           display.setInformationVisible(false);
204           display.setEnclosureBtnVisible(false);
205           EventBus.fire(new PlayerFieldDoneEvent());
206           childFinishedWorking();
207         }
208       }
209     }
210
211     private void harvestSeason() {
212       model.selectAcquisitationCards(AcquisitationType.HARVEST_SEASON);
213       updateModel();
214     }
215
216     private void killAnimals(State s) {
217       int fields = 0;
218       for(int i=0; i<15; i++) {
219         if(model.isPasture(i) && model.containsAnimals(i)) {
220           model.getField(i).setSelectable(true);
221           fields++;
222         } else {
223           model.getField(i).setSelectable(false);
224         }
225       }
226       if(fields == 0) {
227         Window.alert("You␣have␣no␣animals␣to␣kill␣to␣make␣food!");
228         this.state = State.NONE;
229         executeStates();
230       } else {
231         state = s;
232         display.setInformation("kill␣animals");
233         updateModel();
234       }
235     }
236
237     private void familyAdditonNoHouse() {
238       if(resourceModel.getPersonsCount() < 1) {
239         Window.alert("Your␣family␣size␣is␣already␣5.␣You␣do␣not␣have␣a␣free␣
                person␣stone.\nThe␣turn␣finished␣now.");
240         executeStates();
241         return;
242       }
243       for(int i=0; i<15; i++) {
```

```
244       SmallFieldModel m = model.getField(i);
245       if(model.isHouse(i)){
246         m.setSelectable(true);
247       } else {
248         m.setSelectable(false);
249       }
250     }
251     resourceModel.setPersonsCount(resourceModel.getPersonsCount() - 1);
252     state = State.FAMILIY_ADDITION_NO_HOUSE;
253     display.setInformation("choose a room for your new baby");
254     updateModel();
255   }
256
257   private void getCow() {
258     boolean selected = model.selectAcquisitationCards(AcquisitationType.
          CHANGE);
259     int fields = 0;
260     int[] pIds = model.categorizePasture();
261     for(int i=0; i<15; i++) {
262       SmallFieldModel m = model.getField(i);
263       PastureInfo pInfo = model.getPastureInfo(i, pIds);
264       if(pInfo.maxAnimals > pInfo.animalCount && (pInfo.animal == Animal.
            COW || pInfo.animal == Animal.NONE)){
265         m.setSelectable(true);
266         fields++;
267       } else {
268         m.setSelectable(false);
269       }
270     }
271     if(fields == 0 && !selected) {
272       Window.alert("You do not have a free pasture field for your sheep\
              nThe turn finished now.");
273       executeStates();
274     }
275     state = State.GET_COW;
276     display.setInformation("select a pasture for your sheep");
277     updateModel();
278   }
279
280   private void getBoar() {
281     boolean selected = model.selectAcquisitationCards(AcquisitationType.
          CHANGE);
282     int fields = 0;
283     int[] pIds = model.categorizePasture();
284     for(int i=0; i<15; i++) {
285       SmallFieldModel m = model.getField(i);
286       PastureInfo pInfo = model.getPastureInfo(i, pIds);
287       if(pInfo.maxAnimals > pInfo.animalCount && (pInfo.animal == Animal.
            BOAR || pInfo.animal == Animal.NONE)){
288         m.setSelectable(true);
289         fields++;
290       } else {
291         m.setSelectable(false);
292       }
293     }
294     if(fields == 0 && !selected) {
295       Window.alert("You do not have a free pasture field for your sheep\
              nThe turn finished now.");
296       executeStates();
297     }
298     state = State.GET_BOAR;
```

```
299     display.setInformation("select␣a␣pasture␣for␣your␣sheep");
300     updateModel();
301   }
302
303   private void restaurate() {
304     int houses = 0;
305     for(int i=0; i<15; i++) {
306       if(model.isHouse(i)){
307         houses++;
308       }
309     }
310     if(model.getField(4).getField() == FieldCard.F_WOOD_HOUSE) {
311        if(resourceModel.getClayCount() < houses) {
312          Window.alert("You␣do␣not␣have␣enough␣clay␣to␣restaurate␣your␣
               houses!");
313          executeStates();
314          return;
315        }
316        resourceModel.addRessource(Resource.R_CLAY, houses*-1);
317     } else if(model.getField(4).getField() == FieldCard.F_CLAY_HOUSE) {
318       if(resourceModel.getStoneCount() < houses) {
319          Window.alert("You␣do␣not␣have␣enough␣stone␣to␣restaurate␣your␣
               houses!");
320          executeStates();
321          return;
322        }
323        resourceModel.addRessource(Resource.R_STONE, houses*-1);
324     }
325
326     for(int i=0; i<15; i++) {
327       SmallFieldModel m = model.getField(i);
328       if(model.isHouse(i)){
329         if(m.getField() == FieldCard.F_WOOD_HOUSE)
330           m.setField(FieldCard.F_CLAY_HOUSE);
331         else if(m.getField() == FieldCard.F_CLAY_HOUSE)
332           m.setField(FieldCard.F_STONE_HOUSE);
333       }
334     }
335     updateModel();
336     executeStates();
337   }
338
339   private void familyAddition() {
340     if(resourceModel.getPersonsCount() < 1) {
341       Window.alert("Your␣family␣size␣is␣already␣5.␣You␣do␣not␣have␣a␣free␣
             person␣stone.\nThe␣turn␣finished␣now.");
342       executeStates();
343       return;
344     }
345     int fields = 0;
346     for(int i=0; i<15; i++) {
347       SmallFieldModel m = model.getField(i);
348       if(model.isHouse(i) && m.getChild() == Child.C_NONE){
349         m.setSelectable(true);
350         fields++;
351       } else {
352         m.setSelectable(false);
353       }
354     }
355     if(fields == 0) {
```

```
356       Window.alert("You␣do␣not␣have␣a␣free␣room␣for␣your␣baby.\nThe␣turn␣
              finished␣now.");
357       executeStates();
358       return;
359     }
360     resourceModel.setPersonsCount(resourceModel.getPersonsCount() - 1);
361     state = State.FAMILY_ADDITION;
362     display.setInformation("choose␣a␣place␣for␣your␣new␣baby");
363     updateModel();
364   }
365
366   private void seedVegetable() {
367     if(resourceModel.getVegetableCount() < 1) {
368       Window.alert("You␣have␣no␣vegetable␣to␣seed␣your␣fields");
369       executeStates();
370     } else {
371       resourceModel.addRessource(Resource.R_VEGETABLE, -1);
372       int fields = 0;
373       for(int i=0; i<15; i++) {
374         SmallFieldModel m = model.getField(i);
375         if(m.getField() == FieldCard.F_FIELD && m.getRessourceCount() ==
              0){
376           m.setSelectable(true);
377           fields++;
378         } else {
379           m.setSelectable(false);
380         }
381       }
382       if(fields == 0) {
383         Window.alert("You␣have␣no␣free␣field␣to␣seed␣vegetable\nThe␣turn␣
              finished␣now.");
384         executeStates();
385       } else {
386         state = State.SEED_VEGETABLE;
387         display.setInformation("seed␣your␣vegetables");
388         updateModel();
389       }
390     }
391   }
392
393   private void seedGrain() {
394     if(resourceModel.getGrainCount() < 1) {
395       Window.alert("You␣have␣no␣grain␣to␣seed␣your␣fields");
396       executeStates();
397     } else {
398       resourceModel.addRessource(Resource.R_GRAIN, -1);
399       int fields = 0;
400       for(int i=0; i<15; i++) {
401         SmallFieldModel m = model.getField(i);
402         if(m.getField() == FieldCard.F_FIELD && m.getRessourceCount() ==
              0){
403           m.setSelectable(true);
404           fields++;
405         } else {
406           m.setSelectable(false);
407         }
408       }
409       if(fields == 0) {
410         Window.alert("You␣have␣no␣free␣field␣to␣seed␣grain\nThe␣turn␣
              finished␣now.");
411         executeStates();
```

```
412         } else {
413           state = State.SEED_GRAIN;
414           display.setInformation("seed your grain");
415           updateModel();
416         }
417       }
418     }
419
420     private void backBread() {
421       boolean selected = model.selectAcquisitationCards(AcquisitationType.
              BACK_BREAD);
422       if(!selected) {
423         Window.alert("You have no big acquisitation for backing bread");
424         executeStates();
425       } else {
426         if(resourceModel.getGrainCount() < 1) {
427           Window.alert("You have no grain for backing bread");
428           executeStates();
429         } else {
430           state = State.BACK_BREAD;
431           display.setInformation("What card do you want to use for backing
                  bread?");
432           updateModel();
433         }
434       }
435
436     }
437
438     public PlayerFieldPresenter(Display display, PlayerFieldModel model,
          PlayerResourceModel resourceModel) {
439       this.display = display;
440       this.model = model;
441       this.resourceModel = resourceModel;
442       display.registerHandlers(this);
443       EventBus.getEventBus().addChildStartsWorkingHandler(this);
444       EventBus.getEventBus().addShwoingDialogHandler(this);
445       HistoryController.getInstance().addActivityPresenter(this);
446       EventBus.fire(new RequestHistoryEvent(this));
447     }
448
449     @Override
450     public Widget getView() {
451       return display.asWidget();
452     }
453
454     private void getSheep() {
455       boolean selected = model.selectAcquisitationCards(AcquisitationType.
              CHANGE);
456       int fields = 0;
457       int[] pIds = model.categorizePasture();
458       for(int i=0; i<15; i++) {
459         SmallFieldModel m = model.getField(i);
460         PastureInfo pInfo = model.getPastureInfo(i, pIds);
461         if(pInfo.maxAnimals > pInfo.animalCount && (pInfo.animal == Animal.
                SHEEP || pInfo.animal == Animal.NONE)){
462           m.setSelectable(true);
463           fields++;
464         } else {
465           m.setSelectable(false);
466         }
467       }
```

```
468     if(fields == 0 && !selected) {
469       Window.alert("You␣do␣not␣have␣a␣free␣pasture␣field␣for␣your␣sheep\
              nThe␣turn␣finished␣now.");
470       executeStates();
471     }
472     state = State.GET_SHEEP;
473     display.setInformation("select␣a␣pasture␣for␣your␣sheep");
474     updateModel();
475   }
476
477   private void buildFence() {
478     for(int i=0; i<15; i++) {
479       SmallFieldModel m = model.getField(i);
480       if(model.isPasture(i) || model.isHouse(i) || model.isField(i)){
481         m.setSelectable(false);
482       } else {
483         m.setSelectable(true);
484       }
485     }
486     state = State.BUILD_FENCE;
487     display.setInformation("build␣your␣fence␣enclosure");
488     display.setEnclosureBtnVisible(true);
489     updateModel();
490   }
491
492
493
494   private void plowField() {
495     for(int i=0; i<15; i++) {
496       SmallFieldModel m = model.getField(i);
497       if(m.getField() == FieldCard.F_NONE && !model.isPasture(i) && (model
              .countFields() == 0 ||
498           ((i/5 == (i-1)/5) && model.isField(i-1) || (i/5 == (i+1)/5) &&
                  model.isField(i+1) || // fields are next to each other in
                  the same row
499             model.isField(i-5) || model.isField(i+5))    )){ // fields
                    are next to each other in the same column
500         m.setSelectable(true);
501       } else {
502         m.setSelectable(false);
503       }
504     }
505     state = State.PLOW_FIELD;
506     display.setInformation("plow␣your␣field");
507     updateModel();
508   }
509
510   private void buildStable() {
511     if(resourceModel.getWoodCount() < 2 || resourceModel.getStableCount()
          < 1) {
512       if(resourceModel.getWoodCount() < 2)
513         Window.alert("You␣do␣not␣have␣enough␣wood␣to␣build␣a␣stable!");
514       else
515         Window.alert("You␣have␣only␣4␣stables.\nYou␣have␣no␣free␣stable␣
              stone␣anymore.\nYour␣turn␣is␣over␣now.");
516       executeStates();
517     } else {
518       resourceModel.addRessource(Resource.R_WOOD, -2);
519       resourceModel.setStableCount(resourceModel.getStableCount() - 1);
520       for(int i=0; i<15; i++) {
521         SmallFieldModel m = model.getField(i);
```

```
522         if(m.getField() == FieldCard.F_NONE){
523           m.setSelectable(true);
524         } else {
525           m.setSelectable(false);
526         }
527       }
528       state = State.BUILD_STABLE;
529       display.setInformation("build your stable");
530       updateModel();
531     }
532   }
533
534   private void buildHouse() {
535     if(resourceModel.getReedCount() < 2) {
536       Window.alert("You have not enough reed to build any house!");
537       executeStates();
538     } else if(model.getField(4).getField() == FieldCard.F_CLAY_HOUSE &&
          resourceModel.getClayCount() < 5) {
539       Window.alert("You have not enough clay to build a clay house!");
540       executeStates();
541     } else if(model.getField(4).getField() == FieldCard.F_STONE_HOUSE &&
          resourceModel.getStoneCount() < 5) {
542       Window.alert("You have not enough stone to build a stone house!");
543       executeStates();
544     } else if(model.getField(4).getField() == FieldCard.F_WOOD_HOUSE &&
          resourceModel.getWoodCount() < 5) {
545       Window.alert("You have not enough wood to build a wood house!");
546       executeStates();
547     } else { // the player has enough resources
548       resourceModel.addRessource(Resource.R_REED, -2);
549       if(model.getField(4).getField() == FieldCard.F_WOOD_HOUSE) {
550         resourceModel.addRessource(Resource.R_WOOD, -5);
551       } else if(model.getField(4).getField() == FieldCard.F_CLAY_HOUSE) {
552         resourceModel.addRessource(Resource.R_CLAY, -5);
553       } else if(model.getField(4).getField() == FieldCard.F_STONE_HOUSE) {
554         resourceModel.addRessource(Resource.R_STONE, -5);
555       }
556       for(int i=0; i<15; i++) {
557         SmallFieldModel m = model.getField(i);
558         if(m.getField() == FieldCard.F_NONE && !model.isPasture(i) &&
559             ((i/5 == (i-1)/5) && model.isHouse(i-1) || (i/5 == (i+1)/5) &&
                  model.isHouse(i+1) || // houses are next to each other in
                      the same row
560                 model.isHouse(i-5) || model.isHouse(i+5)) ){ // houses are
                      next to each other in the same column
561           m.setSelectable(true);
562         } else {
563           m.setSelectable(false);
564         }
565       }
566       state = State.BUILD_HOUSE;
567       display.setInformation("build your house");
568       updateModel();
569     }
570   }
571
572   private void clearSelection() {
573     for(int i=0; i<15; i++) {
574       SmallFieldModel m = model.getField(i);
575       m.setSelectable(false);
576     }
```

```
577     for(int i=0; i<10; i++) {
578       AcquisitionCardModel m = model.getAcquisitation(i);
579       if(m != null)
580         m.setSelectable(false);
581     }
582   }
583
584   @Override
585   public void onClick(ClickEvent event) {
586     if(event.getSource() instanceof PushButton) {
587       if(((PushButton)event.getSource()).getText().equals("new enclosure")
              ) {
588         model.nextFence();
589       } else {
590         isHarvestSeason = false;
591         display.setFeedingFamilyBtnVisible(false);
592         model.setAllowBasketMaker(true);
593         model.setAllowJoinery(true);
594         model.setAllowPottery(true);
595         clearSelection();
596         feedingFamily();
597       }
598     } else if(event.getSource() instanceof HasBaseFieldModel && event.
          getSource() instanceof HasPosition) {
599       SmallFieldModel m = (SmallFieldModel) ((HasBaseFieldModel)event.
              getSource()).getModel();
600       int pos = ((HasPosition)event.getSource()).getPosition();
601       if(state == State.BUILD_HOUSE) {
602         if(m.isSelectable()) {
603           m.setField(model.getField(4).getField());
604           clearSelection();
605           state = State.NONE;
606           updateModel();
607         } else {
608           Window.alert("You can only build a house on the highlighted
                  fields.");
609         }
610         executeStates();
611       } else if(state == State.BUILD_STABLE) {
612         if(m.isSelectable()) {
613           m.setField(FieldCard.F_STABLE);
614           clearSelection();
615           state = State.NONE;
616           updateModel();
617         } else {
618           Window.alert("You can only build a house on the highlighted
                  fields.");
619         }
620         executeStates();
621       } else if(state == State.PLOW_FIELD) {
622         if(m.isSelectable()) {
623           m.setField(FieldCard.F_FIELD);
624           clearSelection();
625           state = State.NONE;
626           updateModel();
627         } else {
628           Window.alert("You can only build a house on the highlighted
                  fields.");
629         }
630         executeStates();
631       } else if(state == State.BUILD_FENCE) {
```

```
632         if(m.isSelectable()) {
633           int wood = 0;
634           wood += (pos /5 == (pos-1) / 5 && model.hasRightFence(pos-1)) ?
                  -1 : pos /5 == (pos-1) / 5 && pos > 0 && model.getField(pos
                  -1).getRightFence() > 0 ? 0 : 1;
635           wood += (pos /5 == (pos+1) / 5 && model.hasLeftFence(pos+1)) ?
                  -1 :  pos /5 == (pos+1) / 5 && pos < 14 && model.getField(
                  pos+1).getLeftFence() > 0 ? 0 : 1;
636           wood += model.hasTopFence(pos+5) ? -1 : pos<10 && model.getField
                  (pos+5).getTopFence() > 0 ? 0: 1;
637           wood += model.hasBottomFence(pos-5) ? -1 : pos > 4 && model.
                  getField(pos-5).getBottomFence() > 0 ? 0 : 1;
638           if(resourceModel.getWoodCount() < wood || resourceModel.
                  getFenceCount() < wood) {
639             if(resourceModel.getWoodCount() < wood)
640               Window.alert("You␣have␣not␣enough␣wood␣to␣build␣this␣fence")
                      ;
641             else
642               Window.alert("You␣have␣not␣enough␣fence␣parts␣to␣build␣this␣
                      fence");
643             clearSelection();
644             state = State.NONE;
645           } else {
646             resourceModel.addRessource(Resource.R_WOOD, -1*wood);
647             resourceModel.setFenceCount(resourceModel.getFenceCount() -
                    wood);
648             model.setLeftFence(pos, pos / 5 != (pos-1) / 5 || !model.
                    hasRightFence(pos-1));
649             if(pos /5 == (pos-1) / 5)
650               model.setRightFence(pos-1, false);
651             model.setRightFence(pos, pos / 5 != (pos+1) / 5 || !model.
                    hasLeftFence(pos+1));
652             if(pos /5 == (pos+1) / 5)
653               model.setLeftFence(pos+1, false);
654             model.setBottomFence(pos, !model.hasTopFence(pos+5));
655             model.setTopFence(pos+5, false);
656             model.setTopFence(pos, !model.hasBottomFence(pos-5));
657             model.setBottomFence(pos-5, false);
658             clearSelection();
659             state = State.NONE;
660             updateModel();
661           }
662         } else {
663           Window.alert("You␣can␣only␣build␣a␣house␣on␣the␣highlighted␣
                  fields.");
664         }
665         executeStates();
666       } else if(state == State.GET_SHEEP) {
667         if(m.isSelectable()) {
668           m.setRessource(Resource.R_SHEEP);
669           m.setRessourceCount(m.getRessourceCount() + 1);
670           clearSelection();
671           state = State.NONE;
672           updateModel();
673         } else {
674           Window.alert("You␣can␣only␣build␣a␣house␣on␣the␣highlighted␣
                  fields.");
675         }
676         executeStates();
677       } else if(state == State.SEED_GRAIN) {
678         if(m.isSelectable()) {
```

```
679             m.setRessource(Resource.R_GRAIN);
680             m.setRessourceCount(3);
681              clearSelection();
682              state = State.NONE;
683              updateModel();
684           } else {
685             Window.alert("You can only build a house on the highlighted
                  fields.");
686           }
687           executeStates();
688         } else if(state == State.SEED_VEGETABLE) {
689           if(m.isSelectable()) {
690             m.setRessource(Resource.R_VEGETABLE);
691             m.setRessourceCount(2);
692              clearSelection();
693              state = State.NONE;
694              updateModel();
695           } else {
696             Window.alert("You can only build a house on the highlighted
                  fields.");
697           }
698           executeStates();
699         } else if(state == State.FAMILY_ADDITION) {
700           if(m.isSelectable()) {
701             model.setChild(pos);
702             m.setChildCount(1);
703              clearSelection();
704              state = State.NONE;
705              updateModel();
706           } else {
707             Window.alert("You can only build a house on the highlighted
                  fields.");
708           }
709           executeStates();
710         } else if(state == State.GET_BOAR) {
711           if(m.isSelectable()) {
712             m.setRessource(Resource.R_BOAR);
713             m.setRessourceCount(m.getRessourceCount() + 1);
714              clearSelection();
715              state = State.NONE;
716              updateModel();
717           } else {
718             Window.alert("You can only build a house on the highlighted
                  fields.");
719           }
720           executeStates();
721         } else if(state == State.GET_COW) {
722           if(m.isSelectable()) {
723             m.setRessource(Resource.R_COW);
724             m.setRessourceCount(m.getRessourceCount() + 1);
725              clearSelection();
726              state = State.NONE;
727              updateModel();
728           } else {
729             Window.alert("You can only build a house on the highlighted
                  fields.");
730           }
731           executeStates();
732         } else if(state == State.FAMILIY_ADDITION_NO_HOUSE) {
733           if(m.isSelectable()) {
734             model.setChild(pos);
```

```
735           m.setChildCount(1);
736           clearSelection();
737           state = State.NONE;
738           updateModel();
739         } else {
740           Window.alert("You can only build a house on the highlighted
                fields.");
741         }
742         executeStates();
743       } else if(state == State.KILL_ANIMALS_FIREPLACE) {
744         if(m.isSelectable()) {
745           resourceModel.addRessource(Resource.R_FOOD, m.getRessource() ==
                Resource.R_COW ? 3 : 2);
746           m.setRessourceCount(m.getRessourceCount() - 1);
747           clearSelection();
748           state = State.NONE;
749           updateModel();
750         } else {
751           Window.alert("You can only kill animals on the highlighted
                fields.");
752         }
753         executeStates();
754       } else if(state == State.KILL_ANIMALS_COOKERY) {
755         if(m.isSelectable()) {
756           resourceModel.addRessource(Resource.R_FOOD, m.getRessource() ==
                Resource.R_COW ? 4 :
757             m.getRessource() == Resource.R_BOAR ? 3 : 2);
758           m.setRessourceCount(m.getRessourceCount() - 1);
759           clearSelection();
760           state = State.NONE;
761           updateModel();
762         } else {
763           Window.alert("You can only kill animals on the highlighted
                fields.");
764         }
765         executeStates();
766       }
767     } else if(event.getSource() instanceof HasAcquisitionCardModel) {
768       HasAcquisitionCardModel ar = (HasAcquisitionCardModel)event.
            getSource();
769       if(ar.getModel().isSelectable() || state == State.NONE) {
770         BigAcquisitions ba = ar.getModel().getAcquisition();
771         if( (ba == BigAcquisitions.BA_FIRE_PLACE) || (ba ==
              BigAcquisitions.BA_FIRE_PLACE2) ) {
772           switch(state) {
773           case GET_SHEEP:
774           case GET_BOAR:
775             resourceModel.addRessource(Resource.R_FOOD, 2); break;
776           case GET_COW:
777             resourceModel.addRessource(Resource.R_FOOD, 3); break;
778           case BACK_BREAD:
779             String input = Window.prompt("How many grain to want to change
                  into food? (1 grain -> 2 food markers)", "1");
780             try {
781               int nb = Integer.parseInt(input);
782               if(resourceModel.getGrainCount() < nb)
783                 nb = resourceModel.getGrainCount();
784               resourceModel.addRessource(Resource.R_GRAIN, -1*nb);
785               resourceModel.addRessource(Resource.R_FOOD, 2*nb);
786             } catch(Exception e) {
```

```
787                 Window.alert("You should have enter a number, now your turn
                        is over.");
788               }
789               break;
790             case NONE:
791               showDialogFirePlaceCookery(State.KILL_ANIMALS_FIREPLACE);
792               break;
793             default: break;
794             }
795           } else if( (ba == BigAcquisitions.BA_COOKERY) || (ba ==
                  BigAcquisitions.BA_COOKERY2) ) {
796             switch(state) {
797             case GET_SHEEP:
798               resourceModel.addRessource(Resource.R_FOOD, 2); break;
799             case GET_BOAR:
800               resourceModel.addRessource(Resource.R_FOOD, 3); break;
801             case GET_COW:
802               resourceModel.addRessource(Resource.R_FOOD, 4); break;
803             case BACK_BREAD:
804               String input = Window.prompt("How many grain to want to change
                      into food? (1 grain -> 3 food markers)", "1");
805               try {
806                 int nb = Integer.parseInt(input);
807                 if(nb < 0) nb = 0;
808                 if(resourceModel.getGrainCount() < nb)
809                   nb = resourceModel.getGrainCount();
810                 resourceModel.addRessource(Resource.R_GRAIN, -1*nb);
811                 resourceModel.addRessource(Resource.R_FOOD, 3*nb);
812               } catch(Exception e) {
813                 Window.alert("You should have enter a number, now your turn
                        is over.");
814               }
815               break;
816             case NONE:
817               showDialogFirePlaceCookery(State.KILL_ANIMALS_COOKERY);
818               break;
819             default: break;
820             }
821           } else if((ba == BigAcquisitions.BA_CLAY_OVEN) && state == State.
                  BACK_BREAD && resourceModel.getGrainCount() > 0) {
822             resourceModel.addRessource(Resource.R_GRAIN, -1);
823             resourceModel.addRessource(Resource.R_FOOD, 5);
824           } else if((ba == BigAcquisitions.BA_STONE_OVEN) && state == State.
                  BACK_BREAD && resourceModel.getGrainCount() > 0) {
825             String input = Window.prompt("Do you want to change ONE or TWO
                      grain into food? (1 grain -> 4 food markers)", "1");
826             try {
827               int nb = Integer.parseInt(input);
828               if(nb < 0) nb = 0;
829               if(nb > 2) nb = 2;
830               if(resourceModel.getGrainCount() < nb)
831                 nb = resourceModel.getGrainCount();
832               resourceModel.addRessource(Resource.R_GRAIN, -1*nb);
833               resourceModel.addRessource(Resource.R_FOOD, 4*nb);
834             } catch(Exception e) {
835               Window.alert("You should have enter a number, now your turn is
                      over.");
836             }
837           } else if((ba == BigAcquisitions.BA_JOINERY) && isHarvestSeason) {
838             model.setAllowJoinery(false);
839             if(resourceModel.getWoodCount()< 1) {
```

```
840             Window.alert("You have not enough wood!");
841           } else {
842             resourceModel.addRessource(Resource.R_WOOD, -1);
843             resourceModel.addRessource(Resource.R_FOOD, 2);
844           }
845         } else if((ba == BigAcquisitions.BA_POTTERY) && isHarvestSeason) {
846           model.setAllowPottery(false);
847           if(resourceModel.getClayCount()< 1) {
848             Window.alert("You have not enough wood!");
849           } else {
850             resourceModel.addRessource(Resource.R_CLAY, -1);
851             resourceModel.addRessource(Resource.R_FOOD, 2);
852           }
853         } else if((ba == BigAcquisitions.BA_BASKET_MAKER) &&
                isHarvestSeason) {
854           model.setAllowBasketMaker(false);
855           if(resourceModel.getClayCount()< 1) {
856             Window.alert("You have not enough wood!");
857           } else {
858             resourceModel.addRessource(Resource.R_REED, -1);
859             resourceModel.addRessource(Resource.R_FOOD, 3);
860           }
861         }
862       } else {
863         Window.alert("You can only select highlighted cards.");
864       }
865       clearSelection();
866       state = State.NONE;
867       updateModel();
868       executeStates();
869     }
870   }
871   private void showDialogFirePlaceCookery(final State s) {
872     final DialogBox dlg = new DialogBox(false, true);
873     Grid grid = new Grid(4, 2);
874     grid.setHTML(0, 0, "How many resources do you want to change to food:"
          );
875     grid.setHTML(1, 0, "How many <b>animals</b> do you want to kill?");
876     final IntegerBox animals = new IntegerBox();
877     animals.setValue(0);
878     grid.setWidget(1, 1, animals);
879     grid.setHTML(2, 0, "How many <b>vegetables</b> do you want to eat?");
880     final IntegerBox vegetables = new IntegerBox();
881     vegetables.setValue(0);
882     grid.setWidget(2, 1, vegetables);
883     Button ok = new Button("OK");
884     ok.addClickHandler(new ClickHandler() {
885       @Override
886       public void onClick(ClickEvent event) {
887         dlg.hide();
888         EventBus.fire(new ShowingDialogEvent(false));
889         // TODO make other things gray
890         int veg = vegetables.getValue();
891         if(resourceModel.getVegetableCount() < veg)
892           veg = resourceModel.getVegetableCount();
893         resourceModel.addRessource(Resource.R_VEGETABLE, -1*veg);
894         resourceModel.addRessource(Resource.R_FOOD, s == State.
              KILL_ANIMALS_FIREPLACE ? 2*veg : 3*veg);
895         for(int i=0; i<animals.getValue(); i++) {
896           states.add(s);
897         }
```

```
898             executeStates();
899           }
900         });
901         grid.setWidget(3, 1, ok);
902         dlg.setWidget(grid);
903         EventBus.fire(new ShowingDialogEvent(true));
904         dlg.show();
905       }
906
907       @Override
908       public void onShowingDialog(ShowingDialogEvent event) {
909         isShowingDialog = event.isShowingDialog();
910         if(!isShowingDialog && souldHarvestSeasonStart) {
911           souldHarvestSeasonStart = false;
912           childFinishedWorking();
913         }
914       }
915
916       private static Type<PlayerFieldPresenter> TYPE = new Type<
              PlayerFieldPresenter>("PlayerFieldPresenter");
917       @Override
918       public Activity.Type<?> getActivityKey() {
919         return TYPE;
920       }
921
922       @Override
923       public PlayerFieldModel getActualHistory() {
924         return model.clone();
925       }
926
927       @Override
928       public void setActualHistory(PlayerFieldModel state) {
929         model.update(state);
930         display.update(model);
931       }
932
933 }
```

Listing A72: PlayerFieldPresenter.java file

```
1 package de.tu_freiberg.informatik.vonwenckstern.client.presenter;
2
3 import com.google.gwt.user.client.ui.Widget;
4
5 public interface Presenter {
6   public Widget getView();
7 }
```

Listing A73: Presenter.java file

```
1 package de.tu_freiberg.informatik.vonwenckstern.client.presenter;
2
3 import java.util.Iterator;
4
5 import com.google.gwt.event.dom.client.ClickEvent;
6 import com.google.gwt.event.dom.client.ClickHandler;
7 import com.google.gwt.uibinder.client.UiConstructor;
8 import com.google.gwt.user.client.Window;
9 import com.google.gwt.user.client.ui.HasWidgets;
10 import com.google.gwt.user.client.ui.Widget;
11
```

```
12 import de.tu_freiberg.informatik.vonwenckstern.client.EventBus;
13 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       AddResourceEvent;
14 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       AddResourceEvent.RessourceItem;
15 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       ChildStartsWorkingEvent;
16 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       HistoryChangedEvent;
17 import de.tu_freiberg.informatik.vonwenckstern.client.event.NextRoundEvent
       ;
18 import de.tu_freiberg.informatik.vonwenckstern.client.event.NextRoundEvent
       .NextRoundHandler;
19 import de.tu_freiberg.informatik.vonwenckstern.client.model.BaseFieldModel
       ;
20 import de.tu_freiberg.informatik.vonwenckstern.client.model.BigFieldModel;
21 import de.tu_freiberg.informatik.vonwenckstern.client.model.Child;
22 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasBaseFieldModel;
23 import de.tu_freiberg.informatik.vonwenckstern.client.model.Player;
24 import de.tu_freiberg.informatik.vonwenckstern.client.model.Resource;
25 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
26 public class ResourcePresenter implements Presenter, ClickHandler,
       NextRoundHandler {
27   public interface Display {
28     public void registerHandlers(ClickHandler p);
29     public Widget asWidget();
30     public void showOneMoreBigCard(int round);
31     public HasWidgets getPanel();
32   }
33   protected Display display;
34   protected Player player = Player.BLUE;
35
36   @UiConstructor
37   public ResourcePresenter(Display display, Player player) {
38     this.display = display;
39     this.player = player;
40     EventBus.getEventBus().addNextRoundHandler(this);
41     display.registerHandlers(this);
42   }
43
44   @Override
45   public Widget getView() {
46     return display.asWidget();
47   }
48
49   @Override
50   public void onClick(ClickEvent event) {
51     Object s = event.getSource();
52     if(s instanceof Widget) {
53       Renderer renderer = getRenderer((Widget) s);
54       if(renderer instanceof HasBaseFieldModel) {
55         BaseFieldModel model = ((HasBaseFieldModel) renderer).getModel();
56         if(model.getChild() != Child.C_NONE) {
57           Window.alert("This␣place␣is␣already␣occupied.␣You␣cannot␣set␣
                your␣child␣here.");
58           return;
59         }
60
61         if(processAction(model)) {
62
```

```
63            Child child = Child.C_NONE;
64            switch(player) {
65            case BLUE: child = Child.C_BLUE; break;
66            case GREEN: child = Child.C_GREEN; break;
67            case RED: child = Child.C_RED; break;
68            case ROSA: child = Child.C_ROSA; break;
69            case NONE: child = Child.C_NONE; break;
70            }
71            model.setChild(child);
72            renderer.render();
73            if(this instanceof Activity) {
74              EventBus.fire(new HistoryChangedEvent((Activity<?>)this));
75            }
76            EventBus.fire(new ChildStartsWorkingEvent());
77          }
78        } else {
79          processAction((Widget) s);
80        }
81      }
82    }
83
84    /**
85     * overload this method to handle special cards for which no model
          exists
86     * @param w the widget which got clicked
87     */
88    public void processAction(Widget w) {
89
90    }
91
92    /**
93     * overload this method to handle special cards for which a model exists
94     * @param model the model of the card which got clicked <br>and do not
          forget to call super.processAction()
95     * @return true if the operation is allowed and the child should get set
          , otherwise false
96     */
97    public boolean processAction(BaseFieldModel model) {
98      if(model.getRessourceCount() > 0 && model.getRessource() != Resource.
          R_NONE) {
99        RessourceItem[] items = new RessourceItem[] { new RessourceItem(
              model.getRessource(), model.getRessourceCount()) };
100       EventBus.getEventBus().fireEvent(new AddResourceEvent(items));
101       model.setRessourceCount(0);
102     }
103     return true;
104   }
105
106   public Renderer getRenderer(Widget w) {
107     do {
108       if(w instanceof Renderer) {
109         return (Renderer) w;
110       }
111     } while( (w = w.getParent()) != null);
112     return null;
113   }
114
115   @Override
116   public void onNextRound(NextRoundEvent event) {
117     removeChildrenAndAddRessources((Widget) display.getPanel());
118   }
```

```
119
120   private void removeChildrenAndAddRessources(Widget w) {
121     if(w instanceof Renderer) {
122       Renderer r = (Renderer)w;
123       if(r instanceof HasBaseFieldModel) {
124         BaseFieldModel model = ((HasBaseFieldModel) r).getModel();
125         model.setChild(Child.C_NONE);
126         if(model instanceof BigFieldModel && w.isVisible()) {
127           model.setRessourceCount(model.getRessourceCount() + ((
                BigFieldModel)model).getRessourceRoundAddition());
128         }
129         r.render();
130       }
131     } else if(w instanceof HasWidgets) {
132       Iterator<Widget> it = ((HasWidgets) w).iterator();
133       while(it.hasNext()) {
134         removeChildrenAndAddRessources(it.next());
135       }
136     }
137   }
138
139   public void showOneMoreBigCard(int round) {
140     display.showOneMoreBigCard(round);
141   }
142 }
```

Listing A74: ResourcePresenter.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.presenter;
2
3  import de.tu_freiberg.informatik.vonwenckstern.client.EventBus;
4  import de.tu_freiberg.informatik.vonwenckstern.client.HistoryController;
5  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       BuildFenceEvent;
6  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       EnableBigAcquisitionEvent;
7  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       FamilyAdditionEvent;
8  import de.tu_freiberg.informatik.vonwenckstern.client.event.GetSheepEvent;
9  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       HistoryChangedEvent;
10 import de.tu_freiberg.informatik.vonwenckstern.client.event.NextRoundEvent
       ;
11 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       RequestHistoryEvent;
12 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       RestaurateEvent;
13 import de.tu_freiberg.informatik.vonwenckstern.client.event.SeedEvent;
14 import de.tu_freiberg.informatik.vonwenckstern.client.model.BackgroundCard
       ;
15 import de.tu_freiberg.informatik.vonwenckstern.client.model.BaseFieldModel
       ;
16 import de.tu_freiberg.informatik.vonwenckstern.client.model.BigFieldModel;
17 import de.tu_freiberg.informatik.vonwenckstern.client.model.Player;
18 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       Rounds1To7Model;
19
20 public class Rounds1To7Presenter extends ResourcePresenter implements
       Activity<Rounds1To7Model>{
21
22   public interface Display extends ResourcePresenter.Display {
```

```
23      public void update(Rounds1To7Model model);
24    }
25
26    private Rounds1To7Model model = null;
27
28    public Rounds1To7Presenter(Display display, Player player,
          Rounds1To7Model model) {
29      super(display, player);
30      this.model = model;
31      display.update(model);
32      HistoryController.getInstance().addActivityPresenter(this);
33      EventBus.fire(new RequestHistoryEvent(this));
34    }
35
36    public boolean processAction(BaseFieldModel model) {
37      if(model instanceof BigFieldModel && ((BigFieldModel) model).getBgCard
            () == BackgroundCard.SHEEP) {
38        if(model.getRessourceCount() > 0) {
39          EventBus.fire(new GetSheepEvent(model.getRessourceCount()));
40          model.setRessourceCount(0);
41        }
42      } else {
43        super.processAction(model); // process other resources
44      }
45      if(model instanceof BigFieldModel) {
46        switch(((BigFieldModel) model).getBgCard()) {
47        case ACQUISITION: EventBus.fire(new EnableBigAcquisitionEvent());
              break;
48        case FENCE: EventBus.fire(new BuildFenceEvent()); break;
49        case SEEDING_BACKING: EventBus.fire(new SeedEvent()); break;
50        case FAMILIY_ADDITION2: EventBus.fire(new FamilyAdditionEvent());
              break;
51        case RESTAURATION: EventBus.fire(new RestaurateEvent()); break;
52        default: break;
53        }
54      }
55      return true;
56    }
57
58    @Override
59    public void onNextRound(NextRoundEvent event) {
60      super.onNextRound(event);
61      if(event.getRound() < 8)
62        showOneMoreBigCard(event.getRound());
63      EventBus.fire(new HistoryChangedEvent(this));
64    }
65
66    private static Type<Rounds1To7Presenter> TYPE = new Type<
          Rounds1To7Presenter>("Rounds1To7Presenter");
67    @Override
68    public Activity.Type<?> getActivityKey() {
69      return TYPE;
70    }
71
72    @Override
73    public Rounds1To7Model getActualHistory() {
74      return model.clone();
75    }
76
77    @Override
78    public void setActualHistory(Rounds1To7Model state) {
```

```
79      model.update(state);
80      ((Display)display).update(model);
81    }
82
83  }
```

Listing A75: Rounds1To7Presenter.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.presenter;
2
3  import de.tu_freiberg.informatik.vonwenckstern.client.EventBus;
4  import de.tu_freiberg.informatik.vonwenckstern.client.HistoryController;
5  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       AddResourceEvent;
6  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       AddResourceEvent.RessourceItem;
7  import de.tu_freiberg.informatik.vonwenckstern.client.event.
       FamilyAdditionWithoutHouseEvent;
8  import de.tu_freiberg.informatik.vonwenckstern.client.event.GetBoarEvent;
9  import de.tu_freiberg.informatik.vonwenckstern.client.event.GetCowEvent;
10 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       HistoryChangedEvent;
11 import de.tu_freiberg.informatik.vonwenckstern.client.event.NextRoundEvent
       ;
12 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       PlowFieldSeedEvent;
13 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       RequestHistoryEvent;
14 import de.tu_freiberg.informatik.vonwenckstern.client.event.
       RestaurateAndFenceEvent;
15 import de.tu_freiberg.informatik.vonwenckstern.client.model.BackgroundCard
       ;
16 import de.tu_freiberg.informatik.vonwenckstern.client.model.BaseFieldModel
       ;
17 import de.tu_freiberg.informatik.vonwenckstern.client.model.BigFieldModel;
18 import de.tu_freiberg.informatik.vonwenckstern.client.model.Player;
19 import de.tu_freiberg.informatik.vonwenckstern.client.model.Resource;
20 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       Rounds8To14Model;
21
22 public class Rounds8To14Presenter extends ResourcePresenter implements
       Activity<Rounds8To14Model>{
23
24   public interface Display extends ResourcePresenter.Display {
25     public void update(Rounds8To14Model model);
26   }
27
28   private Rounds8To14Model model = null;
29
30   public Rounds8To14Presenter(Display display, Player player,
         Rounds8To14Model model) {
31     super(display, player);
32     this.model = model;
33     display.update(model);
34     HistoryController.getInstance().addActivityPresenter(this);
35     EventBus.fire(new RequestHistoryEvent(this));
36   }
37
38   public boolean processAction(BaseFieldModel model) {
39     if(model instanceof BigFieldModel && ((BigFieldModel) model).getBgCard
           () == BackgroundCard.BOAR) {
```

```
40        if(model.getRessourceCount() > 0) {
41          EventBus.fire(new GetBoarEvent(model.getRessourceCount()));
42          model.setRessourceCount(0);
43        }
44      } else if(model instanceof BigFieldModel && ((BigFieldModel) model).
          getBgCard() == BackgroundCard.COW) {
45        if(model.getRessourceCount() > 0) {
46          EventBus.fire(new GetCowEvent(model.getRessourceCount()));
47          model.setRessourceCount(0);
48        }
49      } else {
50        super.processAction(model); // process other resources
51      }
52      if(model instanceof BigFieldModel) {
53        switch(((BigFieldModel) model).getBgCard()) {
54        case VEGETABLE: {
55          RessourceItem[] items = new RessourceItem[] { new RessourceItem(
              Resource.R_VEGETABLE, 1)};
56          EventBus.fire(new AddResourceEvent(items));
57          break; }
58        case PLOWING_SOWING: EventBus.fire(new PlowFieldSeedEvent()); break;
59        case FAMILY_ADDITION5: EventBus.fire(new
            FamilyAdditionWithoutHouseEvent()); break;
60        case RESTAURATION_FENCE: EventBus.fire(new RestaurateAndFenceEvent()
            ); break;
61        default: break;
62        }
63      }
64      return true;
65    }
66
67    @Override
68    public void onNextRound(NextRoundEvent event) {
69      super.onNextRound(event);
70      if(event.getRound() > 7)
71        showOneMoreBigCard(event.getRound());
72      EventBus.fire(new HistoryChangedEvent(this));
73    }
74
75    private static Type<Rounds8To14Presenter> TYPE = new Type<
        Rounds8To14Presenter>("Rounds8To14Presenter");
76    @Override
77    public Activity.Type<?> getActivityKey() {
78      return TYPE;
79    }
80
81    @Override
82    public Rounds8To14Model getActualHistory() {
83      return model.clone();
84    }
85
86    @Override
87    public void setActualHistory(Rounds8To14Model state) {
88      model.update(state);
89      ((Display)display).update(model);
90    }
91  }
```

Listing A76: Rounds8To14Presenter.java file

de.tu_freiberg.informatik.vonwenckstern.client.resources package

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.resources;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.resources.client.ClientBundle;
5  import com.google.gwt.resources.client.ImageResource;
6
7  public interface Images extends ClientBundle {
8
9    public static class Util {
10     private static Images images = GWT.create(Images.class); // do not set
           it to null, and load if we need it, because we need it for sure
11     public static Images getInstance() {
12       return images;
13     }
14   }
15
16   @Source("clear.cache.gif")
17   ImageResource clear();
18
19   @Source("DSC06140.JPG")
20   ImageResource pottery();
21
22   @Source("DSC06141.JPG")
23   ImageResource basketMaker();
24
25   @Source("DSC06142.JPG")
26   ImageResource cookery2();
27
28   @Source("DSC06143.JPG")
29   ImageResource firePlace();
30
31   @Source("DSC06144.JPG")
32   ImageResource fountain();
33
34   @Source("DSC06145.JPG")
35   ImageResource cookery();
36
37   @Source("DSC06146.JPG")
38   ImageResource clayOven();
39
40   @Source("DSC06147.JPG")
41   ImageResource stoneOven();
42
43   @Source("DSC06148.JPG")
44   ImageResource joinery();
45
46   @Source("DSC06149.JPG")
47   ImageResource firePlace2();
48
49   @Source("DSC06150.JPG")
50   ImageResource sheep();
51
52   @Source("DSC06151.JPG")
53   ImageResource acquisition();
54
55   @Source("DSC06152.JPG")
56   ImageResource fence();
```

```
57
58    @Source("DSC06153.JPG")
59    ImageResource seedingBacking();
60
61    @Source("DSC06154.JPG")
62    ImageResource familyAddition();
63
64    @Source("DSC06155.JPG")
65    ImageResource stone();
66
67    @Source("DSC06156.JPG")
68    ImageResource restauration();
69
70    @Source("DSC06157.JPG")
71    ImageResource boar();
72
73    @Source("DSC06158.JPG")
74    ImageResource vegetable();
75
76    @Source("DSC06159.JPG")
77    ImageResource stone2();
78
79    @Source("DSC06160.JPG")
80    ImageResource cow();
81
82    @Source("DSC06161.JPG")
83    ImageResource plowingField();
84
85    @Source("DSC06162.JPG")
86    ImageResource familyAddition2();
87
88    @Source("DSC06163.JPG")
89    ImageResource restauration2();
90
91    @Source("DSC06164.JPG")
92    ImageResource wood();
93
94    @Source("DSC06165.JPG")
95    ImageResource beggarCard();
96
97    @Source("DSC06166.JPG")
98    ImageResource clayTwo();
99
100   @Source("DSC06167.JPG")
101   ImageResource woodTwo();
102
103   @Source("DSC06170.JPG")
104   ImageResource reedStoneFood();
105
106   @Source("DSC06171.JPG")
107   ImageResource cabaret();
108
109 //    @Source("DSC06172.JPG")
110   @Source("bild3.jpg")
111   ImageResource playerField();
112
113   //@Source("DSC06174.JPG")
114   @Source("bild5.jpg")
115   ImageResource rounds8To14();
116
117   //@Source("DSC06175.JPG")
```

```
118     @Source("bild4.jpg")
119     ImageResource cardField();
120
121     //@Source("DSC06176.JPG")
122     @Source("bild6.jpg")
123     ImageResource rounds1To7();
124
125     //@Source("DSC06177.JPG")
126     @Source("bild2.jpg")
127     ImageResource bigAcquisitionsField();
128
129     @Source("DSC06178.JPG")
130     ImageResource gamePhases();
131
132     @Source("acker.JPG")
133     ImageResource fieldMarker();
134
135     @Source("cover.JPG")
136     ImageResource cover();
137
138     @Source("gemuese.png")
139     ImageResource vegetableStone();
140
141     @Source("getreide.png")
142     ImageResource grainStone();
143
144     @Source("holz.png")
145     ImageResource woodStone();
146
147     @Source("holzhaus.jpg")
148     ImageResource woodHouse();
149
150     @Source("steinhaus.jpg")
151     ImageResource stoneHouse();
152
153     @Source("kind_rosa.JPG")
154     ImageResource childRosa();
155
156     @Source("kind_blau.png")
157     ImageResource childBlue();
158
159     @Source("kind_gruen.JPG")
160     ImageResource childGreen();
161
162     @Source("kind_rot.JPG")
163     ImageResource childRed();
164
165     @Source("haus_rosa.JPG")
166     ImageResource houseRosa();
167
168     @Source("haus_blau.JPG")
169     ImageResource houseBlue();
170
171     @Source("haus_gruen.JPG")
172     ImageResource houseGreen();
173
174     @Source("haus_rot.JPG")
175     ImageResource houseRed();
176
177     @Source("lehm.png")
178     ImageResource clayStone();
```

```
179
180     @Source("lehmhaus.JPG")
181     ImageResource clayHouse();
182
183     @Source("nw.png")
184     ImageResource foodMarker();
185
186     @Source("rind.JPG")
187     ImageResource cowMarker();
188
189     @Source("schaaf.JPG")
190     ImageResource sheepMarker();
191
192     @Source("schwein.JPG")
193     ImageResource boarMarker();
194
195     @Source("stein.png")
196     ImageResource stoneStone();
197
198     @Source("schilf.png")
199     ImageResource reedStone();
200
201     @Source("boarIcon.JPG")
202     ImageResource boarIcon();
203
204     @Source("sheepIcon.JPG")
205     ImageResource sheepIcon();
206
207     @Source("cowIcon.JPG")
208     ImageResource cowIcon();
209
210     @Source("vegetableIcon.png")
211     ImageResource vegetableIcon();
212
213     @Source("grainIcon.JPG")
214     ImageResource grainIcon();
215
216     @Source("clayIcon.png")
217     ImageResource clayIcon();
218
219     @Source("stoneIcon.png")
220     ImageResource stoneIcon();
221
222     @Source("woodIcon.png")
223     ImageResource woodIcon();
224
225     @Source("reedIcon.png")
226     ImageResource reedIcon();
227 }
```

Listing A77: Images.java file

de.tu_freiberg.informatik.vonwenckstern.client.view package

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view;
2
3  import de.tu_freiberg.informatik.vonwenckstern.client.AppController;
4  import de.tu_freiberg.informatik.vonwenckstern.client.AppController.
       Display;
5  import de.tu_freiberg.informatik.vonwenckstern.client.AppView;
6  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       BigAcquisitionsPresenter;
7  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       CardFieldPresenter;
8  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       InfoViewPresenter;
9  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       PlayerFieldPresenter;
10 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       Rounds1To7Presenter;
11 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       Rounds8To14Presenter;
12 import de.tu_freiberg.informatik.vonwenckstern.client.view.desktop.
       BigAcquisitionsFieldView;
13 import de.tu_freiberg.informatik.vonwenckstern.client.view.desktop.
       CardFieldView;
14 import de.tu_freiberg.informatik.vonwenckstern.client.view.desktop.
       InfoView;
15 import de.tu_freiberg.informatik.vonwenckstern.client.view.desktop.
       PlayerFieldView;
16 import de.tu_freiberg.informatik.vonwenckstern.client.view.desktop.
       Rounds1To7View;
17 import de.tu_freiberg.informatik.vonwenckstern.client.view.desktop.
       Rounds8To14View;
18
19 public class DesktopViewFactory implements ViewFactory {
20   private static final AppController.Display appView = new AppView();
21   private static final BigAcquisitionsPresenter.Display acquisitionView =
         new BigAcquisitionsFieldView();
22   private static final CardFieldPresenter.Display cardFieldView = new
         CardFieldView();
23   private static final InfoViewPresenter.Display infoView = new InfoView()
         ;
24   private static final PlayerFieldPresenter.Display playerFieldView = new
         PlayerFieldView();
25   private static final Rounds1To7Presenter.Display rounds1To7View = new
         Rounds1To7View();
26   private static final Rounds8To14Presenter.Display rounds8To14View = new
         Rounds8To14View();
27
28   @Override
29   public BigAcquisitionsPresenter.Display getAcquisitionsView() {
30     return acquisitionView;
31   }
32
33   @Override
34   public CardFieldPresenter.Display getCardFieldView() {
35     return cardFieldView;
36   }
37
38   @Override
```

```
39    public InfoViewPresenter.Display getInfoView() {
40      return infoView;
41    }
42
43    @Override
44    public PlayerFieldPresenter.Display getPlayerFieldView() {
45      return playerFieldView;
46    }
47
48    @Override
49    public Rounds1To7Presenter.Display getRounds1To7View() {
50      return rounds1To7View;
51    }
52
53    @Override
54    public Rounds8To14Presenter.Display getRounds8To14View() {
55      return rounds8To14View;
56    }
57
58    @Override
59    public Display getAppView() {
60      return appView;
61    }
62
63  }
```

Listing A78: DesktopViewFactory.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view;
2
3  public interface HasPosition {
4    public int getPosition();
5  }
```

Listing A79: HasPosition.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view;
2
3  import de.tu_freiberg.informatik.vonwenckstern.client.AppController;
4  import de.tu_freiberg.informatik.vonwenckstern.client.AppController.
       Display;
5  import de.tu_freiberg.informatik.vonwenckstern.client.AppViewMobile;
6  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       BigAcquisitionsPresenter;
7  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       CardFieldPresenter;
8  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       InfoViewPresenter;
9  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       PlayerFieldPresenter;
10 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       Rounds1To7Presenter;
11 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       Rounds8To14Presenter;
12 import de.tu_freiberg.informatik.vonwenckstern.client.view.mobile.
       BigAcquisitionsFieldView;
13 import de.tu_freiberg.informatik.vonwenckstern.client.view.mobile.
       CardFieldView;
14 import de.tu_freiberg.informatik.vonwenckstern.client.view.mobile.InfoView
       ;
15 import de.tu_freiberg.informatik.vonwenckstern.client.view.mobile.
       PlayerFieldView;
```

```
16 import de.tu_freiberg.informatik.vonwenckstern.client.view.mobile.
       Rounds1To7View;
17 import de.tu_freiberg.informatik.vonwenckstern.client.view.mobile.
       Rounds8To14View;
18
19 public class MobileViewFactory implements ViewFactory {
20   private static final AppController.Display appView = new AppViewMobile()
         ;
21   private static final BigAcquisitionsPresenter.Display acquisitionView =
         new BigAcquisitionsFieldView();
22   private static final CardFieldPresenter.Display cardFieldView = new
         CardFieldView();
23   private static final InfoViewPresenter.Display infoView = new InfoView()
         ;
24   private static final PlayerFieldPresenter.Display playerFieldView = new
         PlayerFieldView();
25   private static final Rounds1To7Presenter.Display rounds1To7View = new
         Rounds1To7View();
26   private static final Rounds8To14Presenter.Display rounds8To14View = new
         Rounds8To14View();
27
28   @Override
29   public BigAcquisitionsPresenter.Display getAcquisitionsView() {
30     return acquisitionView;
31   }
32
33   @Override
34   public CardFieldPresenter.Display getCardFieldView() {
35     return cardFieldView;
36   }
37
38   @Override
39   public InfoViewPresenter.Display getInfoView() {
40     return infoView;
41   }
42
43   @Override
44   public PlayerFieldPresenter.Display getPlayerFieldView() {
45     return playerFieldView;
46   }
47
48   @Override
49   public Rounds1To7Presenter.Display getRounds1To7View() {
50     return rounds1To7View;
51   }
52
53   @Override
54   public Rounds8To14Presenter.Display getRounds8To14View() {
55     return rounds8To14View;
56   }
57
58   @Override
59   public Display getAppView() {
60     return appView;
61   }
62
63 }
```

Listing A80: MobileViewFactory.java file

```
1 package de.tu_freiberg.informatik.vonwenckstern.client.view;
```

```
2
3 public interface Renderer {
4   /**
5    * renders the entire model
6    */
7   public void render();
8 }
```

Listing A81: Renderer.java file

```
1 package de.tu_freiberg.informatik.vonwenckstern.client.view;
2
3 import com.google.gwt.event.dom.client.ClickEvent;
4 import com.google.gwt.event.dom.client.ClickHandler;
5 import com.google.gwt.event.dom.client.MouseOutEvent;
6 import com.google.gwt.event.dom.client.MouseOutHandler;
7 import com.google.gwt.event.dom.client.MouseOverEvent;
8 import com.google.gwt.event.dom.client.MouseOverHandler;
9 import com.google.gwt.user.client.DOM;
10 import com.google.gwt.user.client.ui.PopupPanel;
11 import com.google.gwt.user.client.ui.Widget;
12
13 public class Tooltip extends PopupPanel implements MouseOverHandler,
     MouseOutHandler, ClickHandler {
14   private Widget w;
15   public Tooltip(Widget w, Widget tooltip) {
16     super(false, false);
17     this.w = w;
18     w.addDomHandler(this, MouseOverEvent.getType());
19     w.addDomHandler(this, MouseOutEvent.getType());
20     w.addDomHandler(this, ClickEvent.getType());
21     this.add(tooltip);
22     this.show();
23     DOM.setStyleAttribute(this.getElement(), "zIndex", "100");
24     this.setVisible(false);
25   }
26
27   public Tooltip() {
28     super(false, false);
29   }
30
31   @Override
32   public void onMouseOut(MouseOutEvent event) {
33     this.setVisible(false);
34   }
35   @Override
36   public void onMouseOver(MouseOverEvent event) {
37     this.setPopupPosition(w.getAbsoluteLeft() + event.getRelativeX(w.
           getElement()) + 10, w.getAbsoluteTop() + event.getRelativeY(w.
           getElement()) + 10);
38     this.setVisible(true);
39   }
40
41   @Override
42   public void onClick(ClickEvent event) {
43     this.setVisible(false);
44   }
45 }
```

Listing A82: Tooltip.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view;
2
3  import com.google.gwt.core.shared.GWT;
4
5  import de.tu_freiberg.informatik.vonwenckstern.client.AppController;
6  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       BigAcquisitionsPresenter;
7  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       CardFieldPresenter;
8  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       InfoViewPresenter;
9  import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       PlayerFieldPresenter;
10 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       Rounds1To7Presenter;
11 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       Rounds8To14Presenter;
12
13 public interface ViewFactory {
14   public AppController.Display getAppView();
15   public BigAcquisitionsPresenter.Display getAcquisitionsView();
16   public CardFieldPresenter.Display getCardFieldView();
17   public InfoViewPresenter.Display getInfoView();
18   public PlayerFieldPresenter.Display getPlayerFieldView();
19   public Rounds1To7Presenter.Display getRounds1To7View();
20   public Rounds8To14Presenter.Display getRounds8To14View();
21
22   public static class Util {
23     private static ViewFactory viewfactory = GWT.create(ViewFactory.class)
           ;
24     public static ViewFactory getViewFactory() {
25       return viewfactory;
26     }
27   }
28 }
```

Listing A83: ViewFactory.java file

de.tu_freiberg.informatik.vonwenckstern.client.view.desktop package

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import com.google.gwt.user.client.DOM;
4  import com.google.gwt.user.client.ui.AbsolutePanel;
5  import com.google.gwt.user.client.ui.HTML;
6  import com.google.gwt.user.client.ui.Label;
7
8  import de.tu_freiberg.informatik.vonwenckstern.client.model.
       AcquisitionCardModel;
9  import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasAcquisitionCardModel;
10 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
11 import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
12
13 public class AcquisitionCardRenderer extends AbsolutePanel implements
       Renderer, HasAcquisitionCardModel {
14   private AcquisitionCardModel model = null;
15   private int position;
16   private boolean tooltipAdded = false;
17   public int getPosition() {
18     return position;
19   }
20
21   public void setPosition(int position) {
22     this.position = position;
23   }
24
25   public AcquisitionCardRenderer() {
26     this.setPixelSize(65, 110);
27   }
28
29   public AcquisitionCardModel getModel() {
30     return model;
31   }
32
33   public void setModel(AcquisitionCardModel model) {
34     this.model = model;
35     if(!tooltipAdded && model != null) {
36       tooltipAdded = true;
37       new Tooltip(this, new HTML(model.getDescription()));
38     }
39     if(model != null) {
40       render();
41     } else {
42       this.setVisible(false);
43     }
44   }
45
46   public void render() {
47     this.clear();
48     this.add(new TooltipImageAcquisitionRenderer(model),0,0);
49     if(model.isSelectable()) {
50       Label l = new Label();
51       l.setPixelSize(65, 110);
52       DOM.setStyleAttribute(l.getElement(), "backgroundColor", "rgba
             (255,255,255,0.4)");
53       DOM.setStyleAttribute(l.getElement(), "cursor", "pointer");
```

```
54          this.add(1,0,0);
55        }
56        this.setVisible(model.isVisible());
57      }
58  }
```

Listing A84: AcquisitionCardRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.event.dom.client.ClickHandler;
5  import com.google.gwt.uibinder.client.UiBinder;
6  import com.google.gwt.user.client.ui.AbsolutePanel;
7  import com.google.gwt.user.client.ui.Composite;
8  import com.google.gwt.user.client.ui.Widget;
9
10 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       BigAcquisitationsModel;
11 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       BigAcquisitions;
12 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       BigAcquisitionsPresenter;
13
14 public class BigAcquisitionsFieldView extends Composite implements
       BigAcquisitionsPresenter.Display {
15
16   private static final Binder binder = GWT.create(Binder.class);
17
18   interface Binder extends UiBinder<AbsolutePanel,
         BigAcquisitionsFieldView> {
19   }
20
21   private AbsolutePanel panel = null;
22
23   public BigAcquisitionsFieldView() {
24     panel = binder.createAndBindUi(this);
25     initWidget(panel);
26   }
27
28   @Override
29   public void registerHandlers(ClickHandler p) {
30     for(int i=0; i<panel.getWidgetCount(); i++) {
31       Widget w = panel.getWidget(i);
32       if(w instanceof TooltipImageAcquisitionRenderer) {
33         ((TooltipImageAcquisitionRenderer) w).addClickHandler(p);
34       }
35     }
36   }
37
38   @Override
39   public void hideAcquisition(BigAcquisitions aquisition) {
40     for(int i=0; i<panel.getWidgetCount(); i++) {
41       Widget w = panel.getWidget(i);
42       if(w instanceof TooltipImageAcquisitionRenderer &&
43           (((TooltipImageAcquisitionRenderer) w).getModel().getAcquisition
                ) == aquisition) ) {
44         w.setVisible(false);
45         ((TooltipImageAcquisitionRenderer) w).getModel().setVisible(false)
               ;
46         break;
```

```
47        }
48      }
49    }
50
51    @Override
52    public void update(BigAcquisitationsModel model) {
53      for(int i=0; i<10; i++) {
54        ((TooltipImageAcquisitionRenderer)panel.getWidget(i+1)).setModel(
              model.getModel(i));
55      }
56    }
57
58  }
```

Listing A85: BigAcquisitionsFieldView.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       desktop'>
5    <ui:style>
6      .handcursor {
7        cursor: pointer;
8        }
9    </ui:style>
10   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BigAcquisitions.*" />
11   <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
         .resources.Images'/>
12
13     <g:AbsolutePanel width="1000px" height="1000px">
14       <g:at left="0" top="0">
15         <g:HTML>
16           <img src="{im.bigAcquisitionsField.getSafeUri}" width="400px" />
17         </g:HTML>
18       </g:at>
19
20       <g:at left="5" top="25">
21         <a:TooltipImageAcquisitionRenderer bigAcquisiton="{BA_FIRE_PLACE}"
               >
22           <a:tooltip>
23             <g:HTML>
24             <b>fire place</b> <img src="{im.clayIcon.getSafeUri}" width="
                   20px" height="20px"/><img src="{im.clayIcon.getSafeUri}"
                   width="20px"/><br/>
25             Vegetable <img src="{im.vegetableIcon.getSafeUri}" width="20px
                   "/> &rarr; <img src="{im.foodMarker.getSafeUri}" width="20
                   px"/><img src="{im.foodMarker.getSafeUri}" width="20px"/><
                   br/>
26             Sheep <img src="{im.sheepIcon.getSafeUri}" width="20px"/> &
                   rarr; <img src="{im.foodMarker.getSafeUri}" width="20px"
                   /><img src="{im.foodMarker.getSafeUri}" width="20px"/><br
                   />
27             Boar <img src="{im.boarIcon.getSafeUri}" width="20px"/> &rarr;
                    <img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                   src="{im.foodMarker.getSafeUri}" width="20px"/><br/>
28             Cow <img src="{im.cowIcon.getSafeUri}" width="20px"/> &rarr; <
                   img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                   src="{im.foodMarker.getSafeUri}" width="20px"/><img src="{
```

```
                im.foodMarker.getSafeUri}" width="20px"/><br/>
29              <br/><br/>for the action <b>backing bread</b>:<br/>
30              Grain <img src="{im.grainIcon.getSafeUri}" width="20px"/> &
                    rarr; <img src="{im.foodMarker.getSafeUri}" width="20px"
                    /><img src="{im.foodMarker.getSafeUri}" width="20px"/>
31              </g:HTML>
32            </a:tooltip>
33          </a:TooltipImageAcquisitionRenderer>
34        </g:at>
35        <g:at left="85" top="25">
36          <a:TooltipImageAcquisitionRenderer bigAcquisiton="{BA_FIRE_PLACE2}
                ">
37            <a:tooltip>
38              <g:HTML>
39              <b>fire place</b> <img src="{im.clayIcon.getSafeUri}" width="
                    20px"/><img src="{im.clayIcon.getSafeUri}" width="20px"/><
                    img src="{im.clayIcon.getSafeUri}" width="20px"/><br/>
40              Vegetable <img src="{im.vegetableIcon.getSafeUri}" width="20px
                    "/> &rarr; <img src="{im.foodMarker.getSafeUri}" width="20
                    px"/><img src="{im.foodMarker.getSafeUri}" width="20px"/><
                    br/>
41              Sheep <img src="{im.sheepIcon.getSafeUri}" width="20px"/> &
                    rarr; <img src="{im.foodMarker.getSafeUri}" width="20px"
                    /><img src="{im.foodMarker.getSafeUri}" width="20px"/><br
                    />
42              Boar <img src="{im.boarIcon.getSafeUri}" width="20px"/> &rarr;
                     <img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                    src="{im.foodMarker.getSafeUri}" width="20px"/><br/>
43              Cow <img src="{im.cowIcon.getSafeUri}" width="20px"/> &rarr; <
                    img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                    src="{im.foodMarker.getSafeUri}" width="20px"/><img src="{
                    im.foodMarker.getSafeUri}" width="20px"/><br/>
44              <br/><br/>for the action <b>backing bread</b>:<br/>
45              Grain <img src="{im.grainIcon.getSafeUri}" width="20px"/> &
                    rarr; <img src="{im.foodMarker.getSafeUri}" width="20px"
                    /><img src="{im.foodMarker.getSafeUri}" width="20px"/>
46              </g:HTML>
47            </a:tooltip>
48          </a:TooltipImageAcquisitionRenderer>
49        </g:at>
50        <g:at left="168" top="25">
51          <a:TooltipImageAcquisitionRenderer bigAcquisiton="{BA_COOKERY}">
52            <a:tooltip>
53              <g:HTML>
54              <b>cookery</b> <img src="{im.clayIcon.getSafeUri}" width="20
                    px"/><img src="{im.clayIcon.getSafeUri}" width="20px"/><
                    img src="{im.clayIcon.getSafeUri}" width="20px"/><img src=
                    "{im.clayIcon.getSafeUri}" width="20px"/><br/>
55              Vegetable <img src="{im.vegetableIcon.getSafeUri}" width="20px
                    "/> &rarr; <img src="{im.foodMarker.getSafeUri}" width="20
                    px"/><img src="{im.foodMarker.getSafeUri}" width="20px"/><
                    img src="{im.foodMarker.getSafeUri}" width="20px"/><br/>
56              Sheep <img src="{im.sheepIcon.getSafeUri}" width="20px"/> &
                    rarr; <img src="{im.foodMarker.getSafeUri}" width="20px"
                    /><img src="{im.foodMarker.getSafeUri}" width="20px"/><br
                    />
57              Boar <img src="{im.boarIcon.getSafeUri}" width="20px"/> &rarr;
                     <img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                    src="{im.foodMarker.getSafeUri}" width="20px"/><img src="{
                    im.foodMarker.getSafeUri}" width="20px"/><br/>
```

```
58            Cow <img src="{im.cowIcon.getSafeUri}" width="20px"/> &rarr; <
                  img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                  src="{im.foodMarker.getSafeUri}" width="20px"/><img src="{
                  im.foodMarker.getSafeUri}" width="20px"/><img src="{im.
                  foodMarker.getSafeUri}" width="20px"/><br/>
59            <br/><br/>for the action <b>backing bread</b>:<br/>
60            Grain <img src="{im.grainIcon.getSafeUri}" width="20px"/> &
                  rarr; <img src="{im.foodMarker.getSafeUri}" width="20px"
                  /><img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                   src="{im.foodMarker.getSafeUri}" width="20px"/>
61            </g:HTML>
62          </a:tooltip>
63        </a:TooltipImageAcquisitionRenderer>
64      </g:at>
65      <g:at left="250" top="25">
66        <a:TooltipImageAcquisitionRenderer bigAcquisiton="{BA_COOKERY2}">
67          <a:tooltip>
68            <g:HTML>
69            <b>cookery</b> <img src="{im.clayIcon.getSafeUri}" width="20
                  px"/><img src="{im.clayIcon.getSafeUri}" width="20px"/><
                  img src="{im.clayIcon.getSafeUri}" width="20px"/><img src=
                  "{im.clayIcon.getSafeUri}" width="20px"/><img src="{im.
                  clayIcon.getSafeUri}" width="20px"/><br/>
70            Vegetable <img src="{im.vegetableIcon.getSafeUri}" width="20px
                  "/> &rarr; <img src="{im.foodMarker.getSafeUri}" width="20
                  px"/><img src="{im.foodMarker.getSafeUri}" width="20px"/><
                  img src="{im.foodMarker.getSafeUri}" width="20px"/><br/>
71            Sheep <img src="{im.sheepIcon.getSafeUri}" width="20px"/> &
                  rarr; <img src="{im.foodMarker.getSafeUri}" width="20px"
                  /><img src="{im.foodMarker.getSafeUri}" width="20px"/><br
                  />
72            Boar <img src="{im.boarIcon.getSafeUri}" width="20px"/> &rarr;
                   <img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                  src="{im.foodMarker.getSafeUri}" width="20px"/><img src="{
                  im.foodMarker.getSafeUri}" width="20px"/><br/>
73            Cow <img src="{im.cowIcon.getSafeUri}" width="20px"/> &rarr; <
                  img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                  src="{im.foodMarker.getSafeUri}" width="20px"/><img src="{
                  im.foodMarker.getSafeUri}" width="20px"/><img src="{im.
                  foodMarker.getSafeUri}" width="20px"/><br/>
74            <br/><br/>for the action <b>backing bread</b>:<br/>
75            Grain <img src="{im.grainIcon.getSafeUri}" width="20px"/> &
                  rarr; <img src="{im.foodMarker.getSafeUri}" width="20px"
                  /><img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                   src="{im.foodMarker.getSafeUri}" width="20px"/>
76            </g:HTML>
77          </a:tooltip>
78        </a:TooltipImageAcquisitionRenderer>
79      </g:at>
80      <g:at left="330" top="25">
81        <a:TooltipImageAcquisitionRenderer bigAcquisiton="{BA_FOUNTAIN}">
82          <a:tooltip>
83            <g:HTML>
84            <b>fountain</b> <img src="{im.woodIcon.getSafeUri}" width="20
                  px"/><img src="{im.stoneIcon.getSafeUri}" width="20px"/><
                  img src="{im.stoneIcon.getSafeUri}" width="20px"/><img src
                  ="{im.stoneIcon.getSafeUri}" width="20px"/><br/>
85            You will get 5 food markers.
86            </g:HTML>
87          </a:tooltip>
88        </a:TooltipImageAcquisitionRenderer>
```

```
89        </g:at>
90
91        <g:at left="5" top="160">
92          <a:TooltipImageAcquisitionRenderer bigAcquisiton="{BA_CLAY_OVEN}">
93            <a:tooltip>
94              <g:HTML>
95              <b>clay oven</b> <img src="{im.stoneIcon.getSafeUri}" width="
                    20px"/><img src="{im.clayIcon.getSafeUri}" width="20px"/><
                    img src="{im.clayIcon.getSafeUri}" width="20px"/><img src=
                    "{im.clayIcon.getSafeUri}" width="20px"/><br/>
96              for the action <b>backing bread</b>:<br/>
97              <b>1x</b> Grain <img src="{im.grainIcon.getSafeUri}" width="20
                    px"/> &rarr; <img src="{im.foodMarker.getSafeUri}" width="
                    20px"/><img src="{im.foodMarker.getSafeUri}" width="20px"
                    /><img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                     src="{im.foodMarker.getSafeUri}" width="20px"/><img src="
                    {im.foodMarker.getSafeUri}" width="20px"/>
98              </g:HTML>
99            </a:tooltip>
100         </a:TooltipImageAcquisitionRenderer>
101       </g:at>
102       <g:at left="85" top="160">
103         <a:TooltipImageAcquisitionRenderer bigAcquisiton="{BA_STONE_OVEN}"
              >
104           <a:tooltip>
105             <g:HTML>
106             <b>stone oven</b> <img src="{im.clayIcon.getSafeUri}" width="
                    20px"/><img src="{im.stoneIcon.getSafeUri}" width="20px"
                    /><img src="{im.stoneIcon.getSafeUri}" width="20px"/><img
                    src="{im.stoneIcon.getSafeUri}" width="20px"/><br/>
107             for the action <b>backing bread</b>:<br/>
108             <b>2x</b> Grain <img src="{im.grainIcon.getSafeUri}" width="20
                    px"/> &rarr; <img src="{im.foodMarker.getSafeUri}" width="
                    20px"/><img src="{im.foodMarker.getSafeUri}" width="20px"
                    /><img src="{im.foodMarker.getSafeUri}" width="20px"/><img
                     src="{im.foodMarker.getSafeUri}" width="20px"/>
109             </g:HTML>
110           </a:tooltip>
111         </a:TooltipImageAcquisitionRenderer>
112       </g:at>
113       <g:at left="168" top="160">
114         <a:TooltipImageAcquisitionRenderer bigAcquisiton="{BA_JOINERY}">
115           <a:tooltip>
116             <g:HTML>
117             <b>joinery</b> <img src="{im.woodIcon.getSafeUri}" width="20
                    px"/><img src="{im.woodIcon.getSafeUri}" width="20px"/><
                    img src="{im.stoneIcon.getSafeUri}" width="20px"/><img src
                    ="{im.stoneIcon.getSafeUri}" width="20px"/><br/>
118             in every <b>harvest season</b>:<br/>
119             <b>1x</b> Wood <img src="{im.woodIcon.getSafeUri}" width="20px
                    "/> &rarr; <img src="{im.foodMarker.getSafeUri}" width="20
                    px"/><img src="{im.foodMarker.getSafeUri}" width="20px"/>
120             </g:HTML>
121           </a:tooltip>
122         </a:TooltipImageAcquisitionRenderer>
123       </g:at>
124       <g:at left="250" top="160">
125         <a:TooltipImageAcquisitionRenderer bigAcquisiton="{BA_POTTERY}">
126           <a:tooltip>
127             <g:HTML>
```

```
128             <b>pottery</b> <img src="{im.clayIcon.getSafeUri}" width="20
                    px"/><img src="{im.clayIcon.getSafeUri}" width="20px"/><
                    img src="{im.stoneIcon.getSafeUri}" width="20px"/><img src
                    ="{im.stoneIcon.getSafeUri}" width="20px"/><br/>
129             in every <b>harvest season</b>:<br/>
130             <b>1x</b> Clay <img src="{im.clayIcon.getSafeUri}" width="20px
                    "/> &rarr; <img src="{im.foodMarker.getSafeUri}" width="20
                    px"/><img src="{im.foodMarker.getSafeUri}" width="20px"/>
131             </g:HTML>
132           </a:tooltip>
133         </a:TooltipImageAcquisitionRenderer>
134       </g:at>
135       <g:at left="330" top="160">
136         <a:TooltipImageAcquisitionRenderer bigAcquisiton="{BA_BASKET_MAKER
                }">
137           <a:tooltip>
138             <g:HTML>
139             <b>basket maker</b> <img src="{im.reedIcon.getSafeUri}" width
                    ="20px"/><img src="{im.reedIcon.getSafeUri}" width="20px"
                    /><img src="{im.stoneIcon.getSafeUri}" width="20px"/><img
                    src="{im.stoneIcon.getSafeUri}" width="20px"/><br/>
140             in every <b>harvest season</b>:<br/>
141             <b>1x</b> Reed <img src="{im.reedIcon.getSafeUri}" width="20px
                    "/> &rarr; <img src="{im.foodMarker.getSafeUri}" width="20
                    px"/><img src="{im.foodMarker.getSafeUri}" width="20px"/><
                    img src="{im.foodMarker.getSafeUri}" width="20px"/>
142             </g:HTML>
143           </a:tooltip>
144         </a:TooltipImageAcquisitionRenderer>
145       </g:at>
146     </g:AbsolutePanel>
147 </ui:UiBinder>
```

Listing A86: BigAcquisitionsFieldView.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3
4  import com.google.gwt.resources.client.ImageResource;
5  import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
6  import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
7
8  import com.google.gwt.safehtml.shared.SafeHtmlBuilder;
9  import com.google.gwt.user.client.ui.AbsolutePanel;
10 import com.google.gwt.user.client.ui.HTML;
11 import com.google.gwt.user.client.ui.Image;
12
13 import de.tu_freiberg.informatik.vonwenckstern.client.model.BigFieldModel;
14 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasBaseFieldModel;
15 import de.tu_freiberg.informatik.vonwenckstern.client.resources.Images;
16
17 public class BigFieldRenderer extends AbsolutePanel implements
       HasBaseFieldModel, Renderer {
18   private BigFieldModel model;
19
20
21
22   public BigFieldRenderer() {
23     this.setPixelSize(100, 150);
24   }
```

```
25
26    public BigFieldModel getModel() {
27      return model;
28    }
29
30    public void setModel(BigFieldModel model) {
31      this.model = model;
32      if(model != null) {
33        if(model.getDescription() != null)
34          new Tooltip(this, new HTML(new SafeHtmlBuilder().
                appendEscapedLines(model.getDescription()).toSafeHtml()));
35        render();
36      }
37    }
38
39    public void render() {
40      this.clear();
41      Images im = Images.Util.getInstance();
42      ImageResource bgImage = null;
43      switch(model.getBgCard()) {
44      case ACQUISITION: bgImage = im.acquisition(); break;
45      case BOAR: bgImage = im.boar(); break;
46      case CABARET: bgImage = im.cabaret(); break;
47      case COW: bgImage = im.cow(); break;
48      case FAMILIY_ADDITION2: bgImage = im.familyAddition(); break;
49      case FAMILY_ADDITION5: bgImage = im.familyAddition2(); break;
50      case FENCE: bgImage = im.fence(); break;
51      case NONE: bgImage = null; break;
52      case ONE_WOOD: bgImage = im.wood(); break;
53      case PLOWING_SOWING: bgImage = im.plowingField(); break;
54      case REED_STONE_FOOD: bgImage = im.reedStoneFood(); break;
55      case RESTAURATION: bgImage = im.restauration(); break;
56      case RESTAURATION_FENCE: bgImage = im.restauration2(); break;
57      case SEEDING_BACKING: bgImage = im.seedingBacking(); break;
58      case SHEEP: bgImage = im.sheep(); break;
59      case STONE2: bgImage = im.stone(); break;
60      case STONE4: bgImage = im.stone2(); break;
61      case TWO_CLAY: bgImage = im.clayTwo(); break;
62      case TWO_WOOD: bgImage = im.woodTwo(); break;
63      case VEGETABLE: bgImage = im.vegetable(); break;
64      }
65      if(bgImage != null) {
66        Image image = new Image();
67        image.setHeight("150px");
68        image.setUrl(bgImage.getSafeUri());
69        this.add(image);
70      }
71      this.add(new ChildRenderer(model), 20, 10);
72      this.add(new ResourceRenderer(model), 30, 80);
73      this.setVisible(model.isVisible());
74    }
75  }
```

Listing A87: BigFieldRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.event.dom.client.ClickEvent;
5  import com.google.gwt.event.dom.client.ClickHandler;
6  import com.google.gwt.uibinder.client.UiBinder;
7  import com.google.gwt.user.client.ui.AbsolutePanel;
8  import com.google.gwt.user.client.ui.Composite;
9  import com.google.gwt.user.client.ui.HasWidgets;
10 import com.google.gwt.user.client.ui.Widget;
11
12 import de.tu_freiberg.informatik.vonwenckstern.client.model.CardFieldModel
       ;
13 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       CardFieldPresenter;
14 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
15
16 public class CardFieldView extends Composite implements CardFieldPresenter
       .Display{
17
18   private static final Binder binder = GWT.create(Binder.class);
19   interface Binder extends UiBinder<AbsolutePanel, CardFieldView> {
20   }
21   private AbsolutePanel panel = null;
22
23   public CardFieldView() {
24     panel = binder.createAndBindUi(this);
25     initWidget(panel);
26   }
27
28
29   @Override
30   public void registerHandlers(ClickHandler p) {
31     for(int i=0; i<panel.getWidgetCount(); i++) {
32       Widget w = panel.getWidget(i);
33       if(w instanceof Renderer) {
34         w.addDomHandler(p, ClickEvent.getType());
35       }
36     }
37   }
38
39   @Override
40   public void showOneMoreBigCard(int round) {
41   }
42
43
44   @Override
45   public HasWidgets getPanel() {
46     return panel;
47   }
48
49
50   @Override
51   public void update(CardFieldModel model) {
52     ((BigFieldRenderer)panel.getWidget(1)).setModel(model.getModelOneWood
           ());
53     ((BigFieldRenderer)panel.getWidget(2)).setModel(model.getModelTwoClay
           ());
54     ((BigFieldRenderer)panel.getWidget(3)).setModel(model.getModelTwoWood
           ());
```

```
55      ((BigFieldRenderer)panel.getWidget(4)).setModel(model.
            getModelReedStoneFood());
56      ((BigFieldRenderer)panel.getWidget(5)).setModel(model.getModelCabaret
            ());
57
58      ((TooltipImageChildRenderer)panel.getWidget(6)).setModel(model.
            getModelHouse());
59      ((TooltipImageChildRenderer)panel.getWidget(7)).setModel(model.
            getModelStartPlayer());
60      ((TooltipImageChildRenderer)panel.getWidget(8)).setModel(model.
            getModelGrain());
61      ((TooltipImageChildRenderer)panel.getWidget(9)).setModel(model.
            getModelPlowField());
62      ((TooltipImageChildRenderer)panel.getWidget(11)).setModel(model.
            getModelFood());
63    }
64
65  }
```

Listing A88: CardFieldView.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       desktop'>
5    <ui:style>
6      .panel {
7        background-color: ivory;
8        cursor: pointer;
9      }
10     .handcursor {
11       cursor: pointer;
12       }
13       .nocursor {
14       cursor: not-allowed;
15     }
16   </ui:style>
17   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BackgroundCard.*" />
18   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         Resource.*" />
19   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         Child.*" />
20   <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
         .resources.Images'/>
21     <g:AbsolutePanel width="1000px" height="1000px">
22       <g:at left="0" top="0">
23         <g:HTML>
24           <img src="{im.cardField.getSafeUri}" width="400px" />
25         </g:HTML>
26       </g:at>
27
28       <g:at left="27" top="32">
29         <a:BigFieldRenderer styleName="{style.handcursor}"/>
30       </g:at>
31       <g:at left="142" top="32">
32         <a:BigFieldRenderer styleName="{style.handcursor}"/>
33       </g:at>
34
```

```
35        <g:at left="27" top="205">
36          <a:BigFieldRenderer styleName="{style.handcursor}"/>
37        </g:at>
38        <g:at left="142" top="205">
39          <a:BigFieldRenderer styleName="{style.handcursor}"/>
40        </g:at>
41
42        <g:at left="27" top="390">
43          <a:BigFieldRenderer styleName="{style.handcursor}" />
44           <!-- title="get␣food" -->
45
46        </g:at>
47        <g:at left="265" top="20">
48            <a:TooltipImageChildRenderer width="65px" height="65px"
                  styleName="{style.handcursor}" value="buildHouse" url="{im.
                  clear.getSafeUri.asString}">
49            <a:tooltip>
50              <g:HTML>
51              <b>build a house</b><br/>
52              wood house: 5 <img src="{im.woodIcon.getSafeUri}" width="20px"
                    /> + 2 <img src="{im.reedIcon.getSafeUri}" width="20px"/><
                    br/>
53              clay house: 5 <img src="{im.clayIcon.getSafeUri}" width="20px"
                    /> + 2 <img src="{im.reedIcon.getSafeUri}" width="20px"/><
                    br/>
54              stone house: 5 <img src="{im.stoneIcon.getSafeUri}" width="20
                    px"/> + 2 <img src="{im.reedIcon.getSafeUri}" width="20px"
                    /><br/>
55              <br/><br/>and/or <b>build stables</b><br/>
56              2 woods per stable
57              </g:HTML>
58            </a:tooltip>
59          </a:TooltipImageChildRenderer>
60        </g:at>
61
62        <g:at left="265" top="110">
63          <a:TooltipImageChildRenderer width="65px" height="65px" styleName=
                "{style.handcursor}" url="{im.clear.getSafeUri.asString}">
64            <a:tooltip>
65              <g:HTML>
66              become start player
67              </g:HTML>
68            </a:tooltip>
69          </a:TooltipImageChildRenderer>
70        </g:at>
71
72        <g:at left="265" top="205">
73          <a:TooltipImageChildRenderer width="65px" height="65px" styleName=
                "{style.handcursor}" url="{im.clear.getSafeUri.asString}">
74            <a:tooltip>
75              <g:HTML>
76              get one grain<br/>
77              + <img src="{im.grainIcon.getSafeUri}" width="20px"/>
78              </g:HTML>
79            </a:tooltip>
80          </a:TooltipImageChildRenderer>
81        </g:at>
82
83        <g:at left="265" top="290">
84          <a:TooltipImageChildRenderer width="65px" height="65px" styleName=
                "{style.handcursor}" url="{im.clear.getSafeUri.asString}">
```

```
85            <a:tooltip>
86              <g:HTML>
87              plow one field
88              </g:HTML>
89            </a:tooltip>
90          </a:TooltipImageChildRenderer>
91        </g:at>
92
93        <g:at left="265" top="380">
94          <a:TooltipImageChildRenderer width="65px" height="65px" styleName=
                "{style.nocursor}" url="{im.clear.getSafeUri.asString}">
95            <a:tooltip>
96              <g:HTML>
97              this operation is not implemented
98              </g:HTML>
99            </a:tooltip>
100         </a:TooltipImageChildRenderer>
101       </g:at>
102
103       <g:at left="265" top="470">
104         <a:TooltipImageChildRenderer width="65px" height="65px" styleName=
                "{style.handcursor}" url="{im.clear.getSafeUri.asString}">
105           <a:tooltip>
106             <g:HTML>
107             get two food marers<br/>
108             + <img src="{im.foodMarker.getSafeUri}" width="20px"/><img src
                    ="{im.foodMarker.getSafeUri}" width="20px"/>
109             </g:HTML>
110           </a:tooltip>
111         </a:TooltipImageChildRenderer>
112       </g:at>
113     </g:AbsolutePanel>
114 </ui:UiBinder>
```

Listing A89: CardFieldView.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import com.google.gwt.resources.client.ImageResource;
4  import com.google.gwt.user.client.ui.AbsolutePanel;
5  import com.google.gwt.user.client.ui.Image;
6
7  import de.tu_freiberg.informatik.vonwenckstern.client.model.BaseFieldModel
       ;
8  import de.tu_freiberg.informatik.vonwenckstern.client.resources.Images;
9
10 public class ChildRenderer extends AbsolutePanel {
11   private BaseFieldModel model;
12   public ChildRenderer() {
13     this.setPixelSize(55, 55);
14   }
15
16   public ChildRenderer(BaseFieldModel model) {
17     this();
18     setModel(model);
19   }
20
21   public void setModel(BaseFieldModel model) {
22     this.model = model;
23     if(model != null)
24       render();
```

```
25    }
26
27    public void render() {
28      this.clear();
29      Images im = Images.Util.getInstance();
30      ImageResource child = null;
31      switch(model.getChild()) {
32      case C_BLUE: child = im.childBlue(); break;
33      case C_GREEN: child = im.childGreen(); break;
34      case C_NONE: child = null; break;
35      case C_RED: child = im.childRed(); break;
36      case C_ROSA: child = im.childRosa(); break;
37      }
38      if(child != null) {
39        Image image = new Image();
40        image.setHeight("50px");
41        image.setUrl(child.getSafeUri());
42        this.add(image,0,0);
43      }
44      this.setVisible(child != null);
45    }
46  }
```

Listing A90: ChildRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.uibinder.client.UiBinder;
5  import com.google.gwt.uibinder.client.UiField;
6  import com.google.gwt.user.client.ui.Composite;
7  import com.google.gwt.user.client.ui.HTML;
8  import com.google.gwt.user.client.ui.HorizontalPanel;
9  import com.google.gwt.user.client.ui.Image;
10 import com.google.gwt.user.client.ui.IntegerBox;
11 import com.google.gwt.user.client.ui.Widget;
12
13 import de.tu_freiberg.informatik.vonwenckstern.client.model.BigFieldModel;
14 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       PlayerResourceModel;
15 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       InfoViewPresenter;
16 import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
17
18 public class InfoView extends Composite implements InfoViewPresenter.
       Display {
19
20   private static final Binder binder = GWT.create(Binder.class);
21
22   interface Binder extends UiBinder<HorizontalPanel, InfoView> {
23   }
24
25   private HorizontalPanel panel = null;
26
27   public InfoView() {
28     panel = binder.createAndBindUi(this);
29     initWidget(panel);
30     new Tooltip(beggarCard, new HTML("This␣is␣a␣beggar␣card.<br>You␣
           receive␣this␣card,␣if␣you␣have␣not␣enough␣food␣family.<br><br>
           Every␣cards␣gives␣you␣minus␣<b>3</b>␣points␣at␣the␣end."));
```

```
31      new Tooltip(childCard, new HTML("This are your free persons, which you
            can add to your family by playing the card <b>making a baby</b>."
            ));
32      String sStable = "Your available stables, which you can add to your
            fields by playing <b>build a house</b> and/or <b>build stables</b
            >.";
33      new Tooltip(stableCard, new HTML(sStable));
34      new Tooltip(stableCounter, new HTML(sStable));
35      String sFence = "Your available fence parts, which you need to fence
            your fields.";
36      new Tooltip(fenceCard, new HTML(sFence));
37      new Tooltip(fenceCounter, new HTML(sFence));
38    }
39
40    @UiField
41    BigFieldModel modelWood;
42    @UiField
43    BigFieldModel modelClay;
44    @UiField
45    BigFieldModel modelStone;
46    @UiField
47    BigFieldModel modelReed;
48    @UiField
49    BigFieldModel modelFood;
50    @UiField
51    BigFieldModel modelGrain;
52    @UiField
53    BigFieldModel modelVegetable;
54    @UiField
55    Image beggarCard;
56    @UiField
57    IntegerBox beggarCounter;
58    @UiField
59    Image childCard;
60    @UiField
61    IntegerBox childCounter;
62    @UiField
63    Image stableCard;
64    @UiField
65    IntegerBox stableCounter;
66    @UiField
67    HTML fenceCard;
68    @UiField
69    IntegerBox fenceCounter;
70
71
72    @Override
73    public void updateView(PlayerResourceModel model) {
74      modelWood.setRessourceCount(model.getWoodCount());
75      modelClay.setRessourceCount(model.getClayCount());
76      modelStone.setRessourceCount(model.getStoneCount());
77      modelReed.setRessourceCount(model.getReedCount());
78      modelFood.setRessourceCount(model.getFoodCount());
79      modelGrain.setRessourceCount(model.getGrainCount());
80      modelVegetable.setRessourceCount(model.getVegetableCount());
81
82      beggarCard.setVisible(model.getBeggerCards() > 0);
83      beggarCounter.setVisible(model.getBeggerCards() > 1);
84      beggarCounter.setValue(model.getBeggerCards());
85
86      childCard.setVisible(model.getPersonsCount() > 0);
```

```
87      childCounter.setVisible(model.getPersonsCount() > 1);
88      childCounter.setValue(model.getPersonsCount());
89
90      stableCard.setVisible(model.getStableCount() > 0);
91      stableCounter.setVisible(model.getStableCount() > 1);
92      stableCounter.setValue(model.getStableCount());
93
94      fenceCard.setVisible(model.getFenceCount() > 0);
95      fenceCounter.setVisible(model.getFenceCount() > 1);
96      fenceCounter.setValue(model.getFenceCount());
97
98      for(int i=0; i<panel.getWidgetCount(); i++) {
99        Widget w = panel.getWidget(i);
100       if(w instanceof ResourceRenderer) {
101         ((ResourceRenderer) w).render();
102       }
103     }
104   }
105 }
```

Listing A91: InfoView.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       desktop'>
5    <ui:style>
6      .handcursor {
7        cursor: pointer;
8        }
9    </ui:style>
10   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BackgroundCard.*" />
11   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         Resource.*" />
12   <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
         .resources.Images'/>
13   <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BigFieldModel" field="modelWood">
14     <ui:attributes ressource="{R_WOOD}" ressourceCount="0"
           ressourceRoundAddition="0" description="your␣wood␣resources"/>
15   </ui:with>
16   <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BigFieldModel" field="modelClay">
17     <ui:attributes ressource="{R_CLAY}" ressourceCount="0"
           ressourceRoundAddition="0" description="your␣clay␣resources"/>
18   </ui:with>
19   <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BigFieldModel" field="modelStone">
20     <ui:attributes ressource="{R_STONE}" ressourceCount="0"
           ressourceRoundAddition="0" description="your␣stone␣resources"/>
21   </ui:with>
22   <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BigFieldModel" field="modelReed">
23     <ui:attributes ressource="{R_REED}" ressourceCount="0"
           ressourceRoundAddition="0" description="your␣reed␣resources"/>
24   </ui:with>
25   <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BigFieldModel" field="modelFood">
```

```
26      <ui:attributes ressource="{R_FOOD}" ressourceCount="3"
            ressourceRoundAddition="0" description="your␣food␣markers"/>
27    </ui:with>
28    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
          BigFieldModel" field="modelGrain">
29      <ui:attributes ressource="{R_GRAIN}" ressourceCount="0"
            ressourceRoundAddition="0" description="your␣grain␣resources"/>
30    </ui:with>
31    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
          BigFieldModel" field="modelVegetable">
32      <ui:attributes ressource="{R_VEGETABLE}" ressourceCount="0"
            ressourceRoundAddition="0" description="your␣vegetable␣resources"
            />
33    </ui:with>
34
35    <g:HorizontalPanel>
36      <a:ResourceRenderer model="{modelWood}" styleName="{style.handcursor}"
            />
37      <a:ResourceRenderer model="{modelClay}" styleName="{style.handcursor}"
            />
38      <a:ResourceRenderer model="{modelStone}" styleName="{style.handcursor}
            "/>
39      <a:ResourceRenderer model="{modelReed}" styleName="{style.handcursor}"
            />
40      <a:ResourceRenderer model="{modelFood}" styleName="{style.handcursor}"
            />
41      <a:ResourceRenderer model="{modelGrain}" styleName="{style.handcursor}
            "/>
42      <a:ResourceRenderer model="{modelVegetable}" styleName="{style.
            handcursor}"/>
43      <g:AbsolutePanel width="50px" height="50px">
44        <g:at left="0" top="0">
45          <g:Image url="{im.childBlue.getSafeUri.asString}" width="50px" ui:
                field="childCard"/>
46        </g:at>
47        <g:at left="18" top="14">
48          <g:IntegerBox value="3" visibleLength="1" readOnly="true" width="
                10px" ui:field="childCounter"/>
49        </g:at>
50      </g:AbsolutePanel>
51      <g:AbsolutePanel width="30px" height="50px">
52        <g:at left="0" top="0">
53          <g:Image url="{im.houseBlue.getSafeUri.asString}" height="20px" ui
                :field="stableCard"/>
54        </g:at>
55        <g:at left="5" top="25">
56          <g:IntegerBox value="4" visibleLength="1" readOnly="true" width="
                15px" ui:field="stableCounter"/>
57        </g:at>
58      </g:AbsolutePanel>
59      <g:AbsolutePanel width="30px" height="50px">
60        <g:at left="0" top="0">
61          <g:HTML ui:field="fenceCard">
62            <div style="width:␣10px;␣height:␣50px;␣border-left:␣blue␣solid␣5
                  px"></div>
63          </g:HTML>
64        </g:at>
65        <g:at left="7" top="25">
66          <g:IntegerBox value="15" visibleLength="1" readOnly="true" width="
                15px" ui:field="fenceCounter"/>
67        </g:at>
```

```
68      </g:AbsolutePanel>
69      <g:AbsolutePanel width="100px" height="300px">
70        <g:at left="0" top="0">
71          <g:Image url="{im.beggarCard.getSafeUri.asString}" width="100px"
                ui:field="beggarCard" visible="false"/>
72        </g:at>
73        <g:at left="75" top="65">
74          <g:IntegerBox value="20" visibleLength="1" readOnly="true" width="
                15px" ui:field="beggarCounter" visible="false"/>
75        </g:at>
76      </g:AbsolutePanel>
77    </g:HorizontalPanel>
78  </ui:UiBinder>
```

Listing A92: CardFieldView.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.event.dom.client.ClickEvent;
5  import com.google.gwt.event.dom.client.ClickHandler;
6  import com.google.gwt.uibinder.client.UiBinder;
7  import com.google.gwt.uibinder.client.UiField;
8  import com.google.gwt.user.client.ui.AbsolutePanel;
9  import com.google.gwt.user.client.ui.Composite;
10 import com.google.gwt.user.client.ui.Label;
11 import com.google.gwt.user.client.ui.PushButton;
12 import com.google.gwt.user.client.ui.Widget;
13
14 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       PlayerFieldModel;
15 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       PlayerFieldPresenter.Display;
16 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
17
18 public class PlayerFieldView extends Composite implements Display {
19
20   private static final Binder binder = GWT.create(Binder.class);
21
22   interface Binder extends UiBinder<AbsolutePanel, PlayerFieldView> {
23   }
24
25   private AbsolutePanel panel = null;
26
27   @UiField
28   Label info;
29   @UiField
30   PushButton btnEnclosure;
31   @UiField
32   PushButton btnFeedingFamily;
33
34   public PlayerFieldView() {
35     panel = binder.createAndBindUi(this);
36     initWidget(panel);
37   }
38
39   @Override
40   public void update(PlayerFieldModel model) {
41     for(int i=0; i<panel.getWidgetCount(); i++) {
42       Widget w = panel.getWidget(i);
43       if(w instanceof SmallFieldRenderer) {
```

```
44          SmallFieldRenderer s = (SmallFieldRenderer) w;
45          s.setModel(model.getField(s.getPosition()));
46        } else if(w instanceof AcquisitionCardRenderer) {
47          AcquisitionCardRenderer a = (AcquisitionCardRenderer)w;
48          a.setModel(model.getAcquisitation(a.getPosition()));
49        }
50      }
51    }
52
53    @Override
54    public void registerHandlers(ClickHandler p) {
55      for(int i=0; i<panel.getWidgetCount(); i++) {
56        Widget w = panel.getWidget(i);
57        if(w instanceof Renderer) {
58          w.addDomHandler(p, ClickEvent.getType());
59        }
60      }
61      btnEnclosure.addClickHandler(p);
62      btnFeedingFamily.addClickHandler(p);
63    }
64
65    @Override
66    /** sets the text and makes the information visible */
67    public void setInformation(String text) {
68      info.setText(text);
69      info.setVisible(true);
70    }
71
72    @Override
73    public void setInformationVisible(boolean visible) {
74      info.setVisible(visible);
75    }
76
77    @Override
78    public void setEnclosureBtnVisible(boolean visible) {
79      btnEnclosure.setVisible(visible);
80    }
81
82    @Override
83    public void setFeedingFamilyBtnVisible(boolean visible) {
84      btnFeedingFamily.setVisible(visible);
85    }
86
87  }
```

Listing A93: PlayerFieldView.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       desktop'>
5    <ui:style>
6      .panel {
7        background-color: blue;
8        cursor: pointer;
9      }
10     .handcursor {
11       cursor: pointer;
12       }
```

```
13  </ui:style>
14  <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
        FieldCard.*" />
15  <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
        Resource.*" />
16  <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
        Player.*" />
17  <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
        Child.*" />
18  <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
        .resources.Images'/>
19  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field1">
20    <ui:attributes field="{F_NONE}" player="{BLUE}" />
21  </ui:with>
22  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field2">
23    <ui:attributes field="{F_NONE}" player="{BLUE}" />
24  </ui:with>
25  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field3">
26    <ui:attributes field="{F_NONE}" player="{BLUE}" />
27  </ui:with>
28  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field4">
29    <ui:attributes field="{F_NONE}" player="{BLUE}" />
30  </ui:with>
31  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field5">
32    <ui:attributes field="{F_WOOD_HOUSE}" child="{C_BLUE}" player="{BLUE}"
          personsCount="1"/>
33  </ui:with>
34  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field6">
35    <ui:attributes field="{F_NONE}" player="{BLUE}" />
36  </ui:with>
37  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field7">
38    <ui:attributes field="{F_NONE}" player="{BLUE}" />
39  </ui:with>
40  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field8">
41    <ui:attributes field="{F_NONE}" player="{BLUE}" />
42  </ui:with>
43  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field9">
44    <ui:attributes field="{F_NONE}" player="{BLUE}" />
45  </ui:with>
46  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field10">
47    <ui:attributes field="{F_WOOD_HOUSE}" child="{C_BLUE}" player="{BLUE}"
          personsCount="1"/>
48  </ui:with>
49  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field11">
50    <ui:attributes field="{F_NONE}" player="{BLUE}" />
51  </ui:with>
52  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field12">
53    <ui:attributes field="{F_NONE}" player="{BLUE}" />
54  </ui:with>
```

```
55  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field13">
56    <ui:attributes field="{F_NONE}" player="{BLUE}" />
57  </ui:with>
58  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field14">
59    <ui:attributes field="{F_NONE}" player="{BLUE}" />
60  </ui:with>
61  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field15">
62    <ui:attributes field="{F_NONE}" player="{BLUE}" />
63  </ui:with>
64
65    <g:AbsolutePanel width="1000px" height="1000px">
66      <g:at left="0" top="0">
67        <g:HTML>
68          <img src="{im.playerField.getSafeUri}" height="400px" />
69        </g:HTML>
70      </g:at>
71      <g:at left="55" top="335">
72        <g:Label text="Information" ui:field="info" visible="false"/>
73      </g:at>
74      <g:at left="449" top="335">
75        <g:PushButton text="new␣enclosure" ui:field="btnEnclosure" visible
            ="false"/>
76      </g:at>
77      <g:at left="229" top="335">
78        <g:PushButton text="finish␣special␣events␣and␣continue␣feeding␣
            your␣family" ui:field="btnFeedingFamily" visible="false"/>
79      </g:at>
80
81      <g:at left="55" top="35">
82        <a:SmallFieldRenderer model="{field1}" styleName="{style.
            handcursor}" position="0"/>
83      </g:at>
84      <g:at left="159" top="35">
85        <a:SmallFieldRenderer model="{field2}" styleName="{style.
            handcursor}" position="1"/>
86      </g:at>
87      <g:at left="261" top="35">
88        <a:SmallFieldRenderer model="{field3}" styleName="{style.
            handcursor}" position="2"/>
89      </g:at>
90      <g:at left="363" top="35">
91        <a:SmallFieldRenderer model="{field4}" styleName="{style.
            handcursor}" position="3"/>
92      </g:at>
93      <g:at left="465" top="35">
94        <a:SmallFieldRenderer model="{field5}" styleName="{style.
            handcursor}" position="4"/>
95      </g:at>
96
97      <g:at left="55" top="140">
98        <a:SmallFieldRenderer model="{field6}" styleName="{style.
            handcursor}" position="5"/>
99      </g:at>
100     <g:at left="159" top="140">
101       <a:SmallFieldRenderer model="{field7}" styleName="{style.
            handcursor}" position="6"/>
102     </g:at>
103     <g:at left="261" top="140">
```

```
104         <a:SmallFieldRenderer model="{field8}" styleName="{style.
                handcursor}" position="7"/>
105       </g:at>
106       <g:at left="363" top="140">
107         <a:SmallFieldRenderer model="{field9}" styleName="{style.
                handcursor}" position="8"/>
108       </g:at>
109       <g:at left="465" top="140">
110         <a:SmallFieldRenderer model="{field10}" styleName="{style.
                handcursor}" position="9"/>
111       </g:at>
112
113       <g:at left="55" top="242">
114         <a:SmallFieldRenderer model="{field11}" styleName="{style.
                handcursor}" position="10"/>
115       </g:at>
116       <g:at left="159" top="242">
117         <a:SmallFieldRenderer model="{field12}" styleName="{style.
                handcursor}" position="11"/>
118       </g:at>
119       <g:at left="261" top="242">
120         <a:SmallFieldRenderer model="{field13}" styleName="{style.
                handcursor}" position="12"/>
121       </g:at>
122       <g:at left="363" top="242">
123         <a:SmallFieldRenderer model="{field14}" styleName="{style.
                handcursor}" position="13"/>
124       </g:at>
125       <g:at left="465" top="242">
126         <a:SmallFieldRenderer model="{field15}" styleName="{style.
                handcursor}" position="14"/>
127       </g:at>
128
129       <g:at left="475" top="400">
130         <a:AcquisitionCardRenderer position="0"/>
131       </g:at>
132       <g:at left="405" top="400">
133         <a:AcquisitionCardRenderer position="1"/>
134       </g:at>
135       <g:at left="335" top="400">
136         <a:AcquisitionCardRenderer position="2"/>
137       </g:at>
138       <g:at left="265" top="400">
139         <a:AcquisitionCardRenderer position="3"/>
140       </g:at>
141       <g:at left="195" top="400">
142         <a:AcquisitionCardRenderer position="4"/>
143       </g:at>
144       <g:at left="125" top="400">
145         <a:AcquisitionCardRenderer position="5"/>
146       </g:at>
147       <g:at left="55" top="400">
148         <a:AcquisitionCardRenderer position="6"/>
149       </g:at>
150       <g:at left="475" top="515">
151         <a:AcquisitionCardRenderer position="7"/>
152       </g:at>
153       <g:at left="405" top="515">
154         <a:AcquisitionCardRenderer position="8"/>
155       </g:at>
156       <g:at left="335" top="515">
```

```
157         <a:AcquisitionCardRenderer position="9"/>
158       </g:at>
159     </g:AbsolutePanel>
160 </ui:UiBinder>
```

Listing A94: PlayerFieldView.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import com.google.gwt.safehtml.shared.SafeHtmlBuilder;
4  import com.google.gwt.user.client.ui.AbsolutePanel;
5  import com.google.gwt.user.client.ui.HTML;
6
7  import de.tu_freiberg.informatik.vonwenckstern.client.model.BigFieldModel;
8  import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasBaseFieldModel;
9  import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
10 import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
11 public class ResChildRenderer extends AbsolutePanel implements Renderer,
       HasBaseFieldModel {
12   private BigFieldModel model;
13   private boolean childLeft = true;
14
15   public void setChildLeft(boolean b) {
16     childLeft = b;
17     if(model != null)
18       render();
19   }
20
21   public ResChildRenderer() {
22     this.setPixelSize(100, 65);
23   }
24
25   public BigFieldModel getModel() {
26     return model;
27   }
28
29   public void setModel(BigFieldModel model) {
30     this.model = model;
31     if(model != null) {
32       if(model.getDescription() != null)
33         new Tooltip(this, new HTML(new SafeHtmlBuilder().
               appendEscapedLines(model.getDescription()).toSafeHtml()));
34       render();
35     }
36   }
37
38   @Override
39   public void render() {
40     this.clear();
41     this.add(new ChildRenderer(model), childLeft ? 0 : 50, 10);
42     this.add(new ResourceRenderer(model), childLeft ? 65 : 0, 0);
43   }
44
45 }
```

Listing A95: ResChildRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import com.google.gwt.resources.client.ImageResource;
4  import com.google.gwt.safehtml.shared.SafeHtmlBuilder;
5  import com.google.gwt.user.client.ui.AbsolutePanel;
6  import com.google.gwt.user.client.ui.HTML;
7  import com.google.gwt.user.client.ui.Image;
8  import com.google.gwt.user.client.ui.IntegerBox;
9
10 import de.tu_freiberg.informatik.vonwenckstern.client.model.BaseFieldModel
       ;
11 import de.tu_freiberg.informatik.vonwenckstern.client.resources.Images;
12 import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
13
14 public class ResourceRenderer extends AbsolutePanel {
15   private BaseFieldModel model;
16   public ResourceRenderer() {
17     this.setPixelSize(30, 50);
18   }
19
20   public ResourceRenderer(BaseFieldModel model) {
21     this();
22     setModel(model);
23   }
24
25   public BaseFieldModel getModel() {
26     return model;
27   }
28
29   public void setModel(BaseFieldModel model) {
30     this.model = model;
31     if(model != null) {
32       if(model.getDescription() != null)
33         new Tooltip(this, new HTML(new SafeHtmlBuilder().
               appendEscapedLines(model.getDescription()).toSafeHtml()));
34       render();
35     }
36   }
37
38   public void render() {
39     this.clear();
40     Images im = Images.Util.getInstance();
41     ImageResource resImage = null;
42     switch(model.getRessource()) {
43     case R_BOAR: resImage = im.boarMarker(); break;
44     case R_CLAY: resImage = im.clayStone(); break;
45     case R_COW: resImage = im.cowMarker(); break;
46     case R_FOOD: resImage = im.foodMarker(); break;
47     case R_GRAIN: resImage = im.grainStone(); break;
48     case R_NONE: resImage = null; break;
49     case R_REED: resImage = im.reedStone(); break;
50     case R_SHEEP: resImage = im.sheepMarker(); break;
51     case R_STONE: resImage = im.stoneStone(); break;
52     case R_VEGETABLE: resImage = im.vegetableStone(); break;
53     case R_WOOD: resImage = im.woodStone(); break;
54     }
55     if((resImage != null) && (model.getRessourceCount() > 0)) {
56       Image image = new Image();
57       image.setWidth("30px");
58       image.setUrl(resImage.getSafeUri());
59       this.add(image, 0, 25);
```

```
60        if(model.getRessourceCount() > 1) {
61          IntegerBox tb = new IntegerBox();
62          tb.setVisibleLength(2);
63          tb.setValue(model.getRessourceCount());
64          tb.setReadOnly(true);
65          this.add(tb, 0, 0);
66        }
67      }
68      this.setVisible((resImage != null) && (model.getRessourceCount() > 0))
            ;
69    }
70  }
```

Listing A96: ResourceRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import java.util.Iterator;
4
5  import com.google.gwt.core.client.GWT;
6  import com.google.gwt.event.dom.client.ClickEvent;
7  import com.google.gwt.event.dom.client.ClickHandler;
8  import com.google.gwt.uibinder.client.UiBinder;
9  import com.google.gwt.user.client.ui.AbsolutePanel;
10 import com.google.gwt.user.client.ui.Composite;
11 import com.google.gwt.user.client.ui.HasWidgets;
12 import com.google.gwt.user.client.ui.Widget;
13
14 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       Rounds1To7Model;
15 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       Rounds1To7Presenter;
16 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
17 import de.tu_freiberg.informatik.vonwenckstern.client.view.desktop.
       BigFieldRenderer;
18 public class Rounds1To7View extends Composite implements
       Rounds1To7Presenter.Display {
19
20   private static final Binder binder = GWT.create(Binder.class);
21
22   interface Binder extends UiBinder<AbsolutePanel, Rounds1To7View> {
23   }
24
25   private AbsolutePanel panel = null;
26
27   public Rounds1To7View() {
28     panel = binder.createAndBindUi(this);
29     initWidget(panel);
30   }
31
32   @Override
33   public void registerHandlers(ClickHandler p) {
34     for(int i=0; i<panel.getWidgetCount(); i++) {
35       Widget w = panel.getWidget(i);
36       if(w instanceof Renderer) {
37         w.addDomHandler(p, ClickEvent.getType());
38       }
39     }
40   }
41
42   @Override
```

```
43    public void showOneMoreBigCard(int round) {
44      Iterator<Widget> it = panel.iterator();
45      while(it.hasNext()) {
46        Widget w = it.next();
47        if(w instanceof BigFieldRenderer) {
48          if(!w.isVisible()) {
49            w.setVisible(true);
50            ((BigFieldRenderer)w).getModel().setVisible(true);
51            return;
52          }
53        }
54      }
55    }
56
57    @Override
58    public HasWidgets getPanel() {
59      return panel;
60    }
61
62    @Override
63    public void update(Rounds1To7Model model) {
64      ((BigFieldRenderer)panel.getWidget(1)).setModel(model.getModelSheep())
          ;
65      ((BigFieldRenderer)panel.getWidget(2)).setModel(model.
          getModelBigAcquisitation());
66      ((BigFieldRenderer)panel.getWidget(3)).setModel(model.getModelFence())
          ;
67      ((BigFieldRenderer)panel.getWidget(4)).setModel(model.
          getModelSeedingBacking());
68      ((BigFieldRenderer)panel.getWidget(5)).setModel(model.getModelFamily()
          );
69      ((BigFieldRenderer)panel.getWidget(6)).setModel(model.getModelStone())
          ;
70      ((BigFieldRenderer)panel.getWidget(7)).setModel(model.
          getModelRestauration());
71
72      ((ResChildRenderer)panel.getWidget(8)).setModel(model.getModelWood());
73      ((ResChildRenderer)panel.getWidget(9)).setModel(model.getModelClay());
74      ((ResChildRenderer)panel.getWidget(10)).setModel(model.getModelReed())
          ;
75      ((ResChildRenderer)panel.getWidget(11)).setModel(model.getModelFood())
          ;
76    }
77
78  }
```

Listing A97: Rounds1To7View.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       desktop'>
5    <ui:style>
6      .panel {
7        background-color: ivory;
8        cursor: pointer;
9      }
10     .handcursor {
11       cursor: pointer;
```

```
12        }
13    </ui:style>
14    <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
          BackgroundCard.*" />
15    <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
          Resource.*" />
16    <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
          Child.*" />
17    <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
          .resources.Images'/>
18      <g:AbsolutePanel width="1000px" height="1000px">
19        <g:at left="0" top="0">
20          <g:HTML>
21            <img src="{im.rounds1To7.getSafeUri}" width="400px" />
22          </g:HTML>
23        </g:at>
24
25        <g:at left="20" top="25">
26          <a:BigFieldRenderer styleName="{style.handcursor}"/>
27        </g:at>
28        <g:at left="160" top="25">
29          <a:BigFieldRenderer  styleName="{style.handcursor}" visible="false
                "/>
30        </g:at>
31        <g:at left="160" top="200">
32          <a:BigFieldRenderer  styleName="{style.handcursor}" visible="false
                "/>
33        </g:at>
34        <g:at left="160" top="370">
35          <a:BigFieldRenderer  styleName="{style.handcursor}" visible="false
                "/>
36        </g:at>
37
38
39        <g:at left="280" top="25">
40          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                />
41        </g:at>
42        <g:at left="280" top="200">
43          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                />
44        </g:at>
45        <g:at left="280" top="370">
46          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                />
47        </g:at>
48
49        <g:at left="35" top="195">
50          <a:ResChildRenderer childLeft="false" styleName="{style.handcursor
                }"/>
51        </g:at>
52
53        <g:at left="20" top="290">
54          <a:ResChildRenderer styleName="{style.handcursor}"/>
55        </g:at>
56
57        <g:at left="40" top="385">
58          <a:ResChildRenderer childLeft="false" styleName="{style.handcursor
                }"/>
59        </g:at>
60
```

```
61        <g:at left="20" top="475">
62          <a:ResChildRenderer styleName="{style.handcursor}"/>
63        </g:at>
64      </g:AbsolutePanel>
65  </ui:UiBinder>
```

Listing A98: Rounds1To7View.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import java.util.Iterator;
4
5  import com.google.gwt.core.client.GWT;
6  import com.google.gwt.event.dom.client.ClickEvent;
7  import com.google.gwt.event.dom.client.ClickHandler;
8  import com.google.gwt.uibinder.client.UiBinder;
9  import com.google.gwt.user.client.ui.AbsolutePanel;
10 import com.google.gwt.user.client.ui.Composite;
11 import com.google.gwt.user.client.ui.HasWidgets;
12 import com.google.gwt.user.client.ui.Widget;
13
14 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       Rounds8To14Model;
15 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       Rounds8To14Presenter;
16 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
17 public class Rounds8To14View extends Composite implements
       Rounds8To14Presenter.Display {
18
19   private static final Binder binder = GWT.create(Binder.class);
20
21   interface Binder extends UiBinder<AbsolutePanel, Rounds8To14View> {
22   }
23
24   private AbsolutePanel panel = null;
25
26   public Rounds8To14View() {
27     panel = binder.createAndBindUi(this);
28     initWidget(panel);
29   }
30
31   @Override
32   public void registerHandlers(ClickHandler p) {
33     for(int i=0; i<panel.getWidgetCount(); i++) {
34       Widget w = panel.getWidget(i);
35       if(w instanceof Renderer) {
36         w.addDomHandler(p, ClickEvent.getType());
37       }
38     }
39   }
40
41   @Override
42   public void showOneMoreBigCard(int round) {
43     Iterator<Widget> it = panel.iterator();
44     while(it.hasNext()) {
45       Widget w = it.next();
46       if(w instanceof BigFieldRenderer) {
47         if(!w.isVisible()) {
48           w.setVisible(true);
49           ((BigFieldRenderer)w).getModel().setVisible(true);
50           return;
```

```
51          }
52        }
53      }
54    }
55
56    @Override
57    public HasWidgets getPanel() {
58      return panel;
59    }
60
61    @Override
62    public void update(Rounds8To14Model model) {
63      ((BigFieldRenderer)panel.getWidget(1)).setModel(model.getModelBoar());
64      ((BigFieldRenderer)panel.getWidget(2)).setModel(model.
            getModelVegetable());
65      ((BigFieldRenderer)panel.getWidget(3)).setModel(model.getModelStone())
            ;
66      ((BigFieldRenderer)panel.getWidget(4)).setModel(model.getModelCow());
67      ((BigFieldRenderer)panel.getWidget(5)).setModel(model.getModelPlowSow
            ());
68      ((BigFieldRenderer)panel.getWidget(6)).setModel(model.getModelFamily()
            );
69      ((BigFieldRenderer)panel.getWidget(7)).setModel(model.
            getModelRestauration());
70    }
71
72  }
```

Listing A99: Rounds8To14View.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       desktop'>
5    <ui:style>
6      .panel {
7        background-color: ivory;
8        cursor: pointer;
9      }
10     .handcursor {
11       cursor: pointer;
12       }
13   </ui:style>
14   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BackgroundCard.*" />
15   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         Resource.*" />
16   <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
         .resources.Images'/>
17 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
       .BigFieldModel" field="modelBoar"> -->
18 <!--      <ui:attributes bgCard="{BOAR}" ressource="{R_BOAR}"
       ressourceCount="1" ressourceRoundAddition="1" description="get␣boars"
       /> -->
19 <!--    </ui:with> -->
20 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
       .BigFieldModel" field="modelVegetable"> -->
21 <!--      <ui:attributes bgCard="{VEGETABLE}" ressource="{R_NONE}"
       ressourceCount="0" ressourceRoundAddition="0" description="get␣one␣
```

```
      vegetable"/> -->
22 <!--    </ui:with> -->
23 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
      .BigFieldModel" field="modelStone"> -->
24 <!--      <ui:attributes bgCard="{STONE4}" ressource="{R_STONE}"
      ressourceCount="1" ressourceRoundAddition="1" description="get␣stones"
      /> -->
25 <!--    </ui:with> -->
26 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
      .BigFieldModel" field="modelCow"> -->
27 <!--      <ui:attributes bgCard="{COW}" ressource="{R_COW}" ressourceCount=
      "1" ressourceRoundAddition="1" description="get␣cows"/> -->
28 <!--    </ui:with> -->
29 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
      .BigFieldModel" field="modelPlowSow"> -->
30 <!--      <ui:attributes bgCard="{PLOWING_SOWING}" ressource="{R_NONE}"
      ressourceCount="0" ressourceRoundAddition="0" description="plow␣one␣
      field\nand/or\nseed␣grains␣or␣vegetables␣on␣your␣fields"/> -->
31 <!--    </ui:with> -->
32 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
      .BigFieldModel" field="modelFamily"> -->
33 <!--      <ui:attributes bgCard="{FAMILY_ADDITION5}" ressource="{R_NONE}"
      ressourceCount="0" ressourceRoundAddition="0" description="make␣a␣baby
      ␣(you␣do␣NOT␣need␣a␣free␣room)"/> -->
34 <!--    </ui:with> -->
35 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
      .BigFieldModel" field="modelRestauration"> -->
36 <!--      <ui:attributes bgCard="{RESTAURATION_FENCE}" ressource="{R_NONE}"
       ressourceCount="0" ressourceRoundAddition="0" description="restaurate
      ␣your␣homes␣\nand␣if␣wanted␣you␣can\nbuild␣your␣onw␣fence␣(1␣wood␣for␣
      each␣fence␣part)"/> -->
37 <!--    </ui:with> -->
38
39
40     <g:AbsolutePanel width="1000px" height="1000px">
41       <g:at left="0" top="0">
42         <g:HTML>
43           <img src="{im.rounds8To14.getSafeUri}" height="400px" />
44         </g:HTML>
45       </g:at>
46
47       <g:at left="20" top="35">
48         <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
               />
49       </g:at>
50       <g:at left="22" top="205">
51         <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
               />
52       </g:at>
53       <g:at left="145" top="35">
54         <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
               />
55       </g:at>
56       <g:at left="145" top="210">
57         <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
               />
58       </g:at>
59       <g:at left="270" top="30">
60         <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
               />
61       </g:at>
```

```
62        <g:at left="270" top="203">
63          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                />
64        </g:at>
65        <g:at left="390" top="30">
66          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                />
67        </g:at>
68      </g:AbsolutePanel>
69  </ui:UiBinder>
```

Listing A100: Rounds8To14View.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import com.google.gwt.resources.client.ImageResource;
4  import com.google.gwt.user.client.DOM;
5  import com.google.gwt.user.client.ui.AbsolutePanel;
6  import com.google.gwt.user.client.ui.DoubleBox;
7  import com.google.gwt.user.client.ui.Image;
8  import com.google.gwt.user.client.ui.Label;
9
10 import de.tu_freiberg.informatik.vonwenckstern.client.model.FieldCard;
11 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasBaseFieldModel;
12 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       SmallFieldModel;
13 import de.tu_freiberg.informatik.vonwenckstern.client.resources.Images;
14 import de.tu_freiberg.informatik.vonwenckstern.client.view.HasPosition;
15 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
16 public class SmallFieldRenderer extends AbsolutePanel implements Renderer,
       HasBaseFieldModel, HasPosition {
17   private SmallFieldModel model;
18   private int position;
19   public int getPosition() {
20     return position;
21   }
22
23   public void setPosition(int position) {
24     this.position = position;
25   }
26
27   public SmallFieldRenderer() {
28     this.setPixelSize(70, 70);
29   }
30
31   public SmallFieldRenderer(SmallFieldModel model) {
32     this();
33     setModel(model);
34   }
35
36   public void setModel(SmallFieldModel model) {
37     this.model = model;
38     if(model != null)
39       render();
40   }
41
42   public SmallFieldModel getModel() {
43     return model;
44   }
45
```

```
46  public void render() {
47    this.clear();
48    Images im = Images.Util.getInstance();
49    ImageResource field = null;
50    switch(model.getField()) {
51    case F_CLAY_HOUSE: field = im.clayHouse(); break;
52    case F_FIELD: field = im.fieldMarker(); break;
53    case F_NONE: field = null; break;
54    case F_STABLE: {
55        switch(model.getPlayer()) {
56        case BLUE: field = im.houseBlue(); break;
57        case GREEN: field = im.houseGreen(); break;
58        case NONE: field = null; break;
59        case ROSA: field = im.houseRosa(); break;
60        case RED: field = im.houseRed(); break;
61        }
62        break;
63      }
64    case F_STONE_HOUSE: field = im.stoneHouse(); break;
65    case F_WOOD_HOUSE: field = im.woodHouse(); break;
66    }
67    if(field != null) {
68      Image image = new Image();
69      image.setHeight((model.getField() == FieldCard.F_STABLE) ? "20px" :
            "70px");
70      image.setUrl(field.getSafeUri());
71      this.add(image,0,0);
72    }
73    if (model.getBottomFence() > 0 || model.getLeftFence() > 0|| model.
        getRightFence() > 0|| model.getTopFence() > 0) {
74      String color = null;
75      switch(model.getPlayer()) {
76      case BLUE: color = "blue"; break;
77      case GREEN: color = "green"; break;
78      case NONE: color = null; break;
79      case ROSA: color = "pink"; break;
80      case RED: color = "red"; break;
81      }
82      if(color != null) {
83        color += "␣solid␣5px";
84        String noColor = "green␣none␣0px";
85        DOM.setStyleAttribute(getElement(), "borderLeft", model.
              getLeftFence() > 0?color:noColor);
86        DOM.setStyleAttribute(getElement(), "borderRight", model.
              getRightFence() > 0?color:noColor);
87        DOM.setStyleAttribute(getElement(), "borderTop", model.getTopFence
              () > 0?color:noColor);
88        DOM.setStyleAttribute(getElement(), "borderBottom", model.
              getBottomFence() > 0?color:noColor);
89      }
90    } else {
91      DOM.setStyleAttribute(getElement(), "borderLeft", "green␣none␣0px");
92      DOM.setStyleAttribute(getElement(), "borderRight", "green␣none␣0px")
            ;
93      DOM.setStyleAttribute(getElement(), "borderTop", "green␣none␣0px");
94      DOM.setStyleAttribute(getElement(), "borderBottom", "green␣none␣0px"
            );
95    }
96    if(model.isSelectable()) {
97      Label l = new Label();
98      l.setPixelSize(70, 70);
```

```
99        DOM.setStyleAttribute(l.getElement(), "backgroundColor", "rgba
              (255,255,255,0.4)");
100       this.add(l,0,0);
101     }
102
103     if(model.getPersonsCount() - model.getPersonsAtWork() + model.
            getChildCount() > 0) {
104       this.add(new ChildRenderer(model), 8, 8);
105       if(model.getPersonsCount() - model.getPersonsAtWork() + 0.5*model.
              getChildCount() != 1) {
106         DoubleBox tb = new DoubleBox();
107         tb.setVisibleLength(2);
108         tb.setValue(model.getPersonsCount() - model.getPersonsAtWork() +
                0.5*model.getChildCount());
109         tb.setReadOnly(true);
110         this.add(tb, 10, 10);
111       }
112     }
113     this.add(new ResourceRenderer(model), 30, 20);
114   }
115 }
```

Listing A101: SmallFieldRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.desktop;
2
3  import java.util.ArrayList;
4
5  import com.google.gwt.event.dom.client.ClickEvent;
6  import com.google.gwt.event.dom.client.ClickHandler;
7  import com.google.gwt.event.dom.client.HasClickHandlers;
8  import com.google.gwt.event.shared.HandlerRegistration;
9  import com.google.gwt.uibinder.client.UiChild;
10 import com.google.gwt.user.client.ui.Image;
11 import com.google.gwt.user.client.ui.Widget;
12
13 import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
14
15 public class TooltipImage extends Image implements HasClickHandlers {
16   @UiChild(limit=1)
17   public void addTooltip(Widget tooltip) {
18     new Tooltip(this, tooltip);
19   }
20
21   private ArrayList<HandlerRegistration> clickHandlers = new  ArrayList<
         HandlerRegistration>();
22
23   public void removeAllClickHandlers() {
24     int size =clickHandlers.size()-1;
25     for(int i=size; i>=0; i--) {
26       if(clickHandlers.get(i) != null) {
27         clickHandlers.get(i).removeHandler();
28       }
29       clickHandlers.remove(i);
30     }
31   }
32
33   @Override
34   public HandlerRegistration addClickHandler(ClickHandler handler) {
35     HandlerRegistration handlerReg = this.addDomHandler(handler,
           ClickEvent.getType());
```

```
36      clickHandlers.add(handlerReg);
37      return handlerReg;
38    }
39  }
```

Listing A102: TooltipImage.java file

de.tu_freiberg.informatik.vonwenckstern.client.view.mobile package

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import com.google.gwt.user.client.DOM;
4  import com.google.gwt.user.client.ui.HTML;
5  import com.google.gwt.user.client.ui.SimplePanel;
6
7  import de.tu_freiberg.informatik.vonwenckstern.client.model.
       AcquisitionCardModel;
8  import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasAcquisitionCardModel;
9  import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
10 import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
11
12 public class AcquisitionCardRenderer extends SimplePanel implements
       Renderer, HasAcquisitionCardModel {
13   private AcquisitionCardModel model = null;
14   private int position;
15   private boolean tooltipAdded = false;
16   public int getPosition() {
17     return position;
18   }
19
20   public void setPosition(int position) {
21     this.position = position;
22   }
23
24   public AcquisitionCardRenderer() {
25     this.setPixelSize(70, 40);
26     this.setVisible(false);
27     DOM.setStyleAttribute(this.getElement(), "border", "1px␣solid␣black");
28   }
29
30   public AcquisitionCardModel getModel() {
31     return model;
32   }
33
34   public void setModel(AcquisitionCardModel model) {
35     this.model = model;
36     if(!tooltipAdded && model != null) {
37       tooltipAdded = true;
38       new Tooltip(this, new HTML(model.getDescription()));
39     }
40     if(model != null) {
41       render();
42     } else {
43       this.setVisible(false);
44     }
45   }
46
47   public void render() {
48     this.setWidget(new LabelAcquisitionRenderer(model));
```

```
49      if(model.isSelectable()) {
50        DOM.setStyleAttribute(this.getElement(), "backgroundColor", "
              lightgreen");
51      } else {
52        DOM.setStyleAttribute(this.getElement(), "backgroundColor", "white")
              ;
53      }
54      this.setVisible(model.isVisible());
55    }
56  }
```

Listing A103: AcquisitionCardRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import java.util.Iterator;
4
5  import com.google.gwt.core.client.GWT;
6  import com.google.gwt.event.dom.client.ClickHandler;
7  import com.google.gwt.uibinder.client.UiBinder;
8  import com.google.gwt.user.client.ui.Composite;
9  import com.google.gwt.user.client.ui.HasWidgets;
10 import com.google.gwt.user.client.ui.HorizontalPanel;
11 import com.google.gwt.user.client.ui.VerticalPanel;
12 import com.google.gwt.user.client.ui.Widget;
13
14 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       BigAcquisitationsModel;
15 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       BigAcquisitions;
16 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       BigAcquisitionsPresenter;
17
18 public class BigAcquisitionsFieldView extends Composite implements
       BigAcquisitionsPresenter.Display {
19
20   private static final Binder binder = GWT.create(Binder.class);
21
22   interface Binder extends UiBinder<VerticalPanel,
         BigAcquisitionsFieldView> {
23   }
24
25   private VerticalPanel panel = null;
26
27   public BigAcquisitionsFieldView() {
28     panel = binder.createAndBindUi(this);
29     initWidget(panel);
30   }
31
32   @Override
33   public void registerHandlers(ClickHandler p) {
34     regHandlers(p, panel);
35   }
36
37
38   private void regHandlers(ClickHandler p, HasWidgets cont) {
39     Iterator<Widget> it = cont.iterator();
40     while(it.hasNext()) {
41       Widget w = it.next();
42       if(w instanceof LabelAcquisitionRenderer) {
43         ((LabelAcquisitionRenderer) w).addClickHandler(p);
```

```
44        } else if(w instanceof HasWidgets) {
45          regHandlers(p, (HasWidgets) w);
46        }
47      }
48    }
49
50    @Override
51    public void hideAcquisition(BigAcquisitions aquisition) {
52      hideAcq(aquisition, panel);
53    }
54
55    private void hideAcq(BigAcquisitions aquisition, HasWidgets cont) {
56      Iterator<Widget> it = cont.iterator();
57      while(it.hasNext()) {
58        Widget w = it.next();
59        if(w instanceof LabelAcquisitionRenderer &&
60            (((LabelAcquisitionRenderer) w).getModel().getAcquisition() ==
                aquisition)) {
61          w.setVisible(false);
62          ((LabelAcquisitionRenderer) w).getModel().setVisible(false);
63        } else if(w instanceof HasWidgets) {
64          hideAcq(aquisition, (HasWidgets) w);
65        }
66      }
67    }
68
69    @Override
70    public void update(BigAcquisitationsModel model) {
71      HorizontalPanel hp = (HorizontalPanel) panel.getWidget(1);
72      for(int i=0; i<5; i++) {
73        ((LabelAcquisitionRenderer)hp.getWidget(i)).setModel(model.getModel(
              i));
74      }
75      hp = (HorizontalPanel) panel.getWidget(2);
76      for(int i=0; i<5; i++) {
77        ((LabelAcquisitionRenderer)hp.getWidget(i)).setModel(model.getModel(
              i+5));
78      }
79    }
80
81  }
```

Listing A104: BigAcquisitionsFieldView.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       mobile'>
5    <ui:style>
6      .handcursor {
7        cursor: pointer;
8        }
9    </ui:style>
10   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BigAcquisitions.*" />
11   <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
         .resources.Images'/>
12
13     <g:VerticalPanel width="400px" height="100px">
```

```
14        <g:HTML>
15          Big Aquisitions
16        </g:HTML>
17        <g:HorizontalPanel width="400px">
18          <a:LabelAcquisitionRenderer bigAcquisiton="{BA_FIRE_PLACE}">
19            <a:tooltip>
20              <g:HTML>
21              <b>fire place</b>  2x clay<br/>
22              <i>Vegetable</i> 1x vegetable &rarr; 2x food<br/>
23              <i>Sheep</i> 1x sheep &rarr; 2x food<br/>
24              <i>Boar</i> 1x boar &rarr; 2x food<br/>
25              <i>Cow</i> 1x cow &rarr; 3x food<br/>
26              <br/><br/>for the action <b>backing bread</b>:<br/>
27              <i>Grain</i> 1x grain &rarr; 2x food
28              </g:HTML>
29            </a:tooltip>
30          </a:LabelAcquisitionRenderer>
31          <a:LabelAcquisitionRenderer bigAcquisiton="{BA_FIRE_PLACE2}">
32            <a:tooltip>
33              <g:HTML>
34              <b>fire place</b>  3x clay<br/>
35              <i>Vegetable</i> 1x vegetable &rarr; 2x food<br/>
36              <i>Sheep</i> 1x sheep &rarr; 2x food<br/>
37              <i>Boar</i> 1x boar &rarr; 2x food<br/>
38              <i>Cow</i> 1x cow &rarr; 3x food<br/>
39              <br/><br/>for the action <b>backing bread</b>:<br/>
40              <i>Grain</i> 1x grain &rarr; 2x food
41              </g:HTML>
42            </a:tooltip>
43          </a:LabelAcquisitionRenderer>
44          <a:LabelAcquisitionRenderer bigAcquisiton="{BA_COOKERY}">
45            <a:tooltip>
46              <g:HTML>
47              <b>cookery</b>  4x clay<br/>
48              <i>Vegetable</i> 1x vegetable &rarr; 3x food<br/>
49              <i>Sheep</i> 1x sheep &rarr; 2x food<br/>
50              <i>Boar</i> 1x boar &rarr; 3x food<br/>
51              <i>Cow</i> 1x cow &rarr; 4x food<br/>
52              <br/><br/>for the action <b>backing bread</b>:<br/>
53              <i>Grain</i> 1x grain &rarr; 3x food
54              </g:HTML>
55            </a:tooltip>
56          </a:LabelAcquisitionRenderer>
57          <a:LabelAcquisitionRenderer bigAcquisiton="{BA_COOKERY2}">
58            <a:tooltip>
59              <g:HTML>
60              <b>cookery</b>  5x clay<br/>
61              <i>Vegetable</i> 1x vegetable &rarr; 3x food<br/>
62              <i>Sheep</i> 1x sheep &rarr; 2x food<br/>
63              <i>Boar</i> 1x boar &rarr; 3x food<br/>
64              <i>Cow</i> 1x cow &rarr; 4x food<br/>
65              <br/><br/>for the action <b>backing bread</b>:<br/>
66              <i>Grain</i> 1x grain &rarr; 3x food
67              </g:HTML>
68            </a:tooltip>
69          </a:LabelAcquisitionRenderer>
70          <a:LabelAcquisitionRenderer bigAcquisiton="{BA_FOUNTAIN}">
71            <a:tooltip>
72              <g:HTML>
73              <b>fountain</b>  1x wood, 3x stone<br/>
74              You will get 5 food markers.
```

```
75              </g:HTML>
76            </a:tooltip>
77          </a:LabelAcquisitionRenderer>
78        </g:HorizontalPanel>
79        <g:HorizontalPanel width="400px">
80          <a:LabelAcquisitionRenderer bigAcquisiton="{BA_CLAY_OVEN}">
81            <a:tooltip>
82              <g:HTML>
83              <b>clay oven</b> 1x stone, 3x clay<br/>
84              for the action <b>backing bread</b>:<br/>
85              <b>1x</b> 1x grain &rarr; 5x food
86              </g:HTML>
87            </a:tooltip>
88          </a:LabelAcquisitionRenderer>
89          <a:LabelAcquisitionRenderer bigAcquisiton="{BA_STONE_OVEN}">
90            <a:tooltip>
91              <g:HTML>
92              <b>stone oven</b>  1x clay, 3x stone<br/>
93              for the action <b>backing bread</b>:<br/>
94              <b>2x</b> 1x grain  &rarr; 4x food
95              </g:HTML>
96            </a:tooltip>
97          </a:LabelAcquisitionRenderer>
98          <a:LabelAcquisitionRenderer bigAcquisiton="{BA_JOINERY}">
99            <a:tooltip>
100             <g:HTML>
101             <b>joinery</b>  2x wood, 2x stone<br/>
102             in every <b>harvest season</b>:<br/>
103             <b>1x</b> 1x wood &rarr; 2x food
104             </g:HTML>
105           </a:tooltip>
106         </a:LabelAcquisitionRenderer>
107         <a:LabelAcquisitionRenderer bigAcquisiton="{BA_POTTERY}">
108           <a:tooltip>
109             <g:HTML>
110             <b>pottery</b>  2x clay, 2x stone<br/>
111             in every <b>harvest season</b>:<br/>
112             <b>1x</b> 1x clay &rarr; 2x food
113             </g:HTML>
114           </a:tooltip>
115         </a:LabelAcquisitionRenderer>
116         <a:LabelAcquisitionRenderer bigAcquisiton="{BA_BASKET_MAKER}">
117           <a:tooltip>
118             <g:HTML>
119             <b>basket maker</b>  2x reed, 2x stone<br/>
120             in every <b>harvest season</b>:<br/>
121             <b>1x</b> 1x reed &rarr; 3x food
122             </g:HTML>
123           </a:tooltip>
124         </a:LabelAcquisitionRenderer>
125       </g:HorizontalPanel>
126     </g:VerticalPanel>
127 </ui:UiBinder>
```

Listing A105: BigAcquisitionsFieldView.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3
4  import com.google.gwt.safehtml.shared.SafeHtmlBuilder;
5  import com.google.gwt.user.client.DOM;
6  import com.google.gwt.user.client.ui.HTML;
7  import com.google.gwt.user.client.ui.Label;
8  import com.google.gwt.user.client.ui.VerticalPanel;
9
10 import de.tu_freiberg.informatik.vonwenckstern.client.model.BigFieldModel;
11 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasBaseFieldModel;
12 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
13 import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
14
15 public class BigFieldRenderer extends VerticalPanel implements
       HasBaseFieldModel, Renderer {
16   private BigFieldModel model;
17   private int order;
18
19   public int getOrder() {
20     return order;
21   }
22
23   public void setOrder(int order) {
24     this.order = order;
25   }
26
27   public BigFieldRenderer() {
28     this.setPixelSize(100, 100);
29     DOM.setStyleAttribute(this.getElement(), "border", "1px␣solid␣black");
30   }
31
32   public BigFieldModel getModel() {
33     return model;
34   }
35
36   public void setModel(BigFieldModel model) {
37     this.model = model;
38     if(model != null) {
39       if(model.getDescription() != null)
40         new Tooltip(this, new HTML(new SafeHtmlBuilder().
               appendEscapedLines(model.getDescription()).toSafeHtml()));
41       render();
42     }
43   }
44
45   public void render() {
46     this.clear();
47     String s = null;
48     switch(model.getBgCard()) {
49     case ACQUISITION: s = "get␣a␣big␣acquisition"; break;
50     case BOAR: s = "get␣boars"; break;
51     case CABARET: s = "cabaret:␣you␣can␣get␣food"; break;
52     case COW: s = "get␣cows"; break;
53     case FAMILIY_ADDITION2: s = "family␣addition␣(need␣a␣room)"; break;
54     case FAMILY_ADDITION5: s = "family␣addition␣(need␣not␣a␣room)"; break;
55     case FENCE: s = "build␣fences"; break;
56     case NONE: s = null; break;
57     case ONE_WOOD: s = "get␣wood"; break;
58     case PLOWING_SOWING: s = "plow␣field␣and/or␣seed␣grains"; break;
```

```
59      case REED_STONE_FOOD: s = "get␣one␣reed,␣one␣stone,␣and␣one␣food␣
            marker"; break;
60      case RESTAURATION: s = "restaurate␣your␣rooms"; break;
61      case RESTAURATION_FENCE: s = "restaurate␣your␣rooms␣and␣build␣fences␣
            if␣wanted"; break;
62      case SEEDING_BACKING: s = "seed␣grains␣and/or␣back␣bread"; break;
63      case SHEEP: s = "get␣sheep"; break;
64      case STONE2: s = "get␣stone"; break;
65      case STONE4: s = "get␣stone"; break;
66      case TWO_CLAY: s = "get␣clay"; break;
67      case TWO_WOOD: s = "get␣wood"; break;
68      case VEGETABLE: s = "get␣one␣vegetable"; break;
69      }
70      if(s != null) {
71        Label l = new Label(s);
72        DOM.setStyleAttribute(l.getElement(), "backgroundColor", "lightblue"
              );
73        this.add(l);
74      }
75      this.add(new ChildRenderer(model));
76      this.add(new ResourceRenderer(model));
77      this.setVisible(model.isVisible());
78    }
79  }
```

Listing A106: BigFieldRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.event.dom.client.ClickEvent;
5  import com.google.gwt.event.dom.client.ClickHandler;
6  import com.google.gwt.uibinder.client.UiBinder;
7  import com.google.gwt.user.client.ui.Composite;
8  import com.google.gwt.user.client.ui.Grid;
9  import com.google.gwt.user.client.ui.HasWidgets;
10 import com.google.gwt.user.client.ui.Widget;
11
12 import de.tu_freiberg.informatik.vonwenckstern.client.model.CardFieldModel
       ;
13 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       CardFieldPresenter;
14 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
15
16 public class CardFieldView extends Composite implements CardFieldPresenter
       .Display {
17
18   private static final Binder binder = GWT.create(Binder.class);
19   interface Binder extends UiBinder<Grid, CardFieldView> {
20   }
21   private Grid grid = null;
22
23   public CardFieldView() {
24     grid = binder.createAndBindUi(this);
25     initWidget(grid);
26   }
27
28
29   @Override
30   public void registerHandlers(ClickHandler p) {
31     for(int r=0; r<grid.getRowCount(); r++) {
```

```
32        for(int c=0; c<grid.getCellCount(r); c++) {
33          Widget w = grid.getWidget(r,c);
34          if(w instanceof Renderer) {
35            w.addDomHandler(p, ClickEvent.getType());
36          }
37        }
38      }
39    }
40
41
42    @Override
43    public HasWidgets getPanel() {
44      return grid;
45    }
46
47
48    @Override
49    public void showOneMoreBigCard(int round) {
50    }
51
52
53    @Override
54    public void update(CardFieldModel model) {
55      ((BigFieldRenderer)grid.getWidget(0, 0)).setModel(model.
            getModelOneWood());
56      ((BigFieldRenderer)grid.getWidget(0, 1)).setModel(model.
            getModelTwoClay());
57      ((BigFieldRenderer)grid.getWidget(0, 2)).setModel(model.
            getModelTwoWood());
58      ((BigFieldRenderer)grid.getWidget(1, 0)).setModel(model.
            getModelReedStoneFood());
59      ((BigFieldRenderer)grid.getWidget(1, 1)).setModel(model.
            getModelCabaret());
60      ((TooltipPanelChildRenderer)grid.getWidget(1, 2)).setModel(model.
            getModelHouse());
61      ((TooltipPanelChildRenderer)grid.getWidget(2, 0)).setModel(model.
            getModelGrain());
62      ((TooltipPanelChildRenderer)grid.getWidget(2, 1)).setModel(model.
            getModelPlowField());
63      ((TooltipPanelChildRenderer)grid.getWidget(2, 2)).setModel(model.
            getModelFood());
64    }
65
66  }
```

Listing A107: CardFieldView.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       mobile'>
5    <ui:style>
6      .panel {
7        background-color: ivory;
8        cursor: pointer;
9      }
10     .handcursor {
11       cursor: pointer;
12       }
```

```
13        .nocursor {
14        cursor: not-allowed;
15      }
16    </ui:style>
17    <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
          BackgroundCard.*" />
18    <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
          Resource.*" />
19    <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
          Child.*" />
20    <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
          .resources.Images'/>
21 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
      .BigFieldModel" field="modelOneWood"> -->
22 <!--      <ui:attributes bgCard="{ONE_WOOD}" ressource="{R_WOOD}"
      ressourceCount="1" ressourceRoundAddition="1" -->
23 <!--          description="get␣woods"/> -->
24 <!--    </ui:with> -->
25 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
      .BigFieldModel" field="modelTwoWood"> -->
26 <!--      <ui:attributes bgCard="{TWO_WOOD}" ressource="{R_WOOD}"
      ressourceCount="2" ressourceRoundAddition="2" -->
27 <!--          description="get␣woods"/> -->
28 <!--    </ui:with> -->
29 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
      .BigFieldModel" field="modelTwoClay"> -->
30 <!--      <ui:attributes bgCard="{TWO_CLAY}" ressource="{R_CLAY}"
      ressourceCount="2" ressourceRoundAddition="2" -->
31 <!--          description="get␣clays"/> -->
32 <!--    </ui:with> -->
33 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
      .BigFieldModel" field="modelReedStoneFood"> -->
34 <!--      <ui:attributes bgCard="{REED_STONE_FOOD}" ressource="{R_NONE}"
      ressourceCount="0" ressourceRoundAddition="0" -->
35 <!--          description="get␣one␣reed,␣one␣stone␣and␣one␣food␣marker"/> -->
36 <!--    </ui:with> -->
37 <!--    <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model
      .BigFieldModel" field="modelCabaret"> -->
38 <!--      <ui:attributes bgCard="{CABARET}" ressource="{R_FOOD}"
      ressourceCount="1" ressourceRoundAddition="1"  -->
39 <!--          description="get␣food␣markers"/> -->
40 <!--    </ui:with> -->
41        <g:Grid width="300px" height="400px">
42          <g:row>
43            <g:customCell>
44              <a:BigFieldRenderer styleName="{style.handcursor}"/>
45            </g:customCell>
46            <g:customCell>
47              <a:BigFieldRenderer styleName="{style.handcursor}"/>
48            </g:customCell>
49            <g:customCell>
50              <a:BigFieldRenderer styleName="{style.handcursor}"/>
51            </g:customCell>
52          </g:row>
53          <g:row>
54            <g:customCell>
55              <a:BigFieldRenderer styleName="{style.handcursor}"/>
56            </g:customCell>
57            <g:customCell>
58              <a:BigFieldRenderer styleName="{style.handcursor}" />
59            </g:customCell>
```

```
60            <g:customCell>
61              <a:TooltipPanelChildRenderer styleName="{style.handcursor}"
                    value="buildHouse" fixText="build␣a␣house␣and/or␣stables">
62                <a:tooltip>
63                  <g:HTML>
64                  <b>build a house</b><br/>
65                  wood house: 5x wood + 2x reed<br/>
66                  clay house: 5x clay + 2x reed<br/>
67                  stone house: 5x stone + 2x reed<br/>
68                  <br/><br/>and/or <b>build stables</b><br/>
69                  2 woods per stable
70                  </g:HTML>
71                </a:tooltip>
72              </a:TooltipPanelChildRenderer>
73            </g:customCell>
74          </g:row>
75          <g:row>
76            <g:customCell>
77              <a:TooltipPanelChildRenderer styleName="{style.handcursor}"
                    value="oneGrain" fixText="get␣one␣grain">
78                <a:tooltip>
79                  <g:HTML>
80                  get one grain
81                  </g:HTML>
82                </a:tooltip>
83              </a:TooltipPanelChildRenderer>
84            </g:customCell>
85            <g:customCell>
86              <a:TooltipPanelChildRenderer styleName="{style.handcursor}"
                    value="plowField" fixText="plow␣one␣field">
87                <a:tooltip>
88                  <g:HTML>
89                  plow one field
90                  </g:HTML>
91                </a:tooltip>
92              </a:TooltipPanelChildRenderer>
93            </g:customCell>
94            <g:customCell>
95              <a:TooltipPanelChildRenderer styleName="{style.handcursor}"
                    value="twoFoodMarkers" fixText="get␣two␣food␣markers">
96                <a:tooltip>
97                  <g:HTML>
98                  get two food marers
99                  </g:HTML>
100               </a:tooltip>
101             </a:TooltipPanelChildRenderer>
102           </g:customCell>
103         </g:row>
104       </g:Grid>
105 </ui:UiBinder>
```

Listing A108: CardFieldView.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2  import com.google.gwt.user.client.ui.HTML;
3
4  import de.tu_freiberg.informatik.vonwenckstern.client.model.BaseFieldModel
       ;
5
6  public class ChildRenderer extends HTML {
7    private BaseFieldModel model;
8    public ChildRenderer() {
9    }
10
11   public ChildRenderer(BaseFieldModel model) {
12     this();
13     setModel(model);
14   }
15
16   public void setModel(BaseFieldModel model) {
17     this.model = model;
18     if(model != null)
19       render();
20   }
21
22   public void render() {
23     String s = null;
24     switch(model.getChild()) {
25     case C_BLUE: s = "blue"; break;
26     case C_GREEN: s = "green"; break;
27     case C_NONE: s = null; break;
28     case C_RED: s = "red"; break;
29     case C_ROSA: s = "pink"; break;
30     }
31     if(s != null) {
32       setHTML("<font␣color=\"red\">occupied␣by␣" + s + "</font>");
33     } else {
34       setHTML("<font␣color=\"green\">free</font>");
35     }
36   }
37 }
```

Listing A109: ChildRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.uibinder.client.UiBinder;
5  import com.google.gwt.uibinder.client.UiField;
6  import com.google.gwt.user.client.ui.Composite;
7  import com.google.gwt.user.client.ui.HTML;
8  import com.google.gwt.user.client.ui.HorizontalPanel;
9  import com.google.gwt.user.client.ui.IntegerBox;
10 import com.google.gwt.user.client.ui.Label;
11 import com.google.gwt.user.client.ui.Widget;
12
13 import de.tu_freiberg.informatik.vonwenckstern.client.model.BigFieldModel;
14 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       PlayerResourceModel;
15 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       InfoViewPresenter;
16 import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
17 public class InfoView extends Composite implements InfoViewPresenter.
       Display {
```

```
18
19  private static final Binder binder = GWT.create(Binder.class);
20
21  interface Binder extends UiBinder<HorizontalPanel, InfoView> {
22  }
23
24  private HorizontalPanel panel = null;
25
26  public InfoView() {
27    panel = binder.createAndBindUi(this);
28    initWidget(panel);
29    new Tooltip(beggarCard, new HTML("This is a beggar card.<br>You
          receive this card, if you have not enough food family.<br><br>
          Every cards gives you minus <b>3</b> points at the end."));
30    new Tooltip(childCard, new HTML("This are your free persons, which you
          can add to your family by playing the card <b>making a baby</b>."
          ));
31    String sStable = "Your available stables, which you can add to your
          fields by playing <b>build a house</b> and/or <b>build stables</b
          >.";
32    new Tooltip(stableCard, new HTML(sStable));
33    new Tooltip(stableCounter, new HTML(sStable));
34    String sFence = "Your available fence parts, which you need to fence
          your fields.";
35    new Tooltip(fenceCard, new HTML(sFence));
36    new Tooltip(fenceCounter, new HTML(sFence));
37  }
38
39  @UiField
40  BigFieldModel modelWood;
41  @UiField
42  BigFieldModel modelClay;
43  @UiField
44  BigFieldModel modelStone;
45  @UiField
46  BigFieldModel modelReed;
47  @UiField
48  BigFieldModel modelFood;
49  @UiField
50  BigFieldModel modelGrain;
51  @UiField
52  BigFieldModel modelVegetable;
53  @UiField
54  Label beggarCard;
55  @UiField
56  IntegerBox beggarCounter;
57  @UiField
58  Label childCard;
59  @UiField
60  IntegerBox childCounter;
61  @UiField
62  Label stableCard;
63  @UiField
64  IntegerBox stableCounter;
65  @UiField
66  Label fenceCard;
67  @UiField
68  IntegerBox fenceCounter;
69
70
71  @Override
```

```
72    public void updateView(PlayerResourceModel model) {
73      modelWood.setRessourceCount(model.getWoodCount());
74      modelClay.setRessourceCount(model.getClayCount());
75      modelStone.setRessourceCount(model.getStoneCount());
76      modelReed.setRessourceCount(model.getReedCount());
77      modelFood.setRessourceCount(model.getFoodCount());
78      modelGrain.setRessourceCount(model.getGrainCount());
79      modelVegetable.setRessourceCount(model.getVegetableCount());
80
81      beggarCard.setVisible(model.getBeggerCards() > 0);
82      beggarCounter.setVisible(model.getBeggerCards() > 1);
83      beggarCounter.setValue(model.getBeggerCards());
84
85      childCard.setVisible(model.getPersonsCount() > 0);
86      childCounter.setVisible(model.getPersonsCount() > 1);
87      childCounter.setValue(model.getPersonsCount());
88
89      stableCard.setVisible(model.getStableCount() > 0);
90      stableCounter.setVisible(model.getStableCount() > 1);
91      stableCounter.setValue(model.getStableCount());
92
93      fenceCard.setVisible(model.getFenceCount() > 0);
94      fenceCounter.setVisible(model.getFenceCount() > 1);
95      fenceCounter.setValue(model.getFenceCount());
96
97      for(int i=0; i<panel.getWidgetCount(); i++) {
98        Widget w = panel.getWidget(i);
99        if(w instanceof ResourceRenderer) {
100         ((ResourceRenderer) w).render();
101       }
102     }
103   }
104 }
```

Listing A110: InfoView.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       mobile'>
5    <ui:style>
6      .handcursor {
7        cursor: pointer;
8        padding: 20px;
9        }
10       .space {
11       padding: 10px;
12     }
13   </ui:style>
14   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BackgroundCard.*" />
15   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         Resource.*" />
16   <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
         .resources.Images'/>
17   <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BigFieldModel" field="modelWood">
18     <ui:attributes ressource="{R_WOOD}" ressourceCount="0"
           ressourceRoundAddition="0" description="your␣wood␣resources"/>
```

```
19  </ui:with>
20  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        BigFieldModel" field="modelClay">
21    <ui:attributes ressource="{R_CLAY}" ressourceCount="0"
          ressourceRoundAddition="0" description="your␣clay␣resources"/>
22  </ui:with>
23  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        BigFieldModel" field="modelStone">
24    <ui:attributes ressource="{R_STONE}" ressourceCount="0"
          ressourceRoundAddition="0" description="your␣stone␣resources"/>
25  </ui:with>
26  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        BigFieldModel" field="modelReed">
27    <ui:attributes ressource="{R_REED}" ressourceCount="0"
          ressourceRoundAddition="0" description="your␣reed␣resources"/>
28  </ui:with>
29  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        BigFieldModel" field="modelFood">
30    <ui:attributes ressource="{R_FOOD}" ressourceCount="3"
          ressourceRoundAddition="0" description="your␣food␣markers"/>
31  </ui:with>
32  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        BigFieldModel" field="modelGrain">
33    <ui:attributes ressource="{R_GRAIN}" ressourceCount="0"
          ressourceRoundAddition="0" description="your␣grain␣resources"/>
34  </ui:with>
35  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        BigFieldModel" field="modelVegetable">
36    <ui:attributes ressource="{R_VEGETABLE}" ressourceCount="0"
          ressourceRoundAddition="0" description="your␣vegetable␣resources"
          />
37  </ui:with>
38
39  <g:HorizontalPanel>
40    <a:ResourceRenderer model="{modelWood}" styleName="{style.handcursor}"
          />
41    <a:ResourceRenderer model="{modelClay}" styleName="{style.handcursor}"
          />
42    <a:ResourceRenderer model="{modelStone}" styleName="{style.handcursor}
          "/>
43    <a:ResourceRenderer model="{modelReed}" styleName="{style.handcursor}"
          />
44    <a:ResourceRenderer model="{modelFood}" styleName="{style.handcursor}"
          />
45    <a:ResourceRenderer model="{modelGrain}" styleName="{style.handcursor}
          "/>
46    <a:ResourceRenderer model="{modelVegetable}" styleName="{style.
          handcursor}"/>
47    <g:HorizontalPanel width="50px" height="50px" styleName="{style.space}
          ">
48      <g:Label text="persons" width="50px" ui:field="childCard"/>
49      <g:IntegerBox value="3" visibleLength="1" readOnly="true" width="10
            px" ui:field="childCounter"/>
50    </g:HorizontalPanel>
51    <g:HorizontalPanel width="30px" height="50px" styleName="{style.space}
          ">
52      <g:Label text="stables" height="20px" ui:field="stableCard"/>
53      <g:IntegerBox value="4" visibleLength="1" readOnly="true" width="15
            px" ui:field="stableCounter"/>
54    </g:HorizontalPanel>
```

```
55      <g:HorizontalPanel width="30px" height="50px" styleName="{style.space}
            ">
56        <g:HTML ui:field="fenceCard">
57          <div style="width:␣10px;␣height:␣50px;␣border-left:␣blue␣solid␣5px
                "></div>
58        </g:HTML>
59        <g:IntegerBox value="15" visibleLength="1" readOnly="true" width="15
              px" ui:field="fenceCounter"/>
60      </g:HorizontalPanel>
61      <g:HorizontalPanel width="100px" height="300px" styleName="{style.
            space}">
62        <g:Label text="beggar␣cards" width="100px" ui:field="beggarCard"
              visible="false"/>
63        <g:IntegerBox value="20" visibleLength="1" readOnly="true" width="15
              px" ui:field="beggarCounter" visible="false"/>
64      </g:HorizontalPanel>
65    </g:HorizontalPanel>
66  </ui:UiBinder>
```

Listing A111: InfoView.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2  import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
3  import com.google.gwt.uibinder.client.UiChild;
4  import com.google.gwt.user.client.DOM;
5  import com.google.gwt.user.client.ui.HasHTML;
6  import com.google.gwt.user.client.ui.Label;
7  import com.google.gwt.user.client.ui.Widget;
8
9  import de.tu_freiberg.informatik.vonwenckstern.client.model.
       AcquisitionCardModel;
10 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       BigAcquisitions;
11 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasAcquisitionCardModel;
12
13 public class LabelAcquisitionRenderer extends TooltipPanel implements
       Renderer, HasAcquisitionCardModel {
14
15   private AcquisitionCardModel model = new AcquisitionCardModel();
16
17   public LabelAcquisitionRenderer() {
18     super();
19     DOM.setStyleAttribute(this.getElement(), "cursor", "pointer");
20     DOM.setStyleAttribute(this.getElement(), "border", "1px␣solid␣black");
21     DOM.setStyleAttribute(this.getElement(), "textAlign", "center");
22   }
23
24   public LabelAcquisitionRenderer(AcquisitionCardModel model) {
25     this();
26     if(model != null) {
27       this.model = model;
28       render();
29     }
30   }
31
32   @UiChild(limit=1)
33   public void addTooltip(Widget tooltip) {
34     if(tooltip instanceof HasHTML) {
35       model.setDescription(((HasHTML) tooltip).getHTML());
36     }
```

```
37      super.addTooltip(tooltip);
38    }
39
40    @Override
41    public AcquisitionCardModel getModel() {
42      return model;
43    }
44
45    public void setModel(AcquisitionCardModel model) {
46      if(model != null) {
47        this.model = model;
48        render();
49      }
50    }
51
52    public void setBigAcquisiton(BigAcquisitions ba) {
53      model.setAcquisition(ba);
54      render();
55    }
56
57    @Override
58    public void render() {
59      this.clear();
60      String s = null;
61      switch(model.getAcquisition()) {
62      case BA_FIRE_PLACE: s = "fire place"; break;
63      case BA_FIRE_PLACE2: s = "fire place 2"; break;
64      case BA_COOKERY: s = "cookery"; break;
65      case BA_COOKERY2: s = "cookery 2"; break;
66      case BA_FOUNTAIN: s = "fountain"; break;
67
68      case BA_CLAY_OVEN: s = "clay oven"; break;
69      case BA_STONE_OVEN: s = "stone oven"; break;
70      case BA_JOINERY: s = "joinery"; break;
71      case BA_POTTERY: s = "pottery"; break;
72      case BA_BASKET_MAKER: s = "basket maker"; break;
73      case BA_NONE: s = ""; break;
74      }
75
76      this.add(new Label(s));
77      this.setVisible(model.isVisible());
78    }
79
80  }
```

Listing A112: LabelAcquisitionRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import java.util.Iterator;
4
5  import com.google.gwt.core.client.GWT;
6  import com.google.gwt.event.dom.client.ClickEvent;
7  import com.google.gwt.event.dom.client.ClickHandler;
8  import com.google.gwt.uibinder.client.UiBinder;
9  import com.google.gwt.uibinder.client.UiField;
10 import com.google.gwt.user.client.ui.Composite;
11 import com.google.gwt.user.client.ui.HasWidgets;
12 import com.google.gwt.user.client.ui.Label;
13 import com.google.gwt.user.client.ui.PushButton;
14 import com.google.gwt.user.client.ui.VerticalPanel;
```

```
15 import com.google.gwt.user.client.ui.Widget;
16
17 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       PlayerFieldModel;
18 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       PlayerFieldPresenter.Display;
19 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
20 import de.tu_freiberg.informatik.vonwenckstern.client.view.mobile.
       AcquisitionCardRenderer;
21 import de.tu_freiberg.informatik.vonwenckstern.client.view.mobile.
       SmallFieldRenderer;
22
23 public class PlayerFieldView extends Composite implements Display {
24
25   private static final Binder binder = GWT.create(Binder.class);
26
27   interface Binder extends UiBinder<VerticalPanel, PlayerFieldView> {
28   }
29
30   private VerticalPanel panel = null;
31
32   @UiField
33   Label info;
34   @UiField
35   PushButton btnEnclosure;
36   @UiField
37   PushButton btnFeedingFamily;
38
39   public PlayerFieldView() {
40     panel = binder.createAndBindUi(this);
41     initWidget(panel);
42   }
43
44   @Override
45   public void update(PlayerFieldModel model) {
46     _update(model, panel);
47   }
48
49   private void _update(PlayerFieldModel model, HasWidgets cont) {
50     Iterator<Widget> it = cont.iterator();
51     while(it.hasNext()) {
52       Widget w = it.next();
53       if(w instanceof SmallFieldRenderer) {
54         SmallFieldRenderer s = (SmallFieldRenderer) w;
55         s.setModel(model.getField(s.getPosition()));
56       } else if(w instanceof AcquisitionCardRenderer) {
57         AcquisitionCardRenderer a = (AcquisitionCardRenderer)w;
58         a.setModel(model.getAcquisitation(a.getPosition()));
59       } else if(w instanceof HasWidgets) {
60         _update(model, (HasWidgets) w);
61       }
62     }
63   }
64
65   @Override
66   public void registerHandlers(ClickHandler p) {
67     _registerHandlers(p, panel);
68     btnEnclosure.addClickHandler(p);
69     btnFeedingFamily.addClickHandler(p);
70   }
71
```

```
72    private void _registerHandlers(ClickHandler p, HasWidgets cont) {
73      Iterator<Widget> it = cont.iterator();
74      while(it.hasNext()) {
75        Widget w = it.next();
76        if(w instanceof Renderer) {
77          w.addDomHandler(p, ClickEvent.getType());
78        } else if(w instanceof HasWidgets) {
79          _registerHandlers(p, (HasWidgets) w);
80        }
81      }
82    }
83
84    @Override
85    /** sets the text and makes the information visible */
86    public void setInformation(String text) {
87      info.setText(text);
88      info.setVisible(true);
89    }
90
91    @Override
92    public void setInformationVisible(boolean visible) {
93      info.setVisible(visible);
94    }
95
96    @Override
97    public void setEnclosureBtnVisible(boolean visible) {
98      btnEnclosure.setVisible(visible);
99    }
100
101   @Override
102   public void setFeedingFamilyBtnVisible(boolean visible) {
103     btnFeedingFamily.setVisible(visible);
104   }
105
106 }
```

Listing A113: PlayerFieldView.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       mobile'>
5    <ui:style>
6      .panel {
7        background-color: blue;
8        cursor: pointer;
9      }
10     .handcursor {
11       cursor: pointer;
12       }
13   </ui:style>
14   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         FieldCard.*" />
15   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         Resource.*" />
16   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         Player.*" />
17   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         Child.*" />
```

```
18  <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
        .resources.Images'/>
19  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field1">
20    <ui:attributes field="{F_NONE}" player="{BLUE}" />
21  </ui:with>
22  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field2">
23    <ui:attributes field="{F_NONE}" player="{BLUE}" />
24  </ui:with>
25  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field3">
26    <ui:attributes field="{F_NONE}" player="{BLUE}" />
27  </ui:with>
28  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field4">
29    <ui:attributes field="{F_NONE}" player="{BLUE}" />
30  </ui:with>
31  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field5">
32    <ui:attributes field="{F_WOOD_HOUSE}" child="{C_BLUE}" player="{BLUE}"
          personsCount="1"/>
33  </ui:with>
34  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field6">
35    <ui:attributes field="{F_NONE}" player="{BLUE}" />
36  </ui:with>
37  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field7">
38    <ui:attributes field="{F_NONE}" player="{BLUE}" />
39  </ui:with>
40  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field8">
41    <ui:attributes field="{F_NONE}" player="{BLUE}" />
42  </ui:with>
43  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field9">
44    <ui:attributes field="{F_NONE}" player="{BLUE}" />
45  </ui:with>
46  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field10">
47    <ui:attributes field="{F_WOOD_HOUSE}" child="{C_BLUE}" player="{BLUE}"
          personsCount="1"/>
48  </ui:with>
49  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field11">
50    <ui:attributes field="{F_NONE}" player="{BLUE}" />
51  </ui:with>
52  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field12">
53    <ui:attributes field="{F_NONE}" player="{BLUE}" />
54  </ui:with>
55  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field13">
56    <ui:attributes field="{F_NONE}" player="{BLUE}" />
57  </ui:with>
58  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field14">
59    <ui:attributes field="{F_NONE}" player="{BLUE}" />
60  </ui:with>
```

```
61  <ui:with type="de.tu_freiberg.informatik.vonwenckstern.client.model.
        SmallFieldModel" field="field15">
62    <ui:attributes field="{F_NONE}" player="{BLUE}" />
63  </ui:with>
64
65  <g:VerticalPanel width="400px" height="300px">
66    <g:Grid cellSpacing="5">
67      <g:row>
68        <g:customCell>
69          <a:SmallFieldRenderer model="{field1}" styleName="{style.
                handcursor}" position="0"/>
70        </g:customCell>
71        <g:customCell>
72          <a:SmallFieldRenderer model="{field2}" styleName="{style.
                handcursor}" position="1"/>
73        </g:customCell>
74        <g:customCell>
75          <a:SmallFieldRenderer model="{field3}" styleName="{style.
                handcursor}" position="2"/>
76        </g:customCell>
77        <g:customCell>
78          <a:SmallFieldRenderer model="{field4}" styleName="{style.
                handcursor}" position="3"/>
79        </g:customCell>
80        <g:customCell>
81          <a:SmallFieldRenderer model="{field5}" styleName="{style.
                handcursor}" position="4"/>
82        </g:customCell>
83      </g:row>
84      <g:row>
85        <g:customCell>
86          <a:SmallFieldRenderer model="{field6}" styleName="{style.
                handcursor}" position="5"/>
87        </g:customCell>
88        <g:customCell>
89          <a:SmallFieldRenderer model="{field7}" styleName="{style.
                handcursor}" position="6"/>
90        </g:customCell>
91        <g:customCell>
92          <a:SmallFieldRenderer model="{field8}" styleName="{style.
                handcursor}" position="7"/>
93        </g:customCell>
94        <g:customCell>
95          <a:SmallFieldRenderer model="{field9}" styleName="{style.
                handcursor}" position="8"/>
96        </g:customCell>
97        <g:customCell>
98          <a:SmallFieldRenderer model="{field10}" styleName="{style.
                handcursor}" position="9"/>
99        </g:customCell>
100     </g:row>
101     <g:row>
102       <g:customCell>
103         <a:SmallFieldRenderer model="{field11}" styleName="{style.
                handcursor}" position="10"/>
104       </g:customCell>
105       <g:customCell>
106         <a:SmallFieldRenderer model="{field12}" styleName="{style.
                handcursor}" position="11"/>
107       </g:customCell>
108       <g:customCell>
```

```
109           <a:SmallFieldRenderer model="{field13}" styleName="{style.
                  handcursor}" position="12"/>
110         </g:customCell>
111         <g:customCell>
112           <a:SmallFieldRenderer model="{field14}" styleName="{style.
                  handcursor}" position="13"/>
113         </g:customCell>
114         <g:customCell>
115           <a:SmallFieldRenderer model="{field15}" styleName="{style.
                  handcursor}" position="14"/>
116         </g:customCell>
117       </g:row>
118     </g:Grid>
119     <g:HorizontalPanel>
120       <g:Label text="Information" ui:field="info" visible="false"/>
121       <g:PushButton text="new␣enclosure" ui:field="btnEnclosure" visible="
              false"/>
122       <g:PushButton text="finish␣special␣events␣and␣continue␣feeding␣your␣
              family" ui:field="btnFeedingFamily" visible="false"/>
123     </g:HorizontalPanel>
124     <g:Grid>
125       <g:row>
126         <g:customCell>
127           <a:AcquisitionCardRenderer position="4"/>
128         </g:customCell>
129         <g:customCell>
130           <a:AcquisitionCardRenderer position="3"/>
131         </g:customCell>
132         <g:customCell>
133           <a:AcquisitionCardRenderer position="2"/>
134         </g:customCell>
135         <g:customCell>
136           <a:AcquisitionCardRenderer position="1"/>
137         </g:customCell>
138         <g:customCell>
139           <a:AcquisitionCardRenderer position="0"/>
140         </g:customCell>
141       </g:row>
142       <g:row>
143         <g:customCell>
144           <a:AcquisitionCardRenderer position="9"/>
145         </g:customCell>
146         <g:customCell>
147           <a:AcquisitionCardRenderer position="8"/>
148         </g:customCell>
149         <g:customCell>
150           <a:AcquisitionCardRenderer position="7"/>
151         </g:customCell>
152         <g:customCell>
153           <a:AcquisitionCardRenderer position="6"/>
154         </g:customCell>
155         <g:customCell>
156           <a:AcquisitionCardRenderer position="5"/>
157         </g:customCell>
158       </g:row>
159     </g:Grid>
160   </g:VerticalPanel>
161 </ui:UiBinder>
```

Listing A114: PlayerFieldView.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import com.google.gwt.safehtml.shared.SafeHtmlBuilder;
4  import com.google.gwt.user.client.DOM;
5  import com.google.gwt.user.client.ui.HTML;
6  import com.google.gwt.user.client.ui.VerticalPanel;
7
8  import de.tu_freiberg.informatik.vonwenckstern.client.model.BigFieldModel;
9  import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasBaseFieldModel;
10 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
11 import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
12 public class ResChildRenderer extends VerticalPanel implements Renderer,
       HasBaseFieldModel {
13   private BigFieldModel model;
14
15   public ResChildRenderer() {
16     this.setPixelSize(100, 50);
17     DOM.setStyleAttribute(this.getElement(), "border", "1px␣solid␣black");
18   }
19
20   public BigFieldModel getModel() {
21     return model;
22   }
23
24   public void setModel(BigFieldModel model) {
25     this.model = model;
26     if(model != null) {
27       if(model.getDescription() != null)
28         new Tooltip(this, new HTML(new SafeHtmlBuilder().
               appendEscapedLines(model.getDescription()).toSafeHtml()));
29       render();
30     }
31   }
32
33   @Override
34   public void render() {
35     this.clear();
36     this.add(new ChildRenderer(model));
37     this.add(new ResourceRenderer(model));
38   }
39
40 }
```

Listing A115: ResChildRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import com.google.gwt.safehtml.shared.SafeHtmlBuilder;
4  import com.google.gwt.user.client.ui.HTML;
5  import com.google.gwt.user.client.ui.HorizontalPanel;
6  import com.google.gwt.user.client.ui.IntegerBox;
7  import com.google.gwt.user.client.ui.Label;
8
9  import de.tu_freiberg.informatik.vonwenckstern.client.model.BaseFieldModel
       ;
10 import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
11
12 public class ResourceRenderer extends HorizontalPanel {
13   private BaseFieldModel model;
14   public ResourceRenderer() {
```

```
15    }
16
17    public ResourceRenderer(BaseFieldModel model) {
18      this();
19      setModel(model);
20    }
21
22    public BaseFieldModel getModel() {
23      return model;
24    }
25
26    public void setModel(BaseFieldModel model) {
27      this.model = model;
28      if(model != null) {
29        if(model.getDescription() != null)
30          new Tooltip(this, new HTML(new SafeHtmlBuilder().
                appendEscapedLines(model.getDescription()).toSafeHtml()));
31        render();
32      }
33    }
34
35    public void render() {
36      this.clear();
37      String s = null;
38      switch(model.getRessource()) {
39      case R_BOAR: s = "boar"; break;
40      case R_CLAY: s = "clay"; break;
41      case R_COW: s = "cow"; break;
42      case R_FOOD: s = "food"; break;
43      case R_GRAIN: s = "grain"; break;
44      case R_NONE: s = null; break;
45      case R_REED: s = "reed"; break;
46      case R_SHEEP: s = "sheep"; break;
47      case R_STONE: s = "stone"; break;
48      case R_VEGETABLE: s = "vegetable"; break;
49      case R_WOOD: s = "wood"; break;
50      }
51      if((s != null) && (model.getRessourceCount() > 0)) {
52        this.add(new Label(s));
53        if(model.getRessourceCount() > 1) {
54          IntegerBox tb = new IntegerBox();
55          tb.setVisibleLength(2);
56          tb.setValue(model.getRessourceCount());
57          tb.setReadOnly(true);
58          Label l = new Label();
59          l.setPixelSize(5, 5);
60          this.add(l);
61          this.add(tb);
62        }
63      }
64      this.setVisible((s != null) && (model.getRessourceCount() > 0));
65    }
66  }
```

Listing A116: ResourceRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.event.dom.client.ClickEvent;
5  import com.google.gwt.event.dom.client.ClickHandler;
6  import com.google.gwt.uibinder.client.UiBinder;
7  import com.google.gwt.user.client.ui.Composite;
8  import com.google.gwt.user.client.ui.Grid;
9  import com.google.gwt.user.client.ui.HasWidgets;
10 import com.google.gwt.user.client.ui.VerticalPanel;
11 import com.google.gwt.user.client.ui.Widget;
12
13 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       Rounds1To7Model;
14 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       Rounds1To7Presenter;
15 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
16 public class Rounds1To7View extends Composite implements
       Rounds1To7Presenter.Display {
17
18   private static final Binder binder = GWT.create(Binder.class);
19
20   interface Binder extends UiBinder<Grid, Rounds1To7View> {
21   }
22
23   private Grid grid = null;
24
25   public Rounds1To7View() {
26     grid = binder.createAndBindUi(this);
27     initWidget(grid);
28   }
29
30   @Override
31   public void registerHandlers(ClickHandler p) {
32     for(int r=0; r<grid.getRowCount(); r++) {
33       for(int c=0; c<grid.getCellCount(r); c++) {
34         Widget w = grid.getWidget(r,c);
35         if(w instanceof Renderer) {
36           w.addDomHandler(p, ClickEvent.getType());
37         } else if(w instanceof VerticalPanel) {
38           VerticalPanel vp = (VerticalPanel)w;
39           vp.getWidget(0).addDomHandler(p, ClickEvent.getType());
40           vp.getWidget(1).addDomHandler(p, ClickEvent.getType());
41         }
42       }
43     }
44   }
45
46   @Override
47   public void showOneMoreBigCard(int round) {
48     for(int r=0; r<grid.getRowCount(); r++) {
49       for(int c=0; c<grid.getCellCount(r); c++) {
50         Widget w = grid.getWidget(r,c);
51         if(w instanceof BigFieldRenderer && ((BigFieldRenderer)w).getOrder
               () == round) {
52           w.setVisible(true);
53           ((BigFieldRenderer)w).getModel().setVisible(true);
54         }
55       }
56     }
57   }
```

```
58
59    @Override
60    public HasWidgets getPanel() {
61      return grid;
62    }
63
64    @Override
65    public void update(Rounds1To7Model model) {
66      ((BigFieldRenderer)grid.getWidget(0, 0)).setModel(model.getModelSheep
          ());
67      ((BigFieldRenderer)grid.getWidget(0, 1)).setModel(model.
          getModelBigAcquisitation());
68      ((BigFieldRenderer)grid.getWidget(0, 2)).setModel(model.getModelFamily
          ());
69      ((ResChildRenderer)((VerticalPanel)grid.getWidget(1, 0)).getWidget(0))
          .setModel(model.getModelWood());
70      ((ResChildRenderer)((VerticalPanel)grid.getWidget(1, 0)).getWidget(1))
          .setModel(model.getModelClay());
71      ((BigFieldRenderer)grid.getWidget(1, 1)).setModel(model.getModelFence
          ());
72      ((BigFieldRenderer)grid.getWidget(1, 2)).setModel(model.getModelStone
          ());
73      ((ResChildRenderer)((VerticalPanel)grid.getWidget(2, 0)).getWidget(0))
          .setModel(model.getModelReed());
74      ((ResChildRenderer)((VerticalPanel)grid.getWidget(2, 0)).getWidget(1))
          .setModel(model.getModelFood());
75      ((BigFieldRenderer)grid.getWidget(2, 1)).setModel(model.
          getModelSeedingBacking());
76      ((BigFieldRenderer)grid.getWidget(2, 2)).setModel(model.
          getModelRestauration());
77    }
78
79  }
```

Listing A117: Rounds1To7View.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       mobile'>
5    <ui:style>
6      .panel {
7        background-color: ivory;
8        cursor: pointer;
9      }
10     .handcursor {
11       cursor: pointer;
12       }
13   </ui:style>
14   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BackgroundCard.*" />
15   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         Resource.*" />
16   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         Child.*" />
17   <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
         .resources.Images'/>
18   <g:Grid width="300px" height="400px">
19     <g:row>
```

```
20        <g:customCell>
21          <a:BigFieldRenderer styleName="{style.handcursor}" order="1"/>
22        </g:customCell>
23        <g:customCell>
24          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                order="2"/>
25        </g:customCell>
26        <g:customCell>
27          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                order="5"/>
28        </g:customCell>
29      </g:row>
30      <g:row>
31        <g:customCell>
32          <g:VerticalPanel>
33            <a:ResChildRenderer styleName="{style.handcursor}"/>
34            <a:ResChildRenderer styleName="{style.handcursor}"/>
35          </g:VerticalPanel>
36        </g:customCell>
37        <g:customCell>
38          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                order="3"/>
39        </g:customCell>
40        <g:customCell>
41          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                order="6"/>
42        </g:customCell>
43      </g:row>
44      <g:row>
45        <g:customCell>
46          <g:VerticalPanel>
47            <a:ResChildRenderer styleName="{style.handcursor}"/>
48            <a:ResChildRenderer styleName="{style.handcursor}"/>
49          </g:VerticalPanel>
50        </g:customCell>
51        <g:customCell>
52          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                order="4"/>
53        </g:customCell>
54        <g:customCell>
55          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                order="7"/>
56        </g:customCell>
57      </g:row>
58      <g:row>
59        <g:cell></g:cell>
60        <g:cell>harvest season</g:cell>
61        <g:cell>harvest season</g:cell>
62      </g:row>
63    </g:Grid>
64  </ui:UiBinder>
```

Listing A118: Rounds1To7View.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.event.dom.client.ClickEvent;
5  import com.google.gwt.event.dom.client.ClickHandler;
6  import com.google.gwt.uibinder.client.UiBinder;
7  import com.google.gwt.user.client.ui.Composite;
8  import com.google.gwt.user.client.ui.Grid;
9  import com.google.gwt.user.client.ui.HasWidgets;
10 import com.google.gwt.user.client.ui.Widget;
11
12 import de.tu_freiberg.informatik.vonwenckstern.client.model.
       Rounds8To14Model;
13 import de.tu_freiberg.informatik.vonwenckstern.client.presenter.
       Rounds8To14Presenter;
14 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
15 public class Rounds8To14View extends Composite implements
       Rounds8To14Presenter.Display {
16
17   private static final Binder binder = GWT.create(Binder.class);
18
19   interface Binder extends UiBinder<Grid, Rounds8To14View> {
20   }
21
22   private Grid grid = null;
23
24   public Rounds8To14View() {
25     grid = binder.createAndBindUi(this);
26     initWidget(grid);
27   }
28
29   @Override
30   public void registerHandlers(ClickHandler p) {
31     for(int r=0; r<grid.getRowCount(); r++) {
32       for(int c=0; c<grid.getCellCount(r); c++) {
33         Widget w = grid.getWidget(r,c);
34         if(w instanceof Renderer) {
35           w.addDomHandler(p, ClickEvent.getType());
36         }
37       }
38     }
39   }
40
41   @Override
42   public void showOneMoreBigCard(int round) {
43     for(int r=0; r<grid.getRowCount(); r++) {
44       for(int c=0; c<grid.getCellCount(r); c++) {
45         Widget w = grid.getWidget(r,c);
46         if(w instanceof BigFieldRenderer && ((BigFieldRenderer)w).getOrder
               () == round) {
47           w.setVisible(true);
48           ((BigFieldRenderer)w).getModel().setVisible(true);
49         }
50       }
51     }
52   }
53
54   @Override
55   public HasWidgets getPanel() {
56     return grid;
57   }
```

```
58
59    @Override
60    public void update(Rounds8To14Model model) {
61      ((BigFieldRenderer)grid.getWidget(0, 0)).setModel(model.getModelBoar()
          );
62      ((BigFieldRenderer)grid.getWidget(0, 1)).setModel(model.getModelStone
          ());
63      ((BigFieldRenderer)grid.getWidget(0, 2)).setModel(model.
          getModelPlowSow());
64      ((BigFieldRenderer)grid.getWidget(0, 3)).setModel(model.
          getModelRestauration());
65      ((BigFieldRenderer)grid.getWidget(1, 0)).setModel(model.
          getModelVegetable());
66      ((BigFieldRenderer)grid.getWidget(1, 1)).setModel(model.getModelCow())
          ;
67      ((BigFieldRenderer)grid.getWidget(1, 2)).setModel(model.getModelFamily
          ());
68    }
69
70  }
```

Listing A119: Rounds8To14View.java file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
3  <ui:UiBinder xmlns:ui='urn:ui:com.google.gwt.uibinder' xmlns:g='urn:import
       :com.google.gwt.user.client.ui'
4  xmlns:a='urn:import:de.tu_freiberg.informatik.vonwenckstern.client.view.
       mobile'>
5    <ui:style>
6      .panel {
7        background-color: ivory;
8        cursor: pointer;
9      }
10     .handcursor {
11       cursor: pointer;
12       }
13   </ui:style>
14   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         BackgroundCard.*" />
15   <ui:import field="de.tu_freiberg.informatik.vonwenckstern.client.model.
         Resource.*" />
16   <ui:with field='im' type='de.tu_freiberg.informatik.vonwenckstern.client
         .resources.Images'/>
17   <g:Grid width="400px" height="300px">
18     <g:row>
19       <g:customCell>
20         <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
               order="8"/>
21       </g:customCell>
22       <g:customCell>
23         <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
               order="10"/>
24       </g:customCell>
25       <g:customCell>
26         <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
               order="12"/>
27       </g:customCell>
28       <g:customCell>
29         <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
               order="14"/>
```

```
30        </g:customCell>
31      </g:row>
32      <g:row>
33        <g:customCell>
34          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                order="9"/>
35        </g:customCell>
36        <g:customCell>
37          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                order="11"/>
38        </g:customCell>
39        <g:customCell>
40          <a:BigFieldRenderer styleName="{style.handcursor}" visible="false"
                order="13"/>
41        </g:customCell>
42        <g:cell>
43          harvest season
44        </g:cell>
45      </g:row>
46      <g:row>
47        <g:cell>
48          harvest season
49        </g:cell>
50        <g:cell>
51          harvest season
52        </g:cell>
53        <g:cell>
54          harvest season
55        </g:cell>
56        <g:cell></g:cell>
57      </g:row>
58    </g:Grid>
59  </ui:UiBinder>
```

Listing A120: Rounds8To14View.ui.xml file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import com.google.gwt.user.client.DOM;
4  import com.google.gwt.user.client.ui.DoubleBox;
5  import com.google.gwt.user.client.ui.Label;
6  import com.google.gwt.user.client.ui.VerticalPanel;
7
8  import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasBaseFieldModel;
9  import de.tu_freiberg.informatik.vonwenckstern.client.model.
       SmallFieldModel;
10 import de.tu_freiberg.informatik.vonwenckstern.client.view.HasPosition;
11 import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
12 public class SmallFieldRenderer extends VerticalPanel implements Renderer,
       HasBaseFieldModel, HasPosition {
13   private SmallFieldModel model;
14   private int position;
15   public int getPosition() {
16     return position;
17   }
18
19   public void setPosition(int position) {
20     this.position = position;
21   }
22
```

```
23  public SmallFieldRenderer() {
24    this.setPixelSize(70, 70);
25    DOM.setStyleAttribute(this.getElement(), "border", "1px solid black");
26  }
27
28  public SmallFieldRenderer(SmallFieldModel model) {
29    this();
30    setModel(model);
31  }
32
33  public void setModel(SmallFieldModel model) {
34    this.model = model;
35    if(model != null)
36      render();
37  }
38
39  public SmallFieldModel getModel() {
40    return model;
41  }
42
43  public void render() {
44    this.clear();
45    String field = null;
46    switch(model.getField()) {
47    case F_CLAY_HOUSE: field = "clay house"; break;
48    case F_FIELD: field = "field"; break;
49    case F_NONE: field = null; break;
50    case F_STABLE: field = "stable"; break;
51    case F_STONE_HOUSE: field = "stone house"; break;
52    case F_WOOD_HOUSE: field = "wooden house"; break;
53    }
54    if(field != null) {
55      Label l = new Label(field);
56      DOM.setStyleAttribute(l.getElement(), "backgroundColor", "lightblue"
          );
57      this.add(l);
58    }
59    if (model.getBottomFence() > 0 || model.getLeftFence() > 0|| model.
        getRightFence() > 0|| model.getTopFence() > 0) {
60      String color = null;
61      switch(model.getPlayer()) {
62      case BLUE: color = "blue"; break;
63      case GREEN: color = "green"; break;
64      case NONE: color = null; break;
65      case ROSA: color = "pink"; break;
66      case RED: color = "red"; break;
67      }
68      if(color != null) {
69        color += " solid 5px";
70        String noColor = "black 1px solid";
71        DOM.setStyleAttribute(getElement(), "borderLeft", model.
              getLeftFence() > 0?color:noColor);
72        DOM.setStyleAttribute(getElement(), "borderRight", model.
              getRightFence() > 0?color:noColor);
73        DOM.setStyleAttribute(getElement(), "borderTop", model.getTopFence
              () > 0?color:noColor);
74        DOM.setStyleAttribute(getElement(), "borderBottom", model.
              getBottomFence() > 0?color:noColor);
75      }
76    } else {
```

```
77        DOM.setStyleAttribute(getElement(), "borderLeft", "black 1px solid")
              ;
78        DOM.setStyleAttribute(getElement(), "borderRight", "black 1px solid"
              );
79        DOM.setStyleAttribute(getElement(), "borderTop", "black 1px solid");
80        DOM.setStyleAttribute(getElement(), "borderBottom", "black 1px solid
              ");
81      }
82      if(model.isSelectable()) {
83        DOM.setStyleAttribute(this.getElement(), "backgroundColor", "
              lightgreen");
84      } else {
85        DOM.setStyleAttribute(this.getElement(), "backgroundColor", "white")
              ;
86      }
87
88      if(model.getPersonsCount() - model.getPersonsAtWork() + model.
            getChildCount() > 0) {
89        this.add(new ChildRenderer(model));
90        if(model.getPersonsCount() - model.getPersonsAtWork() + 0.5*model.
              getChildCount() != 1) {
91          DoubleBox tb = new DoubleBox();
92          tb.setVisibleLength(2);
93          tb.setValue(model.getPersonsCount() - model.getPersonsAtWork() +
                0.5*model.getChildCount());
94          tb.setReadOnly(true);
95          this.add(tb);
96        }
97      }
98      this.add(new ResourceRenderer(model));
99    }
100 }
```

Listing A121: SmallFieldRenderer.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import java.util.ArrayList;
4
5  import com.google.gwt.event.dom.client.ClickEvent;
6  import com.google.gwt.event.dom.client.ClickHandler;
7  import com.google.gwt.event.dom.client.HasClickHandlers;
8  import com.google.gwt.event.shared.HandlerRegistration;
9  import com.google.gwt.uibinder.client.UiChild;
10 import com.google.gwt.user.client.ui.VerticalPanel;
11 import com.google.gwt.user.client.ui.Widget;
12
13 import de.tu_freiberg.informatik.vonwenckstern.client.view.Tooltip;
14
15 public class TooltipPanel extends VerticalPanel implements
       HasClickHandlers {
16   @UiChild(limit=1)
17   public void addTooltip(Widget tooltip) {
18     new Tooltip(this, tooltip);
19   }
20
21   private ArrayList<HandlerRegistration> clickHandlers = new  ArrayList<
         HandlerRegistration>();
22
23   public void removeAllClickHandlers() {
24     int size =clickHandlers.size()-1;
25     for(int i=size; i>=0; i--) {
26       if(clickHandlers.get(i) != null) {
27         clickHandlers.get(i).removeHandler();
28       }
29       clickHandlers.remove(i);
30     }
31   }
32
33   @Override
34   public HandlerRegistration addClickHandler(ClickHandler handler) {
35     HandlerRegistration handlerReg = this.addDomHandler(handler,
           ClickEvent.getType());
36     clickHandlers.add(handlerReg);
37     return handlerReg;
38   }
39 }
```

Listing A122: TooltipPanel.java file

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.view.mobile;
2
3  import com.google.gwt.user.client.DOM;
4  import com.google.gwt.user.client.ui.Label;
5
6  import de.tu_freiberg.informatik.vonwenckstern.client.model.BaseFieldModel
       ;
7  import de.tu_freiberg.informatik.vonwenckstern.client.model.
       HasBaseFieldModel;
8  import de.tu_freiberg.informatik.vonwenckstern.client.view.Renderer;
9  public class TooltipPanelChildRenderer extends TooltipPanel implements
       Renderer, HasBaseFieldModel {
10
11   public TooltipPanelChildRenderer() {
12     super();
13     this.setPixelSize(100, 100);
```

```
14      DOM.setStyleAttribute(this.getElement(), "border", "1px solid black");
15              }
16
17    private String fixText = null;
18
19    public String getFixText() {
20      return fixText;
21    }
22
23    public void setFixText(String fixText) {
24      this.fixText = fixText;
25      render();
26    }
27
28      // setId will not work in UiBuilder
29      public void setValue(String value) {
30        model.setId(value);
31      }
32
33      BaseFieldModel model = new BaseFieldModel();
34
35      @Override
36      public void render() {
37        this.clear();
38        if(fixText != null) {
39          Label l = new Label(fixText);
40          DOM.setStyleAttribute(l.getElement(), "backgroundColor", "
                lightblue");
41          this.add(l);
42        }
43        this.add(new ChildRenderer(model));
44      }
45
46      @Override
47      public BaseFieldModel getModel() {
48        return model;
49      }
50
51      public void setModel(BaseFieldModel model) {
52        this.model = model;
53        if(model != null) {
54          render();
55        }
56      }
57  }
```

Listing A123: TooltipPanelChildRenderer.java file

A 3.2 Source code of GWT and JSF comparison

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3
4  import javax.faces.bean.*;
5
6  @SessionScoped
7  @ManagedBean
8  public class Info {
9    private String[] uniNews = new String[] {"<b>Minister of Science
       visits TU Freiberg </b><br>Headmaster welcomes Prof. Sabine
       von Schorlemer...",
10     "<b>New faculty directors elected </b><br>Since the election
         on January 31st, we have new faculty directors in our six
         institutes ...",
11     "<b>Prof. Stoyan gives a lection out of his new book </b><br>
         Prof. Dietrich Stoyan was a professor for applied
         mathematics ..."};
12
13   private String[] majors = new String[] {"Applied Mathematics", "
       Computer Science", "Geoecology"};
14
15   public String[] getMajors() {
16     return majors;
17   }
18
19   public void setMajors(String[] majors) {
20     this.majors = majors;
21   }
22
23   private int uniNewsIndex = 0;
24   public int getMaxUniversityNews() {
25     // normally you would do a count query in your database
26     return 3;
27   }
28
29   public String decUniversityNews() {
30     setUniNewsIndex(getUniNewsIndex()- 1);
31     return "main";
32   }
33
34   public String incUniversityNews() {
35     setUniNewsIndex(getUniNewsIndex()+ 1);
36     return "main";
37   }
38
39   public String getUniversityNews() {
40     // normally you would do a database query
41     return uniNews[uniNewsIndex];
42   }
43
44   public void setUniNewsIndex(int i) {
45     if(i>=0 && i<getMaxUniversityNews())
46       uniNewsIndex = i;
47   }
```

```
48
49    public int getUniNewsIndex() {
50      return uniNewsIndex;
51    }
52  }
```

Listing A124: Source code of Info class in JSF GWT comparison in category 1

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml"
4        xmlns:h="http://java.sun.com/jsf/html"
5        xmlns:f="http://java.sun.com/jsf/core">
6  <h:head>
7  <title>TU Freiberg</title>
8  <link href="./css/styles.css"
9        rel="stylesheet" type="text/css"/>
10 </h:head>
11 <h:body><h:form>
12 <h1>TU Home</h1>
13 <table><tr><td>Navigation: <a href="majors.jsf">majors</a> <a
       href="institutes.jsf">institutes</a> ...</td></tr>
14 <tr><td>
15 <table><tr><td  style="vertical-align:top;">Informations for:<br/>
     <a href="pupils.jsf">Pupils</a><br/><a href="">Students</a><br/
     ><a href="">Post-graduate</a></td><td style="vertical-align:top
     ; width:400px;">Logo <br/>
16 <div style="background-color:lightblue; border-top: 1px solid gray
     ; border-bottom: 1px solid gray;">News</div><div>
17 <h:panelGroup id="uniNews">
18   <h:outputText value="#{info.universityNews}" escape="false"/><br
       />
19   <f:ajax render="uniNews">
20     <div style="float: left;"><h:commandLink value="&lt;&lt;"
         action="#{info.decUniversityNews}" rendered="#{info.
         uniNewsIndex != 0}"/></div>
21     <div style="float: right;"><h:commandLink value="&gt;&gt;"
         action="#{info.incUniversityNews}" rendered="#{info.
         uniNewsIndex != info.maxUniversityNews-1}"/></div>
22   </f:ajax>
23 </h:panelGroup>
24 </div></td></tr></table>
25 </td></tr>
26 </table>
27 </h:form>
28 </h:body></html>
```

Listing A125: Complete source code of main.xhtml file in JSF GWT comparison in category 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:c="http://java.sun.com/jsp/jstl/core">
<h:head>
<title>TU Freiberg: Majors</title>
<link href="./css/styles.css"
      rel="stylesheet" type="text/css"/>
</h:head>
<h:body><h:form>
<h1>Majors</h1>
<table><tr><td>Navigation: <a href="">majors</a> <a href="">
    institutes</a> ...</td></tr>
<tr><td>
<h:panelGrid columns="1">
  <c:forEach items="#{info.majors}" var="item">
    <h:panelGroup>
      <h:outputText value="#{item}" />
    </h:panelGroup>
  </c:forEach>
</h:panelGrid>
</td></tr>
</table>
</h:form>
</h:body></html>
```

Listing A126: Complete source code of majors.xhtml file in JSF GWT comparison in category 1

```
package de.tu_freiberg.informatik.vonwenckstern.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.logical.shared.ValueChangeEvent;
import com.google.gwt.event.logical.shared.ValueChangeHandler;
import com.google.gwt.http.client.Request;
import com.google.gwt.http.client.RequestBuilder;
import com.google.gwt.http.client.RequestCallback;
import com.google.gwt.http.client.RequestException;
import com.google.gwt.http.client.Response;
import com.google.gwt.http.client.URL;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Frame;
import com.google.gwt.user.client.ui.HTML;
import com.google.gwt.user.client.ui.HTMLPanel;
import com.google.gwt.user.client.ui.RootPanel;

public class InfoSite implements EntryPoint, ValueChangeHandler<
    String>{
  public void onModuleLoad() {
    History.addValueChangeHandler(this);
    History.fireCurrentHistoryState();
```

```
26    }
27    @Override
28    public void onValueChange(final ValueChangeEvent<String> event)
          {
29      String url;
30      if(event.getValue() == null || event.getValue().length() == 0)
            {
31        url = "main.html";
32      } else {
33        url = URL.decode(event.getValue());
34      }
35      RequestBuilder builder = new RequestBuilder(RequestBuilder.
          POST, url);
36      try {
37        builder.sendRequest("GET " + url, new RequestCallback() {
38          @Override
39          public void onResponseReceived(Request request, Response
              response) {
40            String t = response.getText();
41            t = replaceHyperlinks(t);
42            int id = 0;
43            int uniNews = -1;
44            int majors = -1;
45            if(t.contains("[{[universityNews]}]")) {
46              uniNews = id;
47              t = t.replace("[{[universityNews]}]", "<div id=\"gwt-
                    substitute"+id+"\"></div>");
48              id++;
49            }
50            if(t.contains("[{[majors]}]")) {
51              majors = id;
52              t = t.replace("[{[majors]}]", "<div id=\"gwt-
                    substitute"+id+"\"></div>");
53              id++;
54            }
55            HTMLPanel f = new HTMLPanel(t);
56            if(uniNews > -1) {
57              f.add(new UniversityNews(), "gwt-substitute"+uniNews);
58            }
59            if(majors > -1) {
60              f.add(new Majors(), "gwt-substitute"+majors);
61            }
62            RootPanel.get().clear();
63            RootPanel.get().add(f);
64          }
65          @Override
66          public void onError(Request request, Throwable exception)
                {
67            Window.alert("Failure: Could not load the page:" + URL.
                  decode(event.getValue()) + " from the server.");
68          }
69        });
70      } catch (RequestException e) {
71        e.printStackTrace();
72      }
73    }
```

```
74    private String replaceHyperlinks(String t) {
75      int startIndex = 0;
76      while(t.indexOf("<a", startIndex) > -1) {
77        int aBeg = t.indexOf("<a ", startIndex);
78        int aEnd = t.indexOf("</a", aBeg);
79        startIndex = aEnd;
80        int aHref = t.indexOf("href=", aBeg);
81        if(aHref < aEnd) {
82          int start = t.indexOf("\"", aHref);
83          int end = t.indexOf("\"", start+1);
84          if(end < aEnd) {
85            String link = t.substring(start+1, end);
86            link = "#" + URL.encode(link);
87            t = t.substring(0, start+1) + link + t.substring(end);
88          }
89        }
90      }
91      return t;
92    }
93  }
```

Listing A127: Source code of InfoSite class in JSF GWT comparison in category 1

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.event.dom.client.ClickEvent;
5  import com.google.gwt.event.dom.client.ClickHandler;
6  import com.google.gwt.uibinder.client.UiBinder;
7  import com.google.gwt.uibinder.client.UiField;
8  import com.google.gwt.user.client.Window;
9  import com.google.gwt.user.client.WindowScrollListener;
10 import com.google.gwt.user.client.rpc.AsyncCallback;
11 import com.google.gwt.user.client.ui.Composite;
12 import com.google.gwt.user.client.ui.HTML;
13 import com.google.gwt.user.client.ui.Label;
14 import com.google.gwt.user.client.ui.Widget;
15
16 public class UniversityNews extends Composite implements
       ClickHandler{
17
18   private static final Binder binder = GWT.create(Binder.class);
19
20   interface Binder extends UiBinder<Widget, UniversityNews> {
21   }
22   @UiField
23   HTML content;
24   @UiField
25   HTML prev;
26   @UiField
27   HTML next;
28
29   private int index = 0;
30   private int newsCount = 0;
31
32   public UniversityNews() {
33     initWidget(binder.createAndBindUi(this));
```

```
34      prev.addClickHandler(this);
35      next.addClickHandler(this);
36      InfoService.Util.getInstance().newsCount(new AsyncCallback<
            Integer>() {
37        @Override
38        public void onSuccess(Integer result) {
39          newsCount = result;
40          loadNews();
41        }
42        @Override
43        public void onFailure(Throwable caught) {
44          Window.alert("Could not load news count");
45        }
46      });
47    }
48
49    public void loadNews() {
50      if(index < 0 || index >= newsCount)
51        return;
52      prev.setVisible(index > 0);
53      next.setVisible(index < newsCount - 1);
54      InfoService.Util.getInstance().getNews(index, new
            AsyncCallback<String>() {
55
56        @Override
57        public void onSuccess(String result) {
58          content.setHTML(result);
59        }
60
61        @Override
62        public void onFailure(Throwable caught) {
63          Window.alert("News could not get loaded!");
64        }
65      });
66    }
67
68    @Override
69    public void onClick(ClickEvent event) {
70      if(event.getSource() == prev) {
71        index--;
72      } else if(event.getSource() == next) {
73        index++;
74      }
75      loadNews();
76    }
77
78  }
```

Listing A128: Source code of UniversityNews class in JSF GWT comparison in category 1

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3  import java.util.ArrayList;
4  import javax.faces.bean.*;
5  import javax.faces.model.SelectItem;
6
7  @SessionScoped
8  @ManagedBean
9  public class Survey {
10   private String name="von␣Wenckstern";
11   private String firstName="Michael";
12   private boolean male=true;
13
14   private boolean hobbySoccer=true;
15   private boolean hobbyTennis=true;
16   private boolean hobbyBasketball=false;
17   private boolean hobbyVolleyball=true;
18   private boolean hobbyFootball=false;
19   private boolean hobbyBaseball=false;
20   private ArrayList<SelectItem> friends = new ArrayList<SelectItem
       >();
21   private String newFriend;
22   private int selectedItem;
23
24
25
26   public int getSelectedItem() {
27     return selectedItem;
28   }
29
30   public void setSelectedItem(int selectedItem) {
31     this.selectedItem = selectedItem;
32   }
33
34   public String getNewFriend() {
35     return newFriend;
36   }
37
38   public void setNewFriend(String newFriend) {
39     this.newFriend = newFriend;
40   }
41
42   public Survey() {
43     friends.add(new SelectItem(0, "Maren"));
44     friends.add(new SelectItem(1, "Oliver"));
45     friends.add(new SelectItem(2, "Asti"));
46     friends.add(new SelectItem(3, "Steph"));
47     friends.add(new SelectItem(3, "Tom"));
48   }
49
50   public ArrayList<SelectItem> getFriends() {
51     return friends;
52   }
53
54   public String add() {
```

```
55      friends.add(new SelectItem(friends.size(), newFriend));
56      newFriend = null;
57      return "page2";
58    }
59
60    public String getAllFriends() {
61      String s = "";
62      for(SelectItem item : friends) {
63        s += item.getLabel() + ", ";
64      }
65      return s;
66    }
67
68    public String delete() {
69      try {
70        friends.remove(selectedItem);
71      } catch(Exception e) {}
72      return "page2";
73    }
74
75    public boolean isHobbyBaseball() {
76      return hobbyBaseball;
77    }
78
79    public void setHobbyBaseball(boolean hobbyBaseball) {
80      this.hobbyBaseball = hobbyBaseball;
81    }
82
83    public String goToPage1() {
84      return "page1";
85    }
86
87    public String goToPage2() {
88      return "page2";
89    }
90
91    public String goToPage3() {
92      return "page3";
93    }
94
95    public String getName() {
96      return name;
97    }
98
99    public void setName(String name) {
100     this.name = name;
101   }
102
103   public String getFirstName() {
104     return firstName;
105   }
106
107   public void setFirstName(String firstName) {
108     this.firstName = firstName;
109   }
110
```

```
111   public boolean isMale() {
112     return male;
113   }
114
115   public void setMale(boolean male) {
116     this.male = male;
117   }
118
119   public boolean isHobbySoccer() {
120     return hobbySoccer;
121   }
122
123   public void setHobbySoccer(boolean hobbySoccer) {
124     this.hobbySoccer = hobbySoccer;
125   }
126
127   public boolean isHobbyTennis() {
128     return hobbyTennis;
129   }
130
131   public void setHobbyTennis(boolean hobbyTennis) {
132     this.hobbyTennis = hobbyTennis;
133   }
134
135   public boolean isHobbyBasketball() {
136     return hobbyBasketball;
137   }
138
139   public void setHobbyBasketball(boolean hobbyBasketball) {
140     this.hobbyBasketball = hobbyBasketball;
141   }
142
143   public boolean isHobbyVolleyball() {
144     return hobbyVolleyball;
145   }
146
147   public void setHobbyVolleyball(boolean hobbyVolleyball) {
148     this.hobbyVolleyball = hobbyVolleyball;
149   }
150
151   public boolean isHobbyFootball() {
152     return hobbyFootball;
153   }
154
155   public void setHobbyFootball(boolean hobbyFootball) {
156     this.hobbyFootball = hobbyFootball;
157   }
158
159 }
```

Listing A129: Source code of Survey managed bean class in JSF GWT comparison in category 2

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:f="http://java.sun.com/jsf/core">
<h:head>
<title>Survey page3</title>
<link href="./css/styles.css"
  rel="stylesheet" type="text/css"/>
</h:head>
<h:body>
<h:form>
<h1 style="text-align:left;">Page 3</h1>
<h2 style="text-align:left;">Summery of your input data</h2>
<table><tr><td>Name:</td><td>#{survey.name}</td></tr>
<tr><td>First name:</td><td>#{survey.firstName}</td></tr>
<tr><td>Sex: </td><td>#{survey.male ? 'male' : 'female'}</td></tr>
<tr><td>Hobbies:</td><td>#{survey.hobbySoccer ? 'soccer, ' : ''}
    #{survey.hobbyTennis ? 'tennis, ' : ''} #{survey.
    hobbyBasketball ? 'basketball, ' : ''} #{survey.hobbyBaseball ?
     'baseball, ' : ''} #{survey.hobbyVolleyball ? 'volleyball, ' :
     ''} #{survey.hobbyFootball ? 'football, ' : ''}</td></tr>
<tr><td>Friends:</td><td>#{survey.allFriends}</td></tr>
<tr><td></td><td> <h:commandButton value="&lt;&lt; previous"
    action="#{survey.goToPage2}"/> </td></tr>
</table>
</h:form>
</h:body></html>
```

Listing A130: XML code of page3.xhtml in JSF GWT comparison in category 2

```
package de.tu_freiberg.informatik.vonwenckstern;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;

import javax.faces.bean.*;
import javax.faces.context.FacesContext;

@RequestScoped
@ManagedBean
public class Forum {
  private String newTopicName;
  private String userName;
  private String entryText;
```

```
26   public String getUserName() {
27     return userName;
28   }
29
30   public void setUserName(String userName) {
31     this.userName = userName;
32   }
33
34   public String getEntryText() {
35     return entryText;
36   }
37
38   public void setEntryText(String entryText) {
39     this.entryText = entryText;
40   }
41
42   public String getNewTopicName() {
43     return newTopicName;
44   }
45
46   public void setNewTopicName(String newTopicName) {
47     this.newTopicName = newTopicName;
48   }
49
50   public String openNewTopic() {
51     if(newTopicName != null && newTopicName.length() > 0) {
52       ArrayList<Topic> atopic = new ArrayList<Topic>(Arrays.asList
           (topics));
53       newTopicName = newTopicName.replace("&", "&amp;").replace("<
           ", "&lt;").replace(">", "&gt;").replace("\n", "<br>");
54       atopic.add(new Topic(newTopicName, atopic.size(), new Entry
           [] {}));
55       topics = atopic.toArray(topics);
56       save();
57       newTopicName = null;
58     }
59     return "topics";
60   }
61
62   public String addNewEntry() {
63     if(userName != null && userName.length() > 0 && entryText !=
         null && entryText.length() > 0) {
64       ArrayList<Entry> aentry = new ArrayList<Entry>(Arrays.asList
           (topics[topicId].entries));
65       entryText = entryText.replace("&", "&amp;").replace("<", "&
           lt;").replace(">", "&gt;").replace("\n", "<br>");
66       aentry.add(new Entry(userName, new Date(), entryText));
67       topics[topicId].entries = aentry.toArray(topics[topicId].
           entries);
68       save();
69       userName = null;
70       entryText = null;
71     }
72     return "showtopic";
73   }
74
```

```
75   public Forum() {
76     try {
77       FileInputStream fin = new FileInputStream(new File("C:\\GWT
             \\workspace\\DA_Forum_JSF\\topics.data"));
78       ObjectInputStream objin = new ObjectInputStream(fin);
79       topics = (Topic[]) objin.readObject();
80       objin.close();
81       fin.close();
82     } catch (Exception e) {
83       e.printStackTrace();
84     }
85   }
86
87   private void save() {
88     try {
89       FileOutputStream fout = new FileOutputStream(new File("C:\\
             GWT\\workspace\\DA_Forum_JSF\\topics.data"));
90       ObjectOutputStream objout = new ObjectOutputStream(fout);
91       objout.writeObject(topics);
92       objout.close();
93     } catch (Exception e) {
94       e.printStackTrace();
95     }
96   }
97
98   private Topic[] topics;
99
100  @ManagedProperty(value="#{param.id}")
101  private int topicId;
102
103  public int getTopicId() {
104    return topicId;
105  }
106
107  public Topic getTopic() {
108    return topics[topicId];
109  }
110
111  public void setTopicId(int topicId) {
112    this.topicId = topicId;
113  }
114
115  public Topic[] getTopics() {
116    return topics;
117  }
118
119  public static class Entry implements Serializable {
120    private static final long serialVersionUID = 1L;
121    private String userName; // normally an id and the user has an
           extra table
122    private Date time;
123    private String content;
124
125    public Entry(String userName, Date time, String content) {
126      this.userName = userName;
127      this.time = time;
```

```
128         this.content = content;
129       }
130
131       public String getUserName() {
132         return userName;
133       }
134
135       public Date getTime() {
136         return time;
137       }
138
139       public String getContent() {
140         return content;
141       }
142     }
143
144     public static class Topic implements Serializable {
145       private static final long serialVersionUID = 1L;
146       private String name;
147       private int id;
148       private Entry[] entries;
149
150       public Topic(String name, int id, Entry[] entries) {
151         this.name = name; this.id = id;
152         this.entries = entries;
153       }
154
155       public Entry[] getEntries() {
156         return entries;
157       }
158
159       public Entry getEntry(int i) {
160         return entries[i];
161       }
162
163       public String getName() {
164         return name;
165       }
166
167       public void setName(String name) {
168         this.name = name;
169       }
170
171       public int getId() {
172         return id;
173       }
174
175       public void setId(int id) {
176         this.id = id;
177       }
178     }
179 }
```

Listing A131: Source code of Forum managed bean class in JSF GWT comparison in category 3

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:c="http://java.sun.com/jsp/jstl/core"
    xmlns:fn="http://java.sun.com/jsp/jstl/functions">
<h:head>
<title>Topic #{forum.topic.name}</title>
<link href="./css/styles.css"
    rel="stylesheet" type="text/css"/>
</h:head>
<h:body><h:form>
<h1><h:outputText  value="Topic #{forum.topic.name}" escape="false
    " /></h1>
<a href="topics.jsf">back to topics overview</a>
<h:panelGroup id="entries">
<h:panelGrid columns="2">
  <c:forEach items="#{forum.topic.entries}" var="item">
    <h:panelGroup>
      <h:outputText value="#{item.userName}"/><br/>
      <div style="color: darkgray;">#{item.time}</div>
    </h:panelGroup>
    <h:panelGroup>
      <h:outputText  value="#{item.content}" escape="false" />
    </h:panelGroup>
  </c:forEach>
</h:panelGrid><br/><br/>
<h:panelGroup id="input">
<div style='visibility:hidden'>Id: <h:inputText value="#{forum.
    topicId}"/> <br/></div>
Name: <h:inputText value="#{forum.userName}"/> <br/>
Text: <h:inputTextarea value="#{forum.entryText}" />
</h:panelGroup>
</h:panelGroup>
<f:ajax render="entries" execute="input">
  <h:commandButton value="add new entry" action="#{forum.
      addNewEntry}" />
</f:ajax>
</h:form>
</h:body></html>
```

Listing A132: XML code of showtopic.xhtml in JSF GWT comparison in category 3

```
package de.tu_freiberg.informatik.vonwenckstern.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.logical.shared.ValueChangeEvent;
import com.google.gwt.event.logical.shared.ValueChangeHandler;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Grid;
```

```
13 import com.google.gwt.user.client.ui.HTML;
14 import com.google.gwt.user.client.ui.HorizontalPanel;
15 import com.google.gwt.user.client.ui.RootPanel;
16 import com.google.gwt.user.client.ui.TextArea;
17 import com.google.gwt.user.client.ui.TextBox;
18 import com.google.gwt.user.client.ui.VerticalPanel;
19
20 import de.tu_freiberg.informatik.vonwenckstern.client.shared.Entry
       ;
21 import de.tu_freiberg.informatik.vonwenckstern.client.shared.Topic
       ;
22 import de.tu_freiberg.informatik.vonwenckstern.client.shared.
       TopicInformation;
23
24 public class Forum implements EntryPoint, ValueChangeHandler<
       String>, ClickHandler {
25   private TextBox textbox = new TextBox();
26   private TextArea textarea = new TextArea();
27   private Button btn = new Button();
28   private int id = -1;
29
30   public void onModuleLoad() {
31     btn.addClickHandler(this);
32     History.addValueChangeHandler(this);
33     History.fireCurrentHistoryState();
34   }
35
36   @Override
37   public void onValueChange(ValueChangeEvent<String> e) {
38     String s = e.getValue();
39     if(s == null || s.isEmpty()) {
40       RootPanel.get().clear();
41       RootPanel.get().add(new HTML("<h1>Forum topics</h1>"));
42       ForumService.Util.getInstance().getTopics(new AsyncCallback<
           TopicInformation[]>() {
43         @Override
44         public void onSuccess(TopicInformation[] s) {
45           Grid g = new Grid(s.length, 3);
46           for(int i=0; i<s.length; i++) {
47             g.setHTML(i, 0, s[i].getName());
48             g.setText(i, 1, s[i].getNumberOfEntries() + " Entries"
                   );
49             g.setHTML(i, 2, "<a href=\"#"+ s[i].getId() + "\">open
                   topic</a>");
50           }
51           RootPanel.get().add(g);
52           textbox.setValue(null);
53           btn.setText("open a new topic");
54           HorizontalPanel hp = new HorizontalPanel();
55           hp.add(textbox);
56           hp.add(btn);
57           RootPanel.get().add(hp);
58
59         }
60         @Override
61         public void onFailure(Throwable arg0) {
```

```
62            Window.alert("Failure: could not load forum topics");
63          }
64        });
65      } else {
66        id = -1;
67        try {
68          id = Integer.parseInt(s);
69        } catch(Exception ex) {}
70        if(id >= 0) {
71          ForumService.Util.getInstance().getTopic(id, new
                AsyncCallback<Topic>() {
72            @Override
73            public void onSuccess(Topic t) {
74              RootPanel.get().clear();
75              RootPanel.get().add(new HTML("<h1>Topic " + t.getName
                    () + "</h1><a href=\"#\">back to topics overview</a
                    >"));
76              Entry[] en = t.getEntries();
77              Grid g = new Grid(en.length, 2);
78              for(int i=0; i<en.length; i++) {
79                g.setHTML(i, 0,en[i].getUserName() + "<br><span
                      style=\"color: darkgray;\">" + en[i].getTime() +
                      "</span>");
80                g.setHTML(i, 1, en[i].getContent());
81              }
82              RootPanel.get().add(g);
83              Grid g2 = new Grid(3, 2);
84              g2.setText(0, 0, "Name: ");
85              g2.setWidget(0, 1, textbox);
86              g2.setText(1, 0, "Text: ");
87              g2.setWidget(1, 1, textarea);
88              g2.setText(2, 0, "");
89              g2.setWidget(2, 1, btn);
90              btn.setText("add new entry");
91              RootPanel.get().add(g2);
92            }
93            @Override
94            public void onFailure(Throwable caught) {
95              Window.alert("Failure: could not load topic from
                    server");
96            }
97          });
98        }
99      }
100   }
101
102   @Override
103   public void onClick(ClickEvent event) {
104     if(btn.getText().equals("open a new topic")) {
105       if(textbox.getValue() == null || textbox.getValue().isEmpty
              ()) {
106         Window.alert("You have enter a name for your new topic");
107       } else {
108         ForumService.Util.getInstance().addNewTopic(textbox.
              getValue(), new AsyncCallback<Void>() {
109           @Override
```

```
110             public void onSuccess(Void result) {
111               textbox.setValue(null);
112               History.fireCurrentHistoryState(); // to reload the
                      content
113             }
114
115             @Override
116             public void onFailure(Throwable caught) {
117               Window.alert("Failure: could not add new topic");
118             }
119           });
120         }
121       } else if(btn.getText().equals("add new entry")) {
122         if(textbox.getValue() == null || textbox.getValue().isEmpty
              () || textarea.getValue() == null || textarea.getValue().
              isEmpty()) {
123           Window.alert("You have enter your user name and any
                content for your new entry");
124         } else if(id >= 0){
125           ForumService.Util.getInstance().addNewEntry(textbox.
                getValue(), textarea.getValue(), id, new AsyncCallback<
                Void>() {
126             @Override
127             public void onSuccess(Void result) {
128               textbox.setValue(null);
129               textarea.setValue(null);
130               History.fireCurrentHistoryState(); // to reload the
                      content
131             }
132
133             @Override
134             public void onFailure(Throwable caught) {
135               Window.alert("Failure: could not add new entry");
136             }
137           });
138         }
139       }
140     }
141 }
```

Listing A133: Source code of Forum GWT class in JSF GWT comparison in category 3

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import java.io.Serializable;
4  import java.util.Date;
5
6  import com.google.gwt.core.client.GWT;
7  import com.google.gwt.user.client.rpc.RemoteService;
8  import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;
9
10 import de.tu_freiberg.informatik.vonwenckstern.client.shared.Topic
       ;
11 import de.tu_freiberg.informatik.vonwenckstern.client.shared.
       TopicInformation;
12
13 @RemoteServiceRelativePath("ForumService")
14 public interface ForumService extends RemoteService {
15   public TopicInformation[] getTopics();
16   public Topic getTopic(int id);
17   public void addNewTopic(String name);
18   public void addNewEntry(String userName, String content, int
         topicId);
19
20   public static class Util {
21
22     private static ForumServiceAsync instance;
23     public static ForumServiceAsync getInstance(){
24       if (instance == null) {
25         instance = GWT.create(ForumService.class);
26       }
27       return instance;
28     }
29   }
30 }
```

Listing A134: Source code of ForumService RPC interface in JSF GWT comparison in category 3

```
1  package de.tu_freiberg.informatik.vonwenckstern.client.shared;
2
3  import java.io.Serializable;
4
5  public class TopicInformation implements Serializable {
6    private static final long serialVersionUID = 1L;
7    private String name;
8    private int id;
9    private int numberOfEntries;
10
11   public TopicInformation() {}
12
13   public TopicInformation(String name, int id, int numberOfEntries
         ) {
14     this.name = name; this.id = id; this.numberOfEntries =
           numberOfEntries;
15   }
16
17   public String getName() {
18     return name;
19   }
20   public void setName(String name) {
21     this.name = name;
22   }
23   public int getId() {
24     return id;
25   }
26   public void setId(int id) {
27     this.id = id;
28   }
29   public int getNumberOfEntries() {
30     return numberOfEntries;
31   }
32   public void setNumberOfEntries(int numberOfEntries) {
33     this.numberOfEntries = numberOfEntries;
34   }
35 }
```

Listing A135: Source code of TopicInformation class in JSF GWT comparison in category 3

```
1  package de.tu_freiberg.informatik.vonwenckstern.server;
2
3  import java.io.File;
4  import java.io.FileInputStream;
5  import java.io.FileOutputStream;
6  import java.io.ObjectInputStream;
7  import java.io.ObjectOutputStream;
8  import java.util.ArrayList;
9  import java.util.Arrays;
10 import java.util.Date;
11
12 import de.tu_freiberg.informatik.vonwenckstern.client.ForumService
       ;
13 import de.tu_freiberg.informatik.vonwenckstern.client.shared.Entry
       ;
```

```
14 import de.tu_freiberg.informatik.vonwenckstern.client.shared.Topic
       ;
15 import de.tu_freiberg.informatik.vonwenckstern.client.shared.
       TopicInformation;
16
17 import com.google.gwt.user.server.rpc.RemoteServiceServlet;
18
19 public class ForumServiceImpl extends RemoteServiceServlet
       implements ForumService {
20   private static final long serialVersionUID = 1L;
21
22   public ForumServiceImpl() {
23     try {
24       FileInputStream fin = new FileInputStream(new File("C:\\GWT
           \\workspace\\DA_Forum2\\topics.data"));
25       ObjectInputStream objin = new ObjectInputStream(fin);
26       topics = (Topic[]) objin.readObject();
27       objin.close();
28       fin.close();
29     } catch (Exception e) {
30       e.printStackTrace();
31     }
32   }
33
34   private void save() {
35     try {
36       FileOutputStream fout = new FileOutputStream(new File("C:\\
           GWT\\workspace\\DA_Forum2\\topics.data"));
37       ObjectOutputStream objout = new ObjectOutputStream(fout);
38       objout.writeObject(topics);
39       objout.close();
40     } catch (Exception e) {
41       e.printStackTrace();
42     }
43   }
44
45   private Topic[] topics;
46
47   @Override
48   public TopicInformation[] getTopics() {
49     ArrayList<TopicInformation> list = new ArrayList<
         TopicInformation>();
50     for(Topic t:topics) {
51       list.add(new TopicInformation(t.getName(), t.getId(), t.
           getEntries().length));
52     }
53     TopicInformation[] s = new TopicInformation[list.size()];
54     s = list.toArray(s);
55     return s;
56   }
57
58   @Override
59   public Topic getTopic(int id) {
60     return topics[id];
61   }
62
```

```
63    @Override
64    public void addNewTopic(String name) {
65      if(name != null && !name.isEmpty()) {
66        ArrayList<Topic> atopic = new ArrayList<Topic>(Arrays.asList
              (topics));
67        atopic.add(new Topic(name, atopic.size(), new Entry[] {}));
68        topics = atopic.toArray(topics);
69        save();
70      }
71    }
72
73    @Override
74    public void addNewEntry(String userName, String content, int
          topicId) {
75      if(userName != null && userName.length() > 0 && content !=
            null && content.length() > 0) {
76        ArrayList<Entry> aentry = new ArrayList<Entry>(Arrays.asList
              (topics[topicId].getEntries()));
77        content = content.replace("&", "&amp;").replace("<", "&lt;")
              .replace(">", "&gt;").replace("\n", "<br>");
78        aentry.add(new Entry(userName, new Date(), content));
79        topics[topicId].setEntries(aentry.toArray(topics[topicId].
              getEntries()));
80        save();
81      }
82    }
83  }
```

Listing A136: Source code of ForumServiceImpl RPC server class in JSF GWT comparison in category 3

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3  import java.util.Collections;
4  import java.util.HashMap;
5  import java.util.LinkedList;
6  import java.util.List;
7  import java.util.Map;
8
9  import javax.faces.bean.*;
10 import javax.faces.context.FacesContext;
11
12 @ApplicationScoped
13 @ManagedBean
14 public class Chat {
15   private List<Message> list = Collections.synchronizedList(new
         LinkedList<Message>());
16   private Map<Integer, User> users = Collections.synchronizedMap(
         new HashMap<Integer, User>());
17
18   public void addUser(User u) {
19     users.put(u.getUserId(), u);
20   }
21
22   public String  login() {
23     User user = (User) FacesContext.getCurrentInstance().
           getExternalContext().getSessionMap().get("user");
24     addUser(user);
25     return "chatroom";
26   }
27
28   public String send() {
29     Message msg = (Message) FacesContext.getCurrentInstance().
           getExternalContext().getRequestMap().get("message");
30     addMessage(msg.clone());
31     msg.setMessage(null);
32     return "chatroom";
33   }
34
35   public void addMessage(Message msg) {
36     list.add(msg);
37     synchronized (this) {
38       this.notifyAll();
39     }
40   }
41
42   public String getUpdate() {
43     synchronized(this) {
44       boolean cont = true;
45       while(cont) {
46         try {
47           this.wait(); // waits until a new message comes in
48           cont = false;
49         } catch (InterruptedException e) {
50           e.printStackTrace();
51           cont = true;
```

```
52            }
53          }
54        }
55        return "chatroom";
56      }
57
58      public String getMessages() {
59        StringBuilder sb = new StringBuilder();
60        sb.append("<table␣style=\"width:␣600px;\">");
61        for(int i=0; i<list.size(); i++) {
62          Message m = list.get(i);
63          User u = users.get(m.getUserId());
64          sb.append("<tr>");
65          sb.append("<td>");
66          sb.append("<div␣style=\"width:␣100px;␣overflow:scroll;␣color
                :").append(u.isMale()?"blue":"red").append("\">").append(
                u.getNickName()).append("<br>").append("<span␣style=\"
                color:␣darkgray;\">");
67          sb.append(m.getTime()).append("</span>").append("</div>");
68          sb.append("</td>");
69          sb.append("<td>");
70          sb.append("<div␣style=\"width:␣500px;␣overflow:scroll;\">").
                append(m.getMessage()).append("</div>");
71          sb.append("</td>");
72          sb.append("</tr>");
73        }
74        sb.append("</table>");
75        return sb.toString();
76      }
77
78  }
```

Listing A137: Source code of Chat managed bean class in JSF GWT comparison in category 4

```java
package de.tu_freiberg.informatik.vonwenckstern;

import java.io.Serializable;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@SessionScoped
@ManagedBean
public class User implements Serializable {
  private static final long serialVersionUID = 1L;
  private static int userIds = 0;
  private int userId;
  private String nickName;
  private boolean isMale;

  public User clone() {
    User u = new User(nickName, isMale);
    return u;
  }

  public User() {
    userId = userIds++;
  }

  public User(String nickName, boolean isMale) {
    this();
    this.nickName = nickName;
    this.isMale = isMale;
  }

  public String getNickName() {
    return nickName;
  }
  public void setNickName(String nickName) {
    this.nickName = nickName;
  }
  public boolean isMale() {
    return isMale;
  }
  public void setMale(boolean isMale) {
    this.isMale = isMale;
  }
  public int getUserId() {
    return userId;
  }
}
```

Listing A138: Source code of User managed bean class in JSF GWT comparison in category 4

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3  import java.io.Serializable;
4  import java.util.Date;
5
6  import javax.faces.bean.ManagedBean;
7  import javax.faces.bean.ManagedProperty;
8  import javax.faces.bean.RequestScoped;
9
10 @RequestScoped
11 @ManagedBean
12 public class Message implements Serializable{
13   private static final long serialVersionUID = 1L;
14
15   private String message;
16
17   @ManagedProperty(value="#{user.userId}")
18   private int userId;
19
20   private Date time;
21   public Date getTime() {
22     return time;
23   }
24   public String getMessage() {
25     return message;
26   }
27   public void setMessage(String message) {
28     this.message = message;
29   }
30   public int getUserId() {
31     return userId;
32   }
33   public void setUserId(int userId) {
34     this.userId = userId;
35   }
36
37   public Message() {
38     time = new Date();
39   }
40
41   public Message(String message) {
42     this();
43     this.message = message;
44   }
45
46   public Message clone() {
47     Message m = new Message();
48     m.time = this.time;
49     m.message = this.message;
50     m.userId = this.userId;
51     return m;
52   }
53 }
```

Listing A139: Source code of Message managed bean class in JSF GWT comparison in category 4

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core">
<h:head>
<title>Login</title>
<link href="./css/styles.css"
    rel="stylesheet" type="text/css"/>
</h:head>
<h:body><h:form>
<h1>Login</h1>
Nick name: <h:inputText value="#{user.nickName}" /><br />
Gender: <h:selectOneRadio value="#{user.male}"><f:selectItem
    itemValue="true" itemLabel="male"/><f:selectItem itemValue="
    false" itemLabel="female"/></h:selectOneRadio>
<h:commandButton action="#{chat.login}" value="login" />
</h:form></h:body></html>
```

Listing A140: XML code of login.xhtml in JSF GWT comparison in category 4

```
package de.tu_freiberg.informatik.vonwenckstern.client;

import java.util.HashMap;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.logical.shared.ValueChangeEvent;
import com.google.gwt.event.logical.shared.ValueChangeHandler;
import com.google.gwt.safehtml.shared.SafeHtml;
import com.google.gwt.safehtml.shared.SafeHtmlBuilder;
import com.google.gwt.user.client.History;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Grid;
import com.google.gwt.user.client.ui.HTML;
import com.google.gwt.user.client.ui.HTMLPanel;
import com.google.gwt.user.client.ui.RadioButton;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.ScrollPanel;
import com.google.gwt.user.client.ui.TextArea;
import com.google.gwt.user.client.ui.TextBox;

import de.tu_freiberg.informatik.vonwenckstern.shared.Message;
import de.tu_freiberg.informatik.vonwenckstern.shared.User;

public class Chat implements EntryPoint, ValueChangeHandler<String
    > {
  private User user = null;
  public void onModuleLoad() {
    History.addValueChangeHandler(this);
    History.fireCurrentHistoryState();
  }

```

```
35   private final ClickHandler sendBtn = new ClickHandler() {
36     @Override
37     public void onClick(ClickEvent event) {
38       Message m = new Message(ta.getValue(), user.getUserId());
39       ChatService.Util.getInstance().addMessage(m, new
             AsyncCallback<Void>() {
40         @Override
41         public void onSuccess(Void result) {
42           History.fireCurrentHistoryState(); // reload chat
43         }
44         @Override
45         public void onFailure(Throwable caught) {
46           Window.alert("Failure: Could not send your new messge to
                 server.");
47         }
48       });
49     }
50   };
51
52   private final TextArea ta = new TextArea();
53
54   @Override
55   public void onValueChange(ValueChangeEvent<String> event) {
56     String token = event.getValue();
57     if(token == null || token.isEmpty() || token.equals("login"))
         {
58       final TextBox name = new TextBox();
59       final RadioButton rbMale = new RadioButton("sex", "male");
60       rbMale.setValue(true);
61       RadioButton rbFemale = new RadioButton("sex", "female");
62       final Button btn = new Button("login");
63       HTMLPanel panel = new HTMLPanel("<h1>Login</h1><br><table><
             tr><td>Nick name: </td><td><span id=\"hp-name\"></span></
             td></tr>" +
64           "<tr><td>Sex: </td><td><span id=\"hp-male\"></span><span
                 id=\"hp-female\"></span></td></tr><tr><td> </td
                 ><td><span id=\"hp-login\"></span></td></tr></table>"
                 );
65       panel.add(name, "hp-name");
66       panel.add(rbMale, "hp-male");
67       panel.add(rbFemale, "hp-female");
68       panel.add(btn, "hp-login");
69       btn.addClickHandler(new ClickHandler() {
70         @Override
71         public void onClick(ClickEvent event) {
72           ChatService.Util.getInstance().getNextUserId(new
                 AsyncCallback<Integer>() {
73             @Override
74             public void onSuccess(Integer userId) {
75               user = new User(name.getValue(), rbMale.getValue(),
                     userId);
76               ChatService.Util.getInstance().addUser(user, new
                     AsyncCallback<Void>() {
77                 @Override
78                 public void onSuccess(Void result) {
79                   History.newItem("chat");
```

```
80                  }
81                  @Override
82                  public void onFailure(Throwable caught) {
83                    Window.alert("Failure: You could not get logged
                          in.");
84                  }
85                });
86              }
87
88              @Override
89              public void onFailure(Throwable caught) {
90                // TODO Auto-generated method stub
91
92              }
93            });
94          }
95        });
96        RootPanel.get().clear();
97        ScrollPanel scroll = new ScrollPanel(panel);
98        scroll.setPixelSize(Window.getClientWidth() - 50, Window.
            getClientHeight() - 50);
99        RootPanel.get().add(scroll);
100       scroll.scrollToBottom();
101
102     } else if(token.equals("chat")) {
103       if(user == null) {
104         History.newItem("login"); // we need user information
              first
105       } else {
106         ChatService.Util.getInstance().getMessages(new
              AsyncCallback<Message[]>() {
107           @Override
108           public void onSuccess(final Message[] m) {
109             loadUsers(m);
110           }
111           @Override
112           public void onFailure(Throwable caught) {
113             Window.alert("Failure: Could not load messages");
114           }
115         });
116       }
117     }
118   }
119
120   protected void appendMessages(final Message[] m, HashMap<Integer
        , User> users) {
121     Grid g = new Grid(m.length+1, 2);
122     for(int i=0; i<m.length; i++) {
123       User us = users.get(m[i].getUserId());
124       SafeHtml h = new SafeHtmlBuilder().appendHtmlConstant("<span
            style=\"color: " + (us.isMale() ? "blue" : "red") +
125           "\">").appendEscaped(us.getNickName()).
                appendHtmlConstant("</span><br><span style=\"color:
                darkgray;\">")
126           .appendEscaped(m[i].getTime().toString()).
                appendHtmlConstant("</span>").toSafeHtml();
```

```
127         g.setHTML(i, 0, h);
128         g.setText(i, 1, m[i].getMessage());
129       }
130       Button btn = new Button("Send");
131       btn.addClickHandler(sendBtn);
132       g.setWidget(m.length, 0, ta);
133       g.setWidget(m.length, 1, btn);
134       RootPanel.get().clear();
135       RootPanel.get().add(g);
136     }
137
138     protected void loadUsers(final Message[] m) {
139       ChatService.Util.getInstance().getUsers(new AsyncCallback<User
            []>() {
140         @Override
141         public void onSuccess(User[] u) {
142           HashMap<Integer, User> users = new HashMap<Integer, User
                >();
143           for(User us: u) {
144             users.put(us.getUserId(), us);
145           }
146           appendMessages(m, users);
147           AsyncCallback<Void> cb = new AsyncCallback<Void>() {
148             @Override
149             public void onSuccess(Void result) {
150               History.fireCurrentHistoryState(); // reload chat
151             }
152
153             @Override
154             public void onFailure(Throwable caught) {
155               ChatService.Util.getInstance().update(this);
156             }
157           };
158           ChatService.Util.getInstance().update(cb);
159         }
160         @Override
161         public void onFailure(Throwable caught) {
162           Window.alert("Failure: Could not load users.");
163         }
164       });
165     }
166 }
```

Listing A141: Source code of Chat GWT class in JSF GWT comparison in category 4

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.user.client.rpc.RemoteService;
5  import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;
6
7  import de.tu_freiberg.informatik.vonwenckstern.shared.Message;
8  import de.tu_freiberg.informatik.vonwenckstern.shared.User;
9
10 @RemoteServiceRelativePath("ChatService")
11 public interface ChatService extends RemoteService {
12   public void addUser(User user);
13   public void addMessage(Message message);
14   public Message[] getMessages();
15   public User[] getUsers();
16   public void update();
17   public int getNextUserId();
18   /**
19    * Utility class for simplifying access to the instance of async
          service.
20    */
21   public static class Util {
22     private static ChatServiceAsync instance;
23     public static ChatServiceAsync getInstance(){
24       if (instance == null) {
25         instance = GWT.create(ChatService.class);
26       }
27       return instance;
28     }
29   }
30 }
```

Listing A142: Source code of ChatService RPC interface in JSF GWT comparison in category 4

```
1  package de.tu_freiberg.informatik.vonwenckstern.server;
2
3  import java.util.ArrayList;
4  import java.util.Collections;
5  import java.util.HashMap;
6  import java.util.LinkedList;
7  import java.util.List;
8  import java.util.Map;
9
10 import de.tu_freiberg.informatik.vonwenckstern.client.ChatService;
11 import de.tu_freiberg.informatik.vonwenckstern.shared.Message;
12 import de.tu_freiberg.informatik.vonwenckstern.shared.User;
13
14 import com.google.gwt.user.server.rpc.RemoteServiceServlet;
15
16 public class ChatServiceImpl extends RemoteServiceServlet
       implements ChatService {
17   private static final long serialVersionUID = 1L;
18   private static List<Message> list = Collections.synchronizedList
         (new LinkedList<Message>());
```

```
19    private static List<User> users = Collections.synchronizedList(
          new ArrayList<User>());
20    private static Object lock = new Object();
21
22    @Override
23    public void addUser(User user) {
24      users.add(user);
25    }
26
27    @Override
28    public void addMessage(Message message) {
29      synchronized(lock) {
30        list.add(message);
31        lock.notifyAll(); // notifies all that a new message comes
              in
32      }
33    }
34
35    @Override
36    public Message[] getMessages() {
37      synchronized (lock) {
38        Message[] msg = new Message[list.size()];
39        msg = list.toArray(msg);
40        return msg;
41      }
42    }
43
44    @Override
45    public User[] getUsers() {
46      synchronized (lock) {
47        User[] user = new User[users.size()];
48        user = users.toArray(user);
49        return user;
50      }
51    }
52
53    @Override
54    public void update() {
55      synchronized(lock) {
56        boolean cont = true;
57        while(cont) {
58          try {
59            lock.wait(); // waits until a new message comes in
60            cont = false;
61          } catch (InterruptedException e) {
62            e.printStackTrace();
63            cont = true;
64          }
65        }
66      }
67    }
68
69    @Override
70    public int getNextUserId() {
71      return User.getNextUserId();
72    }
73  }
```

Listing A143: Source code of ChatServiceImpl RPC server class in JSF GWT comparison in category 4

```
1  package de.tu_freiberg.informatik.vonwenckstern;
2
3  import java.io.Serializable;
4  import java.util.ArrayList;
5  import java.util.Random;
6
7  import javax.faces.bean.ManagedBean;
8  import javax.faces.bean.RequestScoped;
9  import javax.faces.bean.SessionScoped;
10 import javax.faces.context.FacesContext;
11 import javax.faces.event.AjaxBehaviorEvent;
12
13 @SessionScoped
14 @ManagedBean
15 public class Snake {
16
17   private ArrayList<Point> snakeList = new ArrayList<Point>();
18   private int keyCode = 39;
19   private String input="hallo";
20   private boolean lost = false;
21   private int iContinue = 1;
22   private Point food = new Point(25, 25);
23   private Random rand = new Random();
24
25   public int getiContinue() {
26     return iContinue;
27   }
28
29   public void setiContinue(int iContinue) {
30     this.iContinue = iContinue;
31   }
32
33   public String getInput() {
34     return input;
35   }
36
37   public void setInput(String input) {
38     this.input = input;
39   }
40
41   public int getKeyCode() {
42     return keyCode;
43   }
44
45   public void setKeyCode(int keyCode) {
46     this.keyCode = keyCode;
47   }
48
49   public Snake() {
50     snakeList.add(new Point(2, 0));
51     snakeList.add(new Point(1, 0));
52     snakeList.add(new Point(0, 0));
53   }
54
55   public boolean isDrawn(int x, int y) {
56     return snakeList.contains(new Point(x,y));
57   }
58
59   public boolean isFood(int x, int y) {
```

```
60      return food.equals(new Point(x,y));
61    }
62
63    public String getImage() {
64      StringBuilder sb = new StringBuilder();
65      sb.append("<table␣cellspacing=\"0\"␣cellpadding=\"0\">");
66      for(int y=0; y<50; y++) {
67        sb.append("<tr>");
68        for(int x=0; x<50; x++) {
69          sb.append("<td>");
70          sb.append("<div␣style=\"width:␣5px;␣height:␣5px;␣
                background-color:␣").append(isDrawn(x, y)?"black":
                isFood(x,y)?"red":"white").append(";\"></div>");
71          sb.append("</td>");
72        }
73        sb.append("</tr>");
74      }
75      sb.append("</table>");
76      if(lost) {
77        sb.append("<div␣style=\"color:␣red;\">You␣lost!</div>");
78      }
79      return sb.toString();
80    }
81
82    public void keyDown(AjaxBehaviorEvent e) {
83      input = null;
84      System.out.println(keyCode);
85    }
86
87    public String restart() {
88      keyCode = 39;
89      snakeList.clear();
90      snakeList.add(new Point(2, 0));
91      snakeList.add(new Point(1, 0));
92      snakeList.add(new Point(0, 0));
93      lost = false;
94      iContinue = 1;
95      food = new Point(25, 25);
96      return "snake";
97    }
98
99    public String timer() {
100     if(lost)
101       return "snake";
102     Point fp = snakeList.get(0).clone();
103     switch(keyCode) {
104     case 37: fp.x = fp.x - 1; break;
105     case 38: fp.y = fp.y - 1; break;
106     case 39:
107     default: fp.x = fp.x + 1; break;
108     case 40: fp.y = fp.y + 1; break;
109     }
110     if(fp.equals(food)) {
111       do {
112         int x = rand.nextInt(50);
113         int y = rand.nextInt(50);
```

```
114           food = new Point(x, y);
115         } while(snakeList.contains(food));
116       } else {
117         snakeList.remove(snakeList.size()-1);
118       }
119       if(snakeList.contains(fp) || fp.x < 0 || fp.x > 49 || fp.y < 0
              || fp.y > 49) {
120         lost = true;
121         iContinue = 0;
122       } else {
123         snakeList.add(0, fp);
124       }
125       return "snake";
126     }
127
128     public static class Point implements Serializable {
129       private static final long serialVersionUID = 1L;
130       public Point clone() {
131         return new Point(x,y);
132       }
133       public Point(int x, int y) { this.x = x; this.y = y; }
134       public boolean equals(Object o) {
135         if( !(o instanceof Point)) {
136           return false;
137         }
138         Point p = (Point)o;
139         return p.x == x && p.y == y;
140       }
141       public int x;
142       public int y;
143     }
144 }
```

Listing A144: Source code of Snake managed bean class in JSF GWT comparison in category 5

```
1  package de.tu_freiberg.informatik.vonwenckstern.client;
2
3  import java.util.ArrayList;
4  import java.util.Random;
5
6  import com.google.gwt.core.client.EntryPoint;
7  import com.google.gwt.event.dom.client.ClickEvent;
8  import com.google.gwt.event.dom.client.ClickHandler;
9  import com.google.gwt.event.dom.client.KeyDownEvent;
10 import com.google.gwt.event.dom.client.KeyDownHandler;
11 import com.google.gwt.user.client.Timer;
12 import com.google.gwt.user.client.ui.Button;
13 import com.google.gwt.user.client.ui.FocusPanel;
14 import com.google.gwt.user.client.ui.HTML;
15 import com.google.gwt.user.client.ui.RootPanel;
16
17 /**
18  * Entry point classes define <code>onModuleLoad()</code>.
19  */
20 public class Snake implements EntryPoint, KeyDownHandler,
       ClickHandler {
```

```
21   private ArrayList<Point> snakeList = new ArrayList<Point>();
22   private int keyCode = 39;
23   private boolean lost = false;
24   private Point food = new Point(25, 25);
25   private Random rand = new Random();
26   private HTML html = new HTML();
27   private Button btn = new Button("Reset");
28   private FocusPanel fp = new FocusPanel(html);
29
30   public boolean isDrawn(int x, int y) {
31     return snakeList.contains(new Point(x,y));
32   }
33
34   public boolean isFood(int x, int y) {
35     return food.equals(new Point(x,y));
36   }
37
38   public void onModuleLoad() {
39     snakeList.add(new Point(2, 0));
40     snakeList.add(new Point(1, 0));
41     snakeList.add(new Point(0, 0));
42     fp.addKeyDownHandler(this);
43     btn.addClickHandler(this);
44     RootPanel.get().add(new HTML("<h1>Snake</h1>"));
45     RootPanel.get().add(fp);
46     RootPanel.get().add(btn);
47     fp.setFocus(true);
48     Timer t = new Timer() {
49       @Override
50       public void run() {
51         if(!lost) {
52           timer();
53           render();
54         }
55       }
56     };
57     render();
58     t.scheduleRepeating(100);
59   }
60
61   public void timer() {
62     if(lost)
63       return;
64     Point fp = snakeList.get(0).clone();
65     switch(keyCode) {
66     case 37: fp.x = fp.x - 1; break;
67     case 38: fp.y = fp.y - 1; break;
68     case 39:
69     default: fp.x = fp.x + 1; break;
70     case 40: fp.y = fp.y + 1; break;
71     }
72     if(fp.equals(food)) {
73       do {
74         int x = rand.nextInt(50);
75         int y = rand.nextInt(50);
76         food = new Point(x, y);
```

```
77        } while(snakeList.contains(food));
78      } else {
79        snakeList.remove(snakeList.size()-1);
80      }
81      if(snakeList.contains(fp) || fp.x < 0 || fp.x > 49 || fp.y < 0
            || fp.y > 49) {
82        lost = true;
83      } else {
84        snakeList.add(0, fp);
85      }
86    }
87
88    public void render() {
89      html.setHTML(getImage());
90    }
91
92    public String getImage() {
93      StringBuilder sb = new StringBuilder();
94      sb.append("<table␣cellspacing=\"0\"␣cellpadding=\"0\"␣style=\"
            border:␣2px␣solid␣black;\">");
95      for(int y=0; y<50; y++) {
96        sb.append("<tr>");
97        for(int x=0; x<50; x++) {
98          sb.append("<td>");
99          sb.append("<div␣style=\"width:␣5px;␣height:␣5px;␣
                background-color:␣").append(isDrawn(x, y)?"black":
                isFood(x,y)?"red":"white").append(";\"></div>");
100         sb.append("</td>");
101       }
102       sb.append("</tr>");
103     }
104     sb.append("</table>");
105     if(lost) {
106       sb.append("<div␣style=\"color:␣red;\">You␣lost!</div>");
107     }
108     return sb.toString();
109   }
110
111   public static class Point {
112     public Point clone() {
113       return new Point(x,y);
114     }
115     public Point(int x, int y) { this.x = x; this.y = y; }
116     public boolean equals(Object o) {
117       if( !(o instanceof Point)) {
118         return false;
119       }
120       Point p = (Point)o;
121       return p.x == x && p.y == y;
122     }
123     public int x;
124     public int y;
125   }
126
127   @Override
128   public void onKeyDown(KeyDownEvent event) {
```

```
129       keyCode = event.getNativeKeyCode();
130     }
131
132     @Override
133     public void onClick(ClickEvent event) {
134       keyCode = 39;
135       snakeList.clear();
136       snakeList.add(new Point(2, 0));
137       snakeList.add(new Point(1, 0));
138       snakeList.add(new Point(0, 0));
139       lost = false;
140       food = new Point(25, 25);
141       fp.setFocus(true);
142     }
143   }
```

Listing A145: Source code of GWT Snake class in JSF GWT comparison in category 5

Source code of SFB799 program

```
1  package de.tu_freiberg;
2
3  import java.io.PrintWriter;
4
5  import com.google.gwt.core.ext.Generator;
6  import com.google.gwt.core.ext.GeneratorContext;
7  import com.google.gwt.core.ext.TreeLogger;
8  import com.google.gwt.core.ext.TreeLogger.Type;
9  import com.google.gwt.core.ext.UnableToCompleteException;
10 import com.google.gwt.core.ext.typeinfo.JClassType;
11 import com.google.gwt.core.ext.typeinfo.NotFoundException;
12 import com.google.gwt.user.rebind.ClassSourceFileComposerFactory;
13 import com.google.gwt.user.rebind.SourceWriter;
14
15 import de.tu_freiberg.client.view.UnicodeCharView.Block;
16 import de.tu_freiberg.client.view.UnicodeCharView.UnicodeBlock;
17 import de.tu_freiberg.client.view.UnicodeCharView.UnicodeView;
18
19 public class UnicodeCharViewGenerator extends Generator {
20
21   @Override
22   /** function will build a new class that will implement
        getStatus() method
23    * @return name of the generated class in the given package
        structure,
24    * so that the compiler can use the generated class
25    * */
26   public String generate(TreeLogger logger, GeneratorContext
        context,
27       String typeName) throws UnableToCompleteException {
28
29     try {
30       SourceWriter sw = getSourceWriter(typeName, context, logger)
           ;
31       if(sw == null)
32         return typeName + "Generated"; // file already exists
33
34       // after we have all the information we need, we write our
           function in the already created class
35       sw.println("private␣Widget␣panel;");
36
37       sw.println("public␣UnicodeCharViewGenerated()␣{"); sw.
           indent();
38         //sw.println("panel = binder.createAndBindUi(this);");
39         sw.println("panel␣=␣createView();");
40         sw.println("initWidget(panel);");
41       sw.outdent(); sw.println("}");
42
43       sw.println("private␣com.google.gwt.user.client.EventListener
           ␣list;"); sw.indent();
44
45       sw.println("private␣void␣_registerHandlers(final␣de.
           tu_freiberg.client.event.ButtonTextClickedHandler␣p,␣com.
           google.gwt.user.client.Element␣cont)␣{"); sw.indent();
```

```
46          sw.println("for(int␣i␣=␣0;␣i<com.google.gwt.user.client.
                DOM.getChildCount(cont);i++)␣{");
47            sw.indent(); sw.println("com.google.gwt.user.client.
                  Element␣el␣=␣com.google.gwt.user.client.DOM.getChild(
                  cont,␣i);");
48            sw.println("if(el.getClassName().equals(\"gwt-Button\"))
                  ␣{"); sw.indent();
49              sw.println("com.google.gwt.user.client.DOM.sinkEvents(
                    el,␣Event.getTypeInt(ClickEvent.getType().getName()
                    )␣|␣com.google.gwt.user.client.DOM.getEventsSunk(el
                    ));");
50              sw.println("com.google.gwt.user.client.DOM.
                    setEventListener(el,␣list);");
51            sw.outdent(); sw.println("}␣else␣{");
52              sw.indent(); sw.println("_registerHandlers(p,␣el);");
53            sw.outdent(); sw.println("}");
54          sw.outdent(); sw.println("}");
55        sw.outdent(); sw.println("}");
56
57        sw.println("@Override");
58        sw.println("public␣void␣registerHandlers(final␣de.
              tu_freiberg.client.event.ButtonTextClickedHandler␣p)␣{");
               sw.indent();
59          sw.println("com.google.gwt.user.client.Element␣cont=panel.
                getElement();");
60            sw.println("list␣=␣new␣com.google.gwt.user.client.
                  EventListener()␣{"); sw.indent();
61            sw.println("@Override");
62            sw.println("public␣void␣onBrowserEvent(com.google.gwt.
                  user.client.Event␣event)␣{"); sw.indent();
63              sw.println("if(event.getTypeInt()␣==␣com.google.gwt.
                    user.client.Event.ONCLICK)␣{"); sw.indent();
64                sw.println("p.onButtonTextClicked(com.google.gwt.
                      user.client.Element.as(event.getEventTarget()).
                      getInnerHTML());");
65              sw.outdent(); sw.println("}");
66            sw.outdent(); sw.println("}");
67          sw.outdent(); sw.println("};");
68          sw.println("_registerHandlers(p,␣cont);");
69        sw.outdent(); sw.println("}");
70
71        createView(typeName, context, sw);
72        // if you forget it then the compiler cannot find the
              generated classes and you get
73        // errors like: Rebind result 'de.tu_freiberg.informatik.
              vonwenckstern.client.StatussafariGenerated' could not be
              found
74        sw.commit(logger);
75        System.out.println("class␣'" + typeName + "Generated'␣was␣
              created␣succesfully");
76        return typeName + "Generated";
77      } catch(Exception e) {
78        e.printStackTrace();
79        return null;
80      }
81    }
```

```
82
83    public void createView(String typeName, GeneratorContext context
          , SourceWriter sw) throws NotFoundException {
84      sw.println("public␣Widget␣createView()␣{"); sw.indent();
85      sw.println("com.google.gwt.user.client.ui.StackLayoutPanel␣st␣
            =␣new␣com.google.gwt.user.client.ui.StackLayoutPanel(com.
            google.gwt.dom.client.Style.Unit.PX);");
86      int i = 0;
87      for(JClassType type: context.getTypeOracle().getTypes()) {
88        UnicodeView unicodeView = type.getAnnotation(UnicodeView.
              class);
89        if(unicodeView != null) {
90          for(UnicodeBlock unicodeBlock: unicodeView.unicodeBlocks()
                ) {
91            StringBuilder sb = new StringBuilder();
92            for(Block block:unicodeBlock.uniCodeBlocks()) {
93              for(int unicode=block.start(); unicode <= block.end();
                    unicode++) {
94                sb.append("<button␣class='gwt-Button'>&#").append(
                      unicode).append(";</button>");
95              }
96            }
97            sw.println("st.add(new␣com.google.gwt.user.client.ui.
                  HTML(\"" + sb.toString() + "\"),␣\"" + unicodeBlock.
                  name() + "\",␣20);");
98            sw.println("DOM.setStyleAttribute(st.getHeaderWidget("+
                  i + ").getElement(),␣\"fontSize\",␣\"0.9em\");");
99            sw.println("DOM.setStyleAttribute(st.getWidget("+ i + ")
                  .getElement(),␣\"overflowY\",␣\"scroll\");");
100           i++;
101         }
102         break;
103       }
104     }
105     sw.println("return␣st;");
106     sw.outdent(); sw.println("}");
107   }
108
109   /** function returns the source writer where you can add the
          inner classes functions */
110   public SourceWriter getSourceWriter(String typeName,
          GeneratorContext context,
111       TreeLogger logger) throws NotFoundException {
112     // gets the type given by the String typeName
113     JClassType classType = context.getTypeOracle().getType(
            typeName);
114     // gets the package in which the new class should get created
115     String packageName = classType.getPackage().getName();
116     // gets the name of the class without the package name
117     String simpleName = classType.getSimpleSourceName();
118     // for us to see what classes were generated by this generator
119     simpleName = simpleName + "Generated";
120
121     // a composer factory which will create a new class in the
            given package with the given name
```

```
122     ClassSourceFileComposerFactory composer = new
            ClassSourceFileComposerFactory(packageName, simpleName);
123     // now we are adding the imports we need
124     composer.addImport("java.util.Iterator");
125     composer.addImport("com.google.gwt.core.client.GWT");
126     composer.addImport("com.google.gwt.event.dom.client.ClickEvent
            ");
127     composer.addImport("com.google.gwt.user.client.DOM");
128     composer.addImport("com.google.gwt.user.client.ui.Composite");
129     composer.addImport("com.google.gwt.user.client.Event");
130     composer.addImport("com.google.gwt.user.client.ui.Widget");
131     composer.addImport("de.tu_freiberg.client.presenter.
            UnicodeCharPresenter.Display");
132
133     // the class which we extend
134     composer.setSuperclass("Composite");
135     // now we are adding the implemented interface Status
136     composer.addImplementedInterface("Display");
137     composer.addImplementedInterface("UnicodeCharView");
138     // creates the source file in the given package with the given
            name
139     PrintWriter printWriter = context.tryCreate(logger,
            packageName, simpleName);
140     if(printWriter == null) {
141       logger.log(Type.SPAM, "printWriter␣is␣null");
142       return null;
143     } else {
144       // will create the class with all the given imports, extends
              and so and will write
145       // it to the source file created by printWriter
146       SourceWriter sw = composer.createSourceWriter(context,
              printWriter);
147       return sw;
148     }
149
150   }
151 }
```

Listing A146: Source code of UnicodeCharViewGenerator class

A 4 Tables

Table A3: Overview of all ASTNode types, this table lists the corresponding classes instead of the direct types

AnnotationType-Declaration	The thing to note is that method declaration are replaced by annotation type member declarations in this context. `AnnotationTypeDeclaration:` `  [ Javadoc ] { ExtendedModifier } @ interface Identifier` ` { { AnnotationTypeBodyDeclaration	; } }` `AnnotationTypeBodyDeclaration:` ` AnnotationTypeMemberDeclaration` ` FieldDeclaration` ` TypeDeclaration` ` EnumDeclaration` ` AnnotationTypeDeclaration`

AnnotationTypeMember-Declaration	Annotation type member declaration AST node type.
	`AnnotationTypeMemberDeclaration:`
	`  [ Javadoc ] { ExtendedModifier } Type Identifier ( ) [ default Expression ] ;`

AnonymousClass-Declaration	Anonymous class declaration AST node type. This type of node appears may also appear as the child of an enum constant declaration.
	`AnonymousClassDeclaration:`
	`  { ClassBodyDeclaration }`

ArrayAccess	Array access expression AST node type.
	`ArrayAccess:`
	`  Expression [ Expression ]`

ArrayCreation	Array creation expression AST node type.
	`ArrayCreation:`
	`  new PrimitiveType [Expression] { [Expression] } { [ ] }`
	`  new TypeName [ < Type { , Type } > ] [ Expression ] { [ Expression ] } { [ ] }`
	`  new PrimitiveType [ ] { [ ] } ArrayInitializer`
	`  new TypeName [ < Type { , Type } > ] [ ] { [ ] } ArrayInitializer`

ArrayInitializer	Array initializer AST node type.
	`ArrayInitializer:`
	`{ [ Expression { , Expression} [ , ]] }`

ArrayType	Type node for an array type. Array types are expressed in a recursive manner, one dimension at a time.
	`ArrayType:`
	`Type [ ]`

AssertStatement	Assert statement AST node type.
	`AssertStatement:`
	`assert Expression [ : Expression ] ;`

Assignment	Assignment expression AST node type.
	`Assignment:`
	`Expression AssignmentOperator Expression`

Block	Block statement AST node type.
	`Block:`
	`{ { Statement } }`

BlockComment	Block comments begin with /*, may contain line breaks, and must end with */.
BooleanLiteral	Boolean literal node: true, false

BreakStatement	Break statement AST node type.
	`BreakStatement:`
	`break [ Identifier ] ;`

CastExpression	Cast expression AST node type.
	`CastExpression:`
	`( Type ) Expression`

CatchClause	Catch clause AST node type.
	`CatchClause:`
	`catch ( FormalParameter ) Block`

CharacterLiteral	Character literal nodes.

ClassInstanceCreation	Class instance creation expression AST node type.
	`ClassInstanceCreation:`
	`[ Expression . ]`
	`new [ < Type { , Type } > ]`
	`Type ( [ Expression { , Expression } ] )`
	`[ AnonymousClassDeclaration ]`

CompilationUnit	Java compilation unit AST node type. This is the type of the root of an AST. The source range for this type of node is ordinarily the entire source file, including leading and trailing whitespace and comments.
	`CompilationUnit:`
	`[ PackageDeclaration ]`
	`{ ImportDeclaration }`
	`{ TypeDeclaration | EnumDeclaration | AnnotationTypeDeclaration | ; }`

ConditionalExpression	Conditional expression AST node type.
	`ConditionalExpression:`
	`Expression ? Expression : Expression`

ConstructorInvocation	Alternate constructor invocation statement AST node type.
	`ConstructorInvocation:`
	`[ < Type { , Type } > ]`
	`this ( [ Expression { , Expression } ] ) ;`

ContinueStatement	Continue statement AST node type.
	`ContinueStatement:`
	`continue [ Identifier ] ;`

DoStatement	Do statement AST node type.
	`DoStatement:`
	`do Statement while ( Expression ) ;`

EmptyStatement	Null statement AST node type.
	`EmptyStatement:`
	`;`

EnhancedForStatement	Enhanced For statement AST node type.
	`EnhancedForStatement:`
	`  for ( FormalParameter : Expression )`
	`  Statement`

EnumConstantDeclaration	Enumeration constant declaration AST node type.
	`EnumConstantDeclaration:`
	`  [ Javadoc ] { ExtendedModifier } Identifier`
	`  [ ( [ Expression { , Expression } ] ) ]`
	`  [ AnonymousClassDeclaration ]`

EnumDeclaration	Enum declaration AST node type.
	`EnumDeclaration:`
	`  [ Javadoc ] { ExtendedModifier } enum`
	`  Identifier`
	`  [ implements Type { , Type } ]`
	`  {`
	`  [ EnumConstantDeclaration { ,`
	`  EnumConstantDeclaration } ] [ , ]`
	`  [ ; { ClassBodyDeclaration | ; } ]`
	`  }`

ExpressionStatement	Expression statement AST node type. This kind of node is used to convert an expression (Expression) into a statement (Statement) by wrapping it.
	`ExpressionStatement:`
	`  StatementExpression ;`

FieldAccess	Field access expression AST node type.
	`FieldAccess:`
	`  Expression . Identifier`

FieldDeclaration	Field declaration node type. This kind of node collects several variable declaration fragments (VariableDeclarationFragment) into a single body declaration (BodyDeclaration), all sharing the same modifiers and base type.
	`FieldDeclaration:`
	`  [Javadoc] { ExtendedModifier } Type`
	`  VariableDeclarationFragment`
	`  { , VariableDeclarationFragment } ;`

ForStatement	For statement AST node type.
	`ForStatement:`
	`  for ( [ ForInit ]; [ Expression ] ; [ ForUpdate ] ) Statement`
	`ForInit:`
	`  Expression { , Expression }`
	`ForUpdate:`
	`  Expression { , Expression }`

IfStatement	If statement AST node type.
	`IfStatement:`
	`  if ( Expression ) Statement [ else Statement ]`

ImportDeclaration	Import declaration AST node type.
	`ImportDeclaration:`
	`  import [ static ] Name [ . * ] ;`

InfixExpression	Infix expression AST node type.
	`InfixExpression:`
	`  Expression InfixOperator Expression { InfixOperator Expression }`

Initializer	Static or instance initializer AST node type.
	`Initializer:`
	`  [ static ] Block`

InstanceofExpression	Instanceof expression AST node type.
	`InstanceofExpression:`
	`  Expression instanceof Type`

Javadoc	AST node for a Javadoc-style doc comment.
	`Javadoc:`
	`  /** { TagElement } */`

LabeledStatement	Labeled statement AST node type.
	`LabeledStatement:`
	`  Identifier : Statement`

LineComment	End-of-line comment AST node type. End-of-line comments begin with //, must end with a line delimiter, and must not contain line breaks.
MALFORMED	Flag constant (bit mask, value 1) indicating that there is something not quite right with this AST node.

MarkerAnnotation	Marker annotation node. The marker annotation @foo is equivalent to the normal annotation @foo().
	`MarkerAnnotation:`
	`  @ TypeName`

MemberRef	AST node for a member reference within a doc comment (Javadoc). The principal uses of these are in @see and @link tag elements, for references to field members (and occasionally to method and constructor members).
	`MemberRef:`
	`  [ Name ] # Identifier`

MemberValuePair	Member value pair node. Member value pairs appear in annotations.
	`MemberValuePair:`
	`  SimpleName = Expression`

MethodDeclaration	Method declaration AST node type. A method declaration is the union of a method declaration and a constructor declaration.
	`MethodDeclaration:`
	`  [ Javadoc ] { ExtendedModifier }`
	`  [ < TypeParameter { , TypeParameter } > ]`
	`  ( Type | void ) Identifier (`
	`  [ FormalParameter`
	`  { , FormalParameter } ] ) { [ ] }`
	`  [ throws TypeName { , TypeName } ] (`
	`  Block | ; )`
	
	`ConstructorDeclaration:`
	`  [ Javadoc ] { ExtendedModifier }`
	`  [ < TypeParameter { , TypeParameter } > ]`
	`  Identifier (`
	`  [ FormalParameter`
	`  { , FormalParameter } ]`
	`  )`
	`  [throws TypeName { , TypeName } ] Block`

MethodInvocation	Method invocation expression AST node type.
	`MethodInvocation:`
	`[ Expression . ]`
	`[ < Type { , Type } > ]`
	`Identifier ( [ Expression { , Expression } ] )`

MethodRef	AST node for a method or constructor reference within a doc comment (Javadoc). The principal uses of these are in "@see" and "@link" tag elements, for references to method and constructor members.
	`MethodRef:`
	`[ Name ] # Identifier`
	`( [MethodRefParameter { , MethodRefParameter }] )`

MethodRefParameter	AST node for a parameter within a method reference (MethodRef). These nodes only occur within doc comments (Javadoc).
	`MethodRefParameter:`
	`Type [ ... ] [ Identifier ]`

Modifier	Modifier node.
	`Modifier:`
	`public , protected , private , static , abstract , final , native , synchronized , transient , volatile , strictfp`

NormalAnnotation	Normal annotation node.
	`NormalAnnotation:`
	`@ TypeName ( [ MemberValuePair { , MemberValuePair } ] )`

NumberLiteral	Number literal nodes.
ORIGINAL	Flag constant (bit mask, value 2) indicating that this is a node that was created by the parser (as opposed to one created by another party).

PackageDeclaration	Package declaration AST node type.
	`PackageDeclaration:`
	`[ Javadoc ] { Annotation } package Name ;`

ParameterizedType	Type node for a parameterized type. These nodes are used for type references (as opposed to declarations of parameterized types.)
	`ParameterizedType:`
	`  Type < Type { , Type } >`

ParenthesizedExpression	Parenthesized expression AST node type.
	`ParenthesizedExpression:`
	`  ( Expression )`

PostfixExpression	Postfix expression AST node type.
	`PostfixExpression:`
	`  Expression PostfixOperator`

PrefixExpression	Prefix expression AST node type.
	`PrefixExpression:`
	`  PrefixOperator Expression`

PrimitiveType	Primitive type nodes.
	`PrimitiveType:`
	`  byte, short, char, int, long, float, double, boolean, void`

PROTECT	Flag constant (bit mask, value 4) indicating that this node is unmodifiable.
QualifiedName	AST node for a qualified name. A qualified name is defined recursively as a simple name preceded by a name, which qualifies it. Expressing it this way means that the qualifier and the simple name get their own AST nodes. QualifiedName: Name . SimpleName

QualifiedType	Type node for a qualified type.
	`QualifiedType:`
	`  Type . SimpleName`

RECOVERED	Flag constant (bit mask, value 8) indicating that this node or a part of this node is recovered from source that contains a syntax error detected in the vicinity.

ReturnStatement	Return statement AST node type.
	`ReturnStatement:`
	`  return [ Expression ] ;`

SimpleName	AST node for a simple name. A simple name is an identifier other than a keyword, boolean literal ("true", "false") or null literal ("null").
	`SimpleName:`
	`  Identifier`

SimpleType	Type node for a named class type, a named interface type, or a type variable. This kind of node is used to convert a name (Name) into a type (Type) by wrapping it.

SingleMemberAnnotation	Single member annotation node. The single member annotation "@foo(bar)" is equivalent to the normal annotation "@foo(value=bar)".
	`SingleMemberAnnotation:`
	`  @ TypeName ( Expression )`

SingleVariableDeclaration	Single variable declaration AST node type. Single variable declaration nodes are used in a limited number of places, including formal parameter lists and catch clauses. They are not used for field declarations and regular variable declaration statements.
	`SingleVariableDeclaration:`
	`  { ExtendedModifier } Type [ ... ] Identifier`
	`  { [] } [ = Expression ]`

StringLiteral	String literal nodes.

SuperConstructor-Invocation	Super constructor invocation statement AST node type.
	`SuperConstructorInvocation:`
	`  [ Expression . ]`
	`  [ < Type { , Type } > ]`
	`  super ( [ Expression { , Expression } ]`
	`  ) ;`

SuperFieldAccess	Simple or qualified "super" field access expression AST node type.
	`SuperFieldAccess:`
	`  [ ClassName . ] super . Identifier`

SuperMethodInvocation	Simple or qualified "super" method invocation expression AST node type.
	`SuperMethodInvocation:`
	`  [ ClassName . ] super .`
	`  [ < Type { , Type } > ]`
	`  Identifier ( [ Expression { ,`
	` Expression }])

SwitchCase	Switch case AST node type. A switch case is a special kind of node used only in switch statements. It is a Statement in name only. SwitchCase: **case** Expression : default :
SwitchStatement	Switch statement AST node type. SwitchStatement: switch (Expression) { { *SwitchCase* | *Statement* } } } SwitchCase: **case** Expression : default :
SynchronizedStatement	Synchronized statement AST node type. SynchronizedStatement: synchronized (Expression) Block
TagElement	AST node for a tag within a doc comment. Tag elements nested within another tag element are called inline doc tags. TagElement: [@ Identifier] { *DocElement* } DocElement: TextElement Name MethodRef MemberRef { *TagElement* }
TextElement	AST node for a text element within a doc comment. TextElement: Sequence **of** characters **not** including a close comment delimiter */
ThisExpression	Simple or qualified "this" AST node type. ThisExpression: [ClassName .] this
ThrowStatement	Throw statement AST node type. ThrowStatement: throw Expression ;

TryStatement	Try statement AST node type.
	`TryStatement:`
	`  try Block`
	`  { CatchClause }`
	`  [ finally Block ]`

TypeDeclaration	Type declaration AST node type. A type declaration is the union of a class declaration and an interface declaration. `TypeDeclaration:` `  ClassDeclaration` `  InterfaceDeclaration` `ClassDeclaration:` `  [ Javadoc ] { ExtendedModifier } class Identifier` `  [ < TypeParameter { , TypeParameter } > ]` `  [ extends Type ]` `  [ implements Type { , Type } ]` ` { { ClassBodyDeclaration	; } }` `InterfaceDeclaration:` ` [Javadoc] { ExtendedModifier } interface Identifier` ` [< TypeParameter { , TypeParameter } >]` ` [extends Type { , Type }]` ` { { InterfaceBodyDeclaration	; } }`

TypeDeclarationStatement	Local type declaration statement AST node type. This kind of node is used to convert a type declaration node into a statement node by wrapping it.
	`TypeDeclarationStatement:`
	`  TypeDeclaration`
	`  EnumDeclaration`

TypeLiteral	Type literal AST node type.
	`TypeLiteral:`
	`  ( Type | void ) . class`

TypeParameter	Type parameter node.
	`TypeParameter:`
	`  TypeVariable [ extends Type { \& Type } ]`

VariableDeclaration-Expression	Local variable declaration expression AST node type. This kind of node collects together several variable declaration fragments (VariableDeclarationFragment) into a single expression (Expression), all sharing the same modifiers and base type. This type of node can be used as the initializer of a ForStatement, or wrapped in an ExpressionStatement to form the equivalent of a VariableDeclarationStatement.
	`VariableDeclarationExpression:`
	`{ ExtendedModifier } Type`
	`VariableDeclarationFragment`
	`{ , VariableDeclarationFragment }`

VariableDeclaration-Fragment	Variable declaration fragment AST node type, used in field declarations, local variable declarations, and ForStatement initializers. It contrast to SingleVariableDeclaration, fragments are missing the modifiers and the type; these are located in the fragment's parent node.
	`VariableDeclarationFragment:`
	`Identifier { [] } [ = Expression ]`

VariableDeclaration-Statement	Variable declaration fragment AST node type, used in field declarations, local variable declarations, and ForStatement initializers. It contrast to SingleVariableDeclaration, fragments are missing the modifiers and the type; these are located in the fragment's parent node.
	`VariableDeclarationFragment:`
	`Identifier { [] } [ = Expression ]`

Variable declaration fragment	AST node type, used in field declarations, local variable declarations, and ForStatement initializers. It contrast to SingleVariableDeclaration, fragments are missing the modifiers and the type; these are located in the fragment's parent node.
	`VariableDeclarationFragment:`
	`Identifier { [] } [ = Expression ]`

WhileStatement	While statement AST node type.
	`WhileStatement:`
	`while ( Expression ) Statement`

WildcardType	Type node for a wildcard type.
	`WildcardType:`
	`? [ ( extends | super) Type ]`

R Lists and References

R 1 Lists

R 1.1 List of Tables

1 Most important HTML tags 4
2 Overview of GWT toolbox 13
3 Passing JavaScript values into Java code 15
4 Passing Java values into JavaScript 16
5 Parameter signature of most common Java types 16
6 Dead code elimination grouped in different steps 33
7 Examples to show the difference of String::match in the Java vs. JavaScript world. 39
8 HTTP status codes 58
9 GWT application's URL of different program states 62
10 Steps how to create a new Server in Eclipse 119
11 Dynamic Web Project wizard part 1. 119
12 Dynamic Web Project wizard part 2. 120
13 GWT module properties 120
14 Wizard Run Configurations 121
15 Wizard showing up when Tomcat is started the first time. 121
16 Wizard Adding new user 126
17 Wizard creating New Connection 127
18 Example how to inject MySQL code into the unsafe function 131
A1 Add GWT Plugin repository. . . . . A 2
A2 Add GWT Designer Plugin repository. . . . . A 3
A3 Overview of all ASTNode types, this table lists the corresponding classes instead of the direct types . . . . . A 310

R 1.2 List of Figures

1 CSS example . . . 6
2 Simple JavaScript calculator example . . . 7
3 HTML DOM of listing 3 . . . 9
4 Files the compiler generated . . . 19
5 Loading sequence of a website . . . 24
6 General steps of the GWT compiler . . . 28
7 General steps of pre compilation process . . . 29
8 General steps of one compilation permutation . . . 30
9 Optimization order in optimizeLoop . . . 32
10 Control flow graph of liveness analysis . . . 37
11 Simple example code for using no JREE class . . . 38
12 Widget class hierarchy . . . 40
13 Label widget class hierarchy . . . 41
14 FocusWidget widget class hierarchy . . . 42
15 DateBox composite contains TextBox, PopupPanel and DatePicker widgets 43
16 Composite widget class hierarchy . . . 43
17 The own Composite widget . . . 43
18 Panel widget class hierarchy . . . 45
19 General handler interfaces . . . 48
20 Special handler interfaces . . . 49
21 Calls and messages of an invoked Remote Procedure Call . . . 52
22 HTTP post result manipulated in IE debugger . . . 62
23 Screenshot of CSS transition navigation menu . . . 69
24 ImageResource as background picture . . . 74
25 Model-View-Controller pattern . . . 75
26 MVC pattern for server intensive web applications . . . 76
27 Taligent Model-View-Presenter pattern . . . 77
28 Dolphin Smalltalk Model-View-Presenter pattern . . . 78
29 Screenshot of Agricola board game . . . 79
30 Mobile view of Agricola board game . . . 89
31 Screenshot of JSF version in category 1 . . . 100
32 Screenshot of Topic.jsf page. . . . 107
33 Screenshot of JSF chat application . . . 111
34 Points earned by GWT and JSF in categories 1 to 5. . . . 118
35 Servers directory in Eclipse . . . 119
36 Screenshot how to compile the GWT project . . . 120
37 Screenshot of build path configuration . . . 122
38 Screenshot showing how to terminate the Development Mode. . . . 122
39 Screenshot of Tomcat's publishing settings . . . 123
40 Screenshot showing how to open the TomcatGWT connection in the MySQL Workbench . . . 126
41 Schema Privileges of TomcatGWT user . . . 127
42 Screenshot of remote database result. . . . 130
43 MySQL injection shows secret database login . . . 131
44 Safe function prevents MySQL injections . . . 131
45 Screenshot of login formula . . . 135
46 Screenshot of Tomcat's session id in Firefox . . . 135
47 Screenshot of final result with logout button. . . . 135

48 HTML injections in unsafe text insert method, no injection by usage of SafeHtml . . . 137
49 Screenshot of old Java version of the SFB799 database client . . . 138
50 Screenshot showing the extended TabPanel allowing to undock TabItems . 139
A1 Download Eclipse IDE for Java EE Developers. . . . A 1
A2 Screenshot of Eclipse file structure and dialog of Workspace Launcher. . . . A 2
A3 Selected entries of the GWT Eclipse plugin, which should be installed. . . . A 2
A4 Selected entries of the GWT Designer Eclipse plugin, which should be installed A 3
A5 Properties of GWT project . . . A 4
A6 Create GWT module in FirstProject . . . A 4
A7 Right click at FirstProject opens popup menu. The menu entry Run As -> Web Application should be selected. . . . A 5
A8 Screenshot of how to install the GWT Developer Plugin in Firefox 20 . . . A 5
A9 Left click at the right icon terminates the web application. . . . A 5
A10 Double click at the left frame in the source code window creates a breakpoint, which is displayed as blue circle. . . . A 6
A11 Screenshot showing how to compile the GWT project . . . A 6
A12 War folder contains the result of the compilation process. . . . A 6
A13 First website . . . A 7
A14 Screenshot of the homepage of the Technische Universität Freiberg . . . A 7
A15 Screenshot of ECMAScript Language test262 with Internet Explorer 10 . . . A 8
A16 Screenshot of ECMAScript Language test262 with Chrome 26 . . . A 9
A17 Screenshot of ECMAScript Language test262 with Firefox 20 . . . A 10
A18 Screenshot of DOM manipulating demonstration. . . . A 11
A19 Screenshot of AJAX example, part I. . . . A 12
A20 Screenshot of AJAX example, part II. . . . A 13
A21 Screenshot of the web interface to paint pictures. . . . A 13
A22 Screenshot of GWT showcase testing Stack Panel component . . . A 14
A23 Screenshot of GXT showcase testing advanced toolbar component . . . A 15
A24 Screenshot of GXT showcase testing border layout . . . A 16
A25 Screenshot of GXT desktop showcase . . . A 17
A26 Screenshot of youtube video playing Quake 2 in the web browser . . . A 17
A27 Screenshot of youtube video testing mgwt showcase on an iPhone simulator . A 18
A28 Picture showing interactions between GWT and Google Maps API V3 . . . A 18
A29 Screenshot of generated website by Eclipse . . . A 18
A30 Screenshot of type conversion in JavaScript . . . A 18
A31 Screenshot of JSNI example . . . A 19
A32 Screenshot testing wrapped syntaxhighlighter JavaScript library in GWT . . . A 19
A33 Screenshot testing deferred binding with replacement . . . A 20
A34 Screenshot testing deferred binding with generator . . . A 20
A35 Eclipse Java AST of listing A20, part I. . . . A 21
A36 Eclipse Java AST of listing A20, part II. . . . A 22
A37 Eclipse Java AST of listing A20, part III. . . . A 23
A38 Eclipse Java AST of listing A20, part IV. . . . A 24
A39 Inheritance pattern for AST . . . A 25
A40 Double dispatch pattern for AST . . . A 25
A41 Example how to use Eclipse AST library together with visitor pattern . . . A 26
A42 CompilerOptions class attributes . . . A 27
A43 UML diagram of Java Runtime Environment Emulation (JREE) . . . A 28
A44 UML diagram of java.lang package of JREE . . . A 29

A45 UML diagram of java.util package of JREE . . . . . A 30
A46 Operation interfaces of GWT widget hierarchy . . . . . A 31
A47 Screenshot of FocusWidget example. . . . . A 32
A48 Screenshot of Composite example, part I. . . . . A 33
A49 Screenshot of Composite example, part II. . . . . A 34
A50 Screenshot of Panel example. . . . . A 35
A51 Screenshot of EventHandler example. . . . . A 36
A52 Screenshot of DOMManipulation example. . . . . A 36
A53 Screenshot of creating view in GWT Designer . . . . . A 37
A54 Screenshot of UiBinder example. . . . . A 37
A55 Screenshot of RMI example . . . . . A 38
A56 Screenshot of Survey example, first page . . . . . A 38
A57 Screenshot of Survey example, second page . . . . . A 39
A58 Screenshot of Survey example, third page . . . . . A 40
A59 Screenshot of Survey example after back button has been clicked . . . . . A 40
A60 Screenshot of Firefox when it has to load all LaTeX icons from the server . . A 41
A61 Screenshot of Firefox when it shows the LaTeX icons from the inlined ClientBundle interface . . . . . A 41
A62 Screenshot of Firefox when it shows the LaTeX icons from the ClientBundle interface with disabled inlining option. . . . . A 42
A63 Screenshot of IE 10 without the usage of conditional CSS . . . . . A 42
A64 Screenshot of IE 10 with the usage of conditional CSS . . . . . A 42
A65 Screenshot of Google Chrome with the usage of conditional CSS . . . . . A 43
A66 Screenshot of Firefox with the usage of conditional CSS . . . . . A 43
A67 Model classes . . . . . A 44
A68 Presenter classes . . . . . A 45
A69 UML diagram displaying the AppController's links to the other presenters . A 46
A70 Screenshot illustrating the topic entries of "Comments on: You'll miss me when I've gone" . . . . . A 47
A71 Screenshot of the new SFB799 web application . . . . . A 48
A72 Screenshot of Microsoft Office specific style definitions in the help file . . . . A 48
A73 Screenshot of widget which gets generated depending on the used Java annotations. . . . . A 49

R 1.3 List of Listings

1 CSS example . . . 7
2 Simple JavaScript calculator example . . . 8
3 Simple HTML code for DOM model . . . 8
4 HTML code of DrawDomExample.html (line 21). . . . 10
5 HTML code of DrawDomExample.html (lines 30-41). . . . 10
6 HTML code of DrawDomExample.html (lines 90, 95, 109, 111, 121). . . . 10
7 HTML code of DrawDomExample.html. (lines 147-151) . . . 11
8 Excerpt from auto generated example source code by Eclipse . . . 14
9 JSNI implementation of window.alert . . . 14
10 Excerpt from code showing typing conversion in JavaScript . . . 15
11 Excerpt from ImageViewer.html . . . 17
13 Example code using syntaxhighlighter_3.0.83 library . . . 17
14 Corresponding JavaScript code of the HTML code in listing 13 . . . 17
15 SyntaxHighlighterImpl.java in LoadJavaScript library's project . . . 18
16 Example JavaScript code choosing the right language . . . 20
17 Status.java in deferred binding with replacement project . . . 20
18 StatusLabelIE9.java in deferred binding with replacement project . . . 20
19 StatusLabelFF.java in deferred binding with replacement project . . . 20
20 ImageViewer.gwt.xml in deferred binding with replacement project . . . 21
21 ImageViewer.gwt.xml in deferred binding with generator project . . . 22
22 StatusGenerator.java in deferred binding with generator project . . . 22
23 Compiler output in deferred binding with generator project . . . 23
24 HTML example describing the order of loading a web page . . . 23
25 Prototype implementation of the accept method . . . 27
26 Main.java in AST project . . . 27
27 ASTRenamer.java in AST project . . . 28
28 Multi-catch block . . . 30
29 Single-catch block . . . 30
30 Uncombined case labels . . . 31
31 Combined case labels . . . 31
32 Same parameter value optimizer: **before** . . . 34
33 Same parameter value optimizer: **after** . . . 34
34 Enum ordinalizer optimizer: **before** . . . 34
35 Enum ordinalizer optimizer: **after** . . . 34
36 Simple dataflow analysis example . . . 34
37 Source code of UnreachableAnalysis.class . . . 35
38 Source code of UnreachabeIntegratedTransformationFunction.class . . . 36
39 Simple copy analysis example . . . 37
40 Simple example code for liveness analysis . . . 37
41 Simple example code for using no JREE class . . . 38
42 Source code of a quote function for regular expressions in JS and Java . . . 39
43 Example project for own login composite widget . . . 44
44 Very simple example of registering a ClickHandler . . . 47
45 Example how to handle click events of many buttons . . . 47
46 Manipulating the widget's border style with GWT Style class . . . 49
47 Manipulating the widget's border style with GWT DOM class . . . 49
48 ListBox in XML . . . 51
49 ListBox in Java . . . 51

50 Styled HTML in XML . . . 51
51 Styled HTML in Java . . . 51
52 Example how to get access to the widgets defined in the XML . . . 52
53 Excerption of RmiClient.java . . . 54
54 Loading car informations using synchronous RMI calls . . . 54
55 Loading car informations using asynchronous RMI calls . . . 55
56 Generated CalendarService.java file by Eclipse . . . 55
57 Generated CalendarServiceAsync.java file by Eclipse . . . 56
58 Generated CalendarServiceImpl.java file by Eclipse . . . 56
59 Modified web.xml file by Eclipse . . . 56
60 Implementation of CalendarServiceImpl's get method . . . 57
61 Invoking remote calls to collect more date information . . . 57
62 CalendarServiceImpl's get method throws exceptions now . . . 58
63 Source code of Result.java . . . 59
64 Using the abstract Result class instead of AsyncCallback . . . 60
67 Source code of Survey.java (lines 129-141) . . . 63
68 Source code of Survey.java (lines 175-181) . . . 63
69 Source code of Survey.java (lines 189-205) . . . 64
70 Java code of IconsURL class . . . 65
71 Java code of IconsClientBundle class . . . 65
72 Java code of Images interface . . . 66
73 Java code of IconsClientBundle class (ClientBundle.enableInlining=false) 67
74 Java code of ScalableImage class . . . 68
75 Example code how to use the ScalableImage class. . . . 68
76 CSS code of transitionNavigation.css file . . . 70
77 Java code of TransitionNavigation interface. . . . 71
78 HTML code of navigation.html text file . . . 71
79 Java code of Resources interface. . . . 71
80 Java code of CSS entry point class. . . . 72
81 CSS parameterized with menuWidth and menuWidthInner . . . 72
82 Saving CSS parameter in Java file . . . 72
83 Using conditional CSS to deliver only browser-specific CSS . . . 73
84 Adding ImageResource function bgImage() to Resource interface. . . . 73
85 Adding ResourceImage object into CSS file . . . 73
86 UiChild annotation in TooltipImage . . . 81
87 Call the addTooltip method in any UiBinder file . . . 82
88 Code example how to create a model in the UiBinder . . . 82
89 BigAcquisitionsPresenter class . . . 84
90 Source code of registerHandlers and hideAcquisition implementations . . . 85
91 Source code of classes GetBigAcquisitionEvent, GetBigAcquisitionHandler, and HasGetBigAcquisitionHandler . . . 85
92 Source code of EventBus class. . . . 87
93 Source code of onGetBigAcquisition method in the AppController class. . . 88
94 Source code of registerHandlers method in Rounds1To7View.java file. . . . 90
95 Source code of registerHandlers method in PlayerField.java file. . . . 91
96 Source code of ViewFactory class. . . . 91
97 Source code parts of DesktopViewFactory class. . . . 92
98 Source code fragments of MobileViewFactory class. . . . 92
99 Source code showing how to create presenters in AppController class. . . . 93
100 Source code defining the deferred binding for the different ViewFactories . 93

101 Source code of HistoryController class . . . 95
102 Source code showing the onSaveHistoryToURL . . . 96
103 Beginning of the serialized historyMap string. . . . 96
104 Shortened and encoded URL string . . . 96
105 Source code of onValueChange method of the HistoryController class. . . . 97
106 Source code of PlayerResourceModel class . . . 98
107 Source code of InfoViewPresenter class. . . . 99
108 XML code of news part in the main.xhtml file . . . 101
109 XML code generating the majors table in the majors.xhtml file . . . 102
110 Java code of onValueChange function in InfoSite.java file . . . 103
111 Java code Majors widget. . . . 104
112 Java code of add method in the Survey.java file. . . . 104
113 XML code of body part in page1.xhtml file . . . 105
114 XML code of body part in page2.xhtml file . . . 106
115 Java code of the openNewTopic function in the Forum class. . . . 108
116 Java code of the addNewEntry function in the Forum class. . . . 108
117 XML code of body part in topics.xhtml file . . . 109
118 Java code excerption of the onValueChange function in the Forum class. . . . 110
119 Java code of the send and getUpdate methods of Chat class . . . 112
120 Java source code of the login function in the Chat class. . . . 112
121 XML code chatroom.xhtml file . . . 113
122 JavaScript code in jsf.js. . . . 114
123 Java code excerption of GWT Chat class . . . 115
124 XML code of snake.xhtml file . . . 116
125 DOS command to create a 2048 Bit RSA key . . . 123
127 DOS command to sign the certificate. . . . 123
126 DOS command to create the signing request . . . 124
128 DOS command converting pem certificate. . . . 124
129 DOS command how to execute ImportKey program. . . . 124
130 DOS command how to import the chain certificate into the key store. . . . 125
131 DOS command how to change the password of the key store. . . . 125
132 XML Connector code in server.xml . . . 125
133 Enabling HTTP:8080 redirect to HTTPS:8443. . . . 125
134 Forcing the web application to use only HTTPS connections . . . 126
135 XML code connecting Tomcat with the database . . . 128
136 Context link to the database server resource. . . . 128
137 Java code requesting all countries in the database in the onModuleLoad function . . . 129
138 Java code of getConnection function . . . 129
139 Java code of getCountries remote implementation. . . . 129
140 Java code of unsafe function. . . . 130
141 Java code of safe function. . . . 131
142 MySQL code to create the login tables . . . 132
143 XML code registering users in Tomcat . . . 133
144 XML code granting only access to users with the role "user" . . . 133
145 XML code setting up form based login mechanism . . . 133
146 HTML and Java code of login.jsp file . . . 134
147 Java code of getLoginName and logout functions. . . . 134
148 HTML injection code . . . 136
149 Java code with and without the usage of SafeHtml. . . . 136

150 Java code showing paste and copy widgets . . . 140
151 Java code of setCopyHTML method. . . . 140
152 Java code of onKeyDown function in the TableView::TableController class. 140
153 Java code of copyDataToClipboard function . . . 141
154 Source code showing how to define Java annotations. . . . 141
155 Source code showing how to use the self created annotations. . . . 142
156 XML code defining the generator class which should handle the annotations 142
157 Source code of UnicodeCharViewGenerator class . . . 143
A1 Java installation succeeded. . . . A 1
A2 Java installation failed. . . . A 1
A3 HTML code of first website . . . A 50
A4 HTML code of DrawDomExample.html . . . A 51
A5 HTML code of AJAX.html . . . A 56
A6 Auto generated example source code by Eclipse . . . A 57
A7 Code from de.tu_freiberg.informatik.vonwenckstern.ImageViewer.nocache.js at line 2 (excerption) . . . A 58
A8 Code from de.tu_freiberg.informatik.vonwenckstern.ImageViewer.nocache.js A 58
A9 Code showing typing conversion in JavaScript . . . A 58
A10 JSNI example code . . . A 59
A11 Code generated by the GWT compiler in obfuscated mode of the JSNI function . . . A 62
A12 ImageViewer.html in LoadJavaScript library's project . . . A 63
A13 SyntaxHighlighter.java in LoadJavaScript library's project . . . A 64
A14 SyntaxHighlighterWidget.java in LoadJavaScript library's project . . . A 65
A15 ImageViewer.java in LoadJavaScript library's project . . . A 66
A16 ImageViewer.java in deferred binding with replacement project . . . A 67
A17 StatusLabel.java in deferred binding with replacement project . . . A 67
A18 StatusGenerator.java in deferred binding with generator project . . . A 68
A19 Compiler output in deferred binding with generator project . . . A 71
A20 ImageViewer.java in compiler optimization project . . . A 71
A21 Main.java (top code with line numbers) and console output (bottom display wihtout line numbers) in dispatch project (shape example). . . . A 73
A22 Main.java (top code with line numbers) and console output (bottom display without line numbers) in visitor pattern project (shape example). . . . A 75
A23 Output and configuration of AST project . . . A 77
A24 ASTPrinter.java in AST project . . . A 78
A25 ASTOptimizer.java in AST project . . . A 80
A26 ASTRenamer.java in AST project . . . A 83
A27 ImageViewer.java in JREE calendar project . . . A 84
A28 Today.java in JREE calendar project . . . A 84
A29 Example project showing all extended classes of FocusWidget . . . A 85
A30 Example project showing nearly all extended classes of Composite . . . A 87
A31 Example project showing nearly all extended classes of Panel . . . A 91
A32 Example project showing different EventHandler implementations . . . A 94
A33 DOMManipulation.java . . . A 101
A34 UiBinderExample.ui.xml . . . A 102
A35 RMI example . . . A 105
A36 Source code of Survey.java . . . A 107
A37 Source code of URLlength class . . . A 112
A38 Agricola.java file . . . A 113

A39 AppController.java file . . . . . A 113
A40 AppView.java file . . . . . A 123
A41 AppView.ui.xml file . . . . . A 124
A42 AppViewMobile.java file . . . . . A 126
A43 AppViewMobile.ui.xml file . . . . . A 127
A44 EventBus.java file . . . . . A 128
A45 HistoryController.java . . . . . A 131
A46 Utils.java file . . . . . A 134
A47 AddResourceEvent.java file . . . . . A 135
A48 AcquisitionCardModel.java file . . . . . A 137
A49 BackgroundCard.java file . . . . . A 138
A50 BaseFieldModel.java file . . . . . A 138
A51 BigAcquisitationsModel.java file . . . . . A 139
A52 BigAcquisitions.java file . . . . . A 140
A53 BigFieldModel.java file . . . . . A 140
A54 CardFieldModel.java file . . . . . A 141
A55 Child.java file . . . . . A 144
A56 FieldCard.java file . . . . . A 144
A57 HasAcquisitionCardModel.java file . . . . . A 144
A58 HasBaseFieldModel.java file . . . . . A 144
A59 HistoryMap.java file . . . . . A 144
A60 HistoryMap_CustomFieldSerializer.java file . . . . . A 145
A61 Player.java file . . . . . A 145
A62 PlayerFieldModel.java file . . . . . A 145
A63 PlayerResourceModel.java file . . . . . A 156
A64 Resource.java file . . . . . A 159
A65 Rounds1To7Model.java file . . . . . A 159
A66 Rounds8To14Model.java file . . . . . A 162
A67 SmallFieldModel.java file . . . . . A 164
A68 Activity.java file . . . . . A 166
A69 BigAcquisitionsPresenter.java file . . . . . A 167
A70 CardFieldPresenter.java file . . . . . A 168
A71 InfoViewPresenter.java file . . . . . A 170
A72 PlayerFieldPresenter.java file . . . . . A 171
A73 Presenter.java file . . . . . A 188
A74 ResourcePresenter.java file . . . . . A 188
A75 Rounds1To7Presenter.java file . . . . . A 191
A76 Rounds8To14Presenter.java file . . . . . A 193
A77 Images.java file . . . . . A 195
A78 DesktopViewFactory.java file . . . . . A 199
A79 HasPosition.java file . . . . . A 200
A80 MobileViewFactory.java file . . . . . A 200
A81 Renderer.java file . . . . . A 201
A82 Tooltip.java file . . . . . A 202
A83 ViewFactory.java file . . . . . A 203
A84 AcquisitionCardRenderer.java file . . . . . A 204
A85 BigAcquisitionsFieldView.java file . . . . . A 205
A86 BigAcquisitionsFieldView.ui.xml file . . . . . A 206
A87 BigFieldRenderer.java file . . . . . A 210
A88 CardFieldView.java file . . . . . A 212

A89 CardFieldView.ui.xml file . . . A 213
A90 ChildRenderer.java file . . . A 215
A91 InfoView.java file . . . A 216
A92 CardFieldView.ui.xml file . . . A 218
A93 PlayerFieldView.java file . . . A 220
A94 PlayerFieldView.ui.xml file . . . A 221
A95 ResChildRenderer.java file . . . A 225
A96 ResourceRenderer.java file . . . A 226
A97 Rounds1To7View.java file . . . A 227
A98 Rounds1To7View.ui.xml file . . . A 228
A99 Rounds8To14View.java file . . . A 230
A100 Rounds8To14View.ui.xml file . . . A 231
A101 SmallFieldRenderer.java file . . . A 233
A102 TooltipImage.java file . . . A 235
A103 AcquisitionCardRenderer.java file . . . A 236
A104 BigAcquisitionsFieldView.java file . . . A 237
A105 BigAcquisitionsFieldView.ui.xml file . . . A 238
A106 BigFieldRenderer.java file . . . A 241
A107 CardFieldView.java file . . . A 242
A108 CardFieldView.ui.xml file . . . A 243
A109 ChildRenderer.java file . . . A 246
A110 InfoView.java file . . . A 246
A111 InfoView.ui.xml file . . . A 248
A112 LabelAcquisitionRenderer.java file . . . A 250
A113 PlayerFieldView.java file . . . A 251
A114 PlayerFieldView.ui.xml file . . . A 253
A115 ResChildRenderer.java file . . . A 257
A116 ResourceRenderer.java file . . . A 257
A117 Rounds1To7View.java file . . . A 259
A118 Rounds1To7View.ui.xml file . . . A 260
A119 Rounds8To14View.java file . . . A 262
A120 Rounds8To14View.ui.xml file . . . A 263
A121 SmallFieldRenderer.java file . . . A 264
A122 TooltipPanel.java file . . . A 267
A123 TooltipPanelChildRenderer.java file . . . A 267
A124 Source code of Info class (category 1) . . . A 269
A125 Complete source code of main.xhtml (category 1) . . . A 270
A126 Complete source code of majors.xhtml (category 1) . . . A 271
A127 Source code of InfoSite class (category 1) . . . A 271
A128 Source code of UniversityNews class (category 1) . . . A 273
A129 Source code of Survey managed bean class (category 2) . . . A 275
A130 XML code of page3.xhtml (category 2) . . . A 278
A131 Source code of Forum managed bean class (category 3) . . . A 278
A132 XML code of showtopic.xhtml (category 3) . . . A 282
A133 Source code of Forum GWT class (category 3) . . . A 282
A134 Source code of ForumService RPC interface (category 3) . . . A 286
A135 Source code of TopicInformation class (category 3) . . . A 287
A136 Source code of ForumServiceImpl RPC server class (category 3) . . . A 287
A137 Source code of Chat managed bean class (category 4) . . . A 290
A138 Source code of User managed bean class (category 4) . . . A 292

A139 Source code of Message managed bean class (category 4) . . . . . A 293
A140 XML code of login.xhtml (category 4) . . . . . A 294
A141 Source code of Chat GWT class (category 4) . . . . . A 294
A142 Source code of ChatService RPC interface (category 4) . . . . . A 298
A143 Source code of ChatServiceImpl RPC server class (category 4) . . . . . A 298
A144 Source code of Snake managed bean class (category 5) . . . . . A 300
A145 Source code of GWT Snake class (category 5) . . . . . A 302
A146 Source code of UnicodeCharViewGenerator class . . . . . A 306

R 2 References

R 2.1 Books

[Alb07] Tom Alby. *Web 2.0: Konzepte, Anwendungen, Technologien : [ajax, api, atom, blog, folksonomy, feeds, long tail, mashup permalink, podcast, rich user experience, rss, social software, tagging]*. München [u.a.]: Hanser, 2007. ISBN: 3-446-40931-9.

[FK00] Duane K. Fields and Mark A. Kolb. *Web development with JavaServer pages*. Greenwich and CT: Manning Publications, 2000. ISBN: 1-884777-99-6.

[HT07] Robert Hanson and Adam Tacy. *GWT in action: Easy Ajax with the Google Web toolkit*. Greenwich and CT: Manning, 2007. ISBN: 1-933988-23-1.

[HWM06] Tobias Hauser, Christian Wenz, and Florence Maurice. *Das Website-Handbuch: Programmierung und Design*. München: Markt+Technik, 2006. ISBN: 978-3-8272-4242-6.

[Ker11] Federico Kereki. *Essential GWT: Building for the web with Google Web toolkit 2*. Upper Saddle River and NJ: Addison-Wesley, 2011. ISBN: 978-0-321-70514-3.

[Vau10] Daniel Vaughn. *Ext GWT 2.0: Beginner's guide ; take the user experience of your website to a new level with Ext GWT*. Birmingham and U.K: Packt, 2010. ISBN: 978-1-849511-84-1.

[XW] Suman RoychoudhuryBarrett R. Bryant Jeffrey G. Gray Marjan Mernik Xiaoqing Wu. "A Two-Dimensional Separation of Concerns for Compiler Construction" ().

[Chi] Chip magazine. "Web Design 2013: Special Edition" (), pp. 1–148.

[Dan10] Daniel Guermeur. *Google App Engine Java and Gwt Application Development*. Gardners Books, 2010. ISBN: 978-1-849690-44-7.

R 2.2 Online resources

[Aes] Martin Aeschlimann. *JDT fundamentals – Become a JDT tool smith*.
URL: http://www.eclipsecon.org/2008/sub/attachments/JDT_fundamentals.ppt.

[AMD] Inc Advanced Micro Devices. *Aparapi | AMD*.
URL: http://developer.amd.com/tools/heterogeneous-computing/aparapi/.

[api] gonevertical apis. *GWT-Maps-V3-Api Showcase*.
URL: http://gonevertical-apis.appspot.com/.

[Avia] Inc Aviary. *Aviary.com*.
URL: http://www.aviary.com/.

[Avib] Inc Aviary. *Aviary.com*.
URL: http://www.aviary.com/web#.

[AW11] Andreas Schmidt and Wassili Kazakos. *Web-Technologien im Überblick*. 2.11.2009.
URL: http://klick-and-bau.com/files/WS0910/03.pdf.

[Cha06] Patrick Charollais. *Final draft Standard ECMA-262 edition 5.1, March 2011 (Rev. 6)*. 3.06.2011.
URL: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf.

[cos03] cosicimiento.blogspot.de. *My New Knowledge: Styling Excel cells with mso-number-format*. 13.03.2013.
URL: http://cosicimiento.blogspot.de/2008/11/styling-excel-cells-with-mso-number.html.

[dif] differencebetween.net. *Difference Between RPC and RMI | Difference Between | RPC vs RMI*.
URL: http://www.differencebetween.net/technology/protocols-formats/difference-between-rpc-and-rmi/.

[Ecl07] Eclipse. *org.eclipse.jdt.core.dom*. 15.07.2010.
URL: http://www.eclipse.org/jdt/core/codecoverage/B01/org.eclipse.jdt.core/org.eclipse.jdt.core.dom/index.html.

[FI] Eclipse Foundation and Inc. *Eclipse Downloads*.
URL: http://www.eclipse.org/downloads/.

[Fie09] Fielding. *HTTP/1.1: Status Code Definitions*. 1.09.2004.
URL: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

[Gab08] Stefan Gabos. *List of world countries, as MySQL dump*. 10.08.2011.
URL: http://stefangabos.ro/other-projects/list-of-world-countries-with-national-flags/#download.

[GHV] Google, Hyperakt, and Vizzuality. *The Evolution of the Web*.
URL: http://www.evolutionoftheweb.com/?hl=en.

[Gooa] Inc Google. *UserAgentGenerator.java - google-web-toolkit - Google Web Toolkit (GWT) - Google Project Hosting*.
URL: http://code.google.com/p/google-web-toolkit/source/browse/trunk/user/src/com/google/gwt/useragent/rebind/UserAgentGenerator.java?r=10943.

[Goob] Inc Google. *Willkommen bei Google Drive*.
URL: https://docs.google.com.

[Goo02a] Inc Google. *Philosophy and Goals | Android Open Source*. 1.02.2012.
URL: http://source.android.com/about/philosophy.html.

[Goo02b] Inc Google. *Declarative Layout with UiBinder - Google Web Toolkit — Google Developers*. 14.02.2013.
URL: https://developers.google.com/web-toolkit/doc/latest/DevGuideUiBinder.

[Goo06a] Inc Google. *Coding Basics - Deferred Binding - Google Web Toolkit — Google Developers*. 27.06.2012.
URL: https://developers.google.com/web-toolkit/doc/latest/DevGuideCodingBasicsDeferred.

[Goo06b] Inc Google. *Coding Basics - JavaScript Native Interface (JSNI) - Google Web Toolkit — Google Developers*. 27.06.2012.
URL: https://developers.google.com/web-toolkit/doc/latest/DevGuideCodingBasicsJSNI.

[Goo06c] Inc Google. *Coding Basics - JavaScript Overlay Types - Google Web Toolkit — Google Developers*. 27.06.2012.
URL: https://developers.google.com/web-toolkit/doc/latest/DevGuideCodingBasicsOverlay.

[Goo06d] Inc Google. *Compiling for Production Mode - FAQ - Google Web Toolkit — Google Developers*. 27.06.2012.
URL: https://developers.google.com/web-toolkit/doc/latest/FAQ_DebuggingAndCompiling.

[Goo06e] Inc Google. *HTML5 Feature Support - Google Web Toolkit — Google Developers*. 27.06.2012.
URL: https://developers.google.com/web-toolkit/doc/latest/DevGuideHtml5.

[Goo06f] Inc Google. *Introducing Elemental - Google Web Toolkit — Google Developers*. 27.06.2012.
URL: https://developers.google.com/web-toolkit/articles/elemental.

[Goo06g] Inc Google. *Locales in GWT - Google Web Toolkit — Google Developers*. 27.06.2012.
URL: https://developers.google.com/web-toolkit/doc/latest/DevGuideI18nLocale.

[Goo06h] Inc Google. *Organize Projects - Google Web Toolkit — Google Developers*. 27.06.2012.
URL: https://developers.google.com/web-toolkit/doc/latest/DevGuideOrganizingProjects#DevGuideBootstrap.

[Goo10a] Inc Google. *DOM (Google Web Toolkit Javadoc)*. 4.10.2012.
URL: http://google-web-toolkit.googlecode.com/svn/javadoc/latest/com/google/gwt/user/client/DOM.html.

[Goo10b] Inc Google. *Large scale application development and MVP - Google Web Toolkit — Google Developers*. 4.10.2012.
URL: https://developers.google.com/web-toolkit/articles/mvp-architecture.

[Goo10c] Inc Google. *Node (Google Web Toolkit Javadoc)*. 4.10.2012.
URL: http://google-web-toolkit.googlecode.com/svn/javadoc/latest/com/google/gwt/dom/client/Node.html.

[Goo10d] Inc Google. *Developer's Guide - Cell Widgets - Google Web Toolkit — Google Developers*. 25.10.2012.
URL: https://developers.google.com/web-toolkit/doc/latest/DevGuideUiCellWidgets.

[Goo10e] Inc Google. *Developer's Guide - Client Bundle - Google Web Toolkit — Google Developers*. 25.10.2012.
URL: https://developers.google.com/web-toolkit/doc/latest/DevGuideClientBundle.

[Goo10f] Inc Google. *Developer's Guide - SafeHtml - Google Web Toolkit — Google Developers*. 25.10.2012.
URL: https://developers.google.com/web-toolkit/doc/latest/DevGuideSecuritySafeHtml.

[Goo11] Google. *Showcase of Features: Stack Panel*. 4.11.2011.
URL: http://gwt.google.com/samples/Showcase/Showcase.html#!CwStackPanel.

[Gos07] James Gosling. *The Java' Language Specification*. 27.07.2012.
URL: http://docs.oracle.com/javase/specs/jls/se7/jls7.pdf.

[Goy] Jan Goyvaerts. *JavaScript RegExp Example: Online Regular Expression Tester*.
URL: http://www.regular-expressions.info/javascriptexample.html.

[Gut] Ron Gutierrez. *Unlocking the Toolkit*.
URL: https://www.owasp.org/images/7/77/Attacking_Google_Web_Toolkit.ppt.

[Han] Scott Hanselman. *JavaScript is Assembly Language for the Web: Sematic Markup is Dead! Clean vs. Machine-coded HTML - Scott Hanselman*.
URL: http://www.hanselman.com/blog/JavaScriptIsAssemblyLanguageForTheWebSematicMarkupIsDeadCleanVsMachinecodedHTML.aspx.

[Int] Intel. *Home · RiverTrail/RiverTrail Wiki · GitHub*.
URL: https://github.com/RiverTrail/RiverTrail/wiki.

[Mica] Microsoft. *About - Internet Explorer 9 Guide for Developers*.
URL: http://msdn.microsoft.com/en-us/ie/ff468705.aspx#_Intro.

[Micb] Microsoft. *Microsoft Office HTML and XML Reference*.
URL: http://msdn.microsoft.com/en-us/library/Aa155477(office.10).aspx.

[NR08] Jo~ao Dias Simon Peyton Jones Norman Ramsey. *Hoopl: Dataflow Optimization Made Simple*. 1.08.2009.
URL: http://www.cs.tufts.edu/~nr/pubs/dfopt.pdf.

[Plaa] PlayN.Game. *Angry Birds Chrome*.
URL: http://chrome.angrybirds.com/.

[Plab] PlayN.Game. *Cross Platform Game Programming with PlayN*.
URL: http://playn-2011.appspot.com/slides/index.html#5.

[Plac] PlayN.Game. *playn - Cross platform game library for N≥5 platforms - Google Project Hosting*.
URL: http://code.google.com/p/playn/.

[Reg] RegexPlanet. *RegEx: Online-Tests regulären Ausdruck für Java*.
URL: http://www.regexplanet.com/advanced/java/index.html.

[Sha] Andrey Shadrin. *Operating Systems: Remote Procedure Call*.
URL: http://www-wjp.cs.uni-saarland.de/lehre/seminar/ss04/reports/A_Shadrin-RPC-folien.pdf.

[Sha09] Francis Shanahan. *A Simple GWT Generator Example | Francis Shanahan[.com]*. 13.09.2012.
URL: http://francisshanahan.com/index.php/2010/a-simple-gwt-generator-example/.

[Son] Sonecc. *SWT & Swing - Grundlegende Informationen - java-forum.org*.
URL: http://www.java-forum.org/bilder-gui-damit-zusammenhaengt/135693-swt-swing-grundlegende-informationen.html.

[sys] eZ systems. *iSaSiLk / Communication & Messaging Security / Products / Home - Stiftung SIC*.
URL: http://jce.iaik.tugraz.at/sic/Products/Communication-Messaging-Security/iSaSiLk.

[tea] JDT/Core team. *Eclipse Java development tools (JDT)*.
URL: http://www.eclipse.org/jdt/.

[Tea] JDT/UI Team. *org.eclipse.jdt.astview - AST View*.
URL: http://www.eclipse.org/jdt/ui/astview/index.php.

[Wik03] Wikipedia. *ECMAScript - Wikipedia, the free encyclopedia*. Ed. by Wikipedia. 14.03.2013.
URL: http://en.wikipedia.org/w/index.php?oldid=541096642.

[XOO09] XOOPS. *WebOnSwing - Multi environment application framework*. 25.09.2009.
URL: http://webonswing.sourceforge.net/xoops.htm.

[z00] z00bs. *Set selected text in GWT (in order to make copy paste easier) - Stack Overflow*.
URL: http://stackoverflow.com/questions/3987824/set-selected-text-in-gwt-in-order-to-make-copy-paste-easier.

[Ado] Adobe Systems Incorporated. *Free, open-source framework | Adobe Flex*.
URL: http://www.adobe.com/de/products/flex.html.

[Ale05] Alex Gorbatchev. *SyntaxHighlighter - Download*. 9.05.2012.
URL: http://alexgorbatchev.com/SyntaxHighlighter/download/.

[And08] Andrew Valums. *Web apps vs desktop apps*. 25.08.2012.
URL: http://valums.com/web-apps/.

[Apa] Apache Software Foundation. *Apache Tomcat - Apache Tomcat 7 Downloads*.
URL: http://tomcat.apache.org/download-70.cgi.

[Bra] Brandon Donnelson. *gwt-maps-api - Google Web Toolkit Maps API V3 - Google Project Hosting*.
URL: http://code.google.com/p/gwt-maps-api/.

[CJ 01] CJ Carey. *Doppio: A JVM in Coffeescript*. 15.01.2013.
URL: http://int3.github.com/doppio/.

[Dan] Daniel Kurka. *mgwt*.
URL: http://mobilegwt.appspot.com/showcase/.

[Dan06] Daniel Kurka. *mgwt - Making GWT work with mobile*. 28.06.2012.
URL: http://www.m-gwt.com/.

[Der] Derek Greer. *Interactive Application Architecture Patterns*.
URL: http://aspiringcraftsman.com/2007/08/25/interactive-application-architecture/.

[Ecm03] Ecma International. *ECMAScript Language – test262*. 25.03.2013.
URL: http://test262.ecmascript.org/.

[Ema] Emanuelis.eu website. *ImportKey Java class*.
URL: http://www.emanuelis.eu/wp-content/uploads/2010/05/ImportKey.class.zip.

[Fed] Fedora Forum. *OpenSSL CNF file*.
URL: http://www.fedoraforum.de/openssl/openssl.cnf.

[Gar05] Garrett Smith. *Javascript Type-Conversion*. 1.05.2010.
URL: http://jibbering.com/faq/notes/type-conversion/.

[Gil02] Gildas Lormeau. *zip.js*. 24.02.2013.
URL: http://gildas-lormeau.github.com/zip.js/.

[Gre02] Gregg Tavares. *WebGL Fundamentals - HTML5 Rocks*. 9.02.2012.
URL: http://www.html5rocks.com/en/tutorials/webgl/webgl_fundamentals/.

[Kei09] Keir Clarke. *Google Maps Mania: Using GWT with the Google Maps API V3*. 2.09.2012.
URL: http://googlemapsmania.blogspot.de/2012/02/using-gwt-with-google-maps-api-v3.html.

[Khr] Khronos Group. *WebCL - Heterogeneous parallel computing in HTML5 web browsers*.
URL: http://www.khronos.org/webcl/.

[Mica] Michael von Wenckstern. *Issue 8032 - google-web-toolkit - Server cannot deserialize a serializable array if it was passed as java.io.Serializable type to a GWT RPC method [GWT 2.5.0/GWT 2.5.1 RC 1] - Google Web Toolkit (GWT) - Google Project Hosting*.
URL: http://code.google.com/p/google-web-toolkit/issues/detail?id=8032.

[Micb] Michael von Wenckstern. *java - Why does the GWT generator does not save my file to disk when I create a new one with context.tryCreateResource? - Stack Overflow*.
URL: http://stackoverflow.com/questions/15252922/why-does-the-gwt-generator-does-not-save-my-file-to-disk-when-i-create-a-new-one.

[Micc] Michael von Wenckstern. *MVP example: Agricola borad game*.
URL: http://www.informatik.tu-freiberg.de/prof2/GWT/Agricola.html#start.

[Moz03a] Mozilla Developer Network. *mozilla/shumway @ GitHub*. 12.03.2013.
URL: http://mozilla.github.com/shumway/.

[Moz03b] Mozilla Developer Network. *JavaScript pdf reader*. 13.03.2013.
URL: http://mozilla.github.com/pdf.js/web/viewer.html.

[Moz08] Mozilla Developer Network. *SpiderMonkey*. 15.08.2012.
URL: https://developer.mozilla.org/en-US/docs/SpiderMonkey.

[Ope02] OpenSSL Community. *Shining Light Productions - Win32 OpenSSL*. 11.02.2013.
URL: http://slproweb.com/products/Win32OpenSSL.html.

[Oraa] Oracle Corporation. *Java SE Downloads*.
URL: http://www.oracle.com/technetwork/java/javase/downloads/index.html.

[Orab] Oracle Corporation. *MySQL :: MySQL 5.0 Reference Manual :: 1.3.3 History of MySQL*.
URL: http://dev.mysql.com/doc/refman/5.0/en/history.html.

[Orac] Oracle Corporation. *MySQL :: MySQL Downloads*.
URL: http://dev.mysql.com/downloads/.

[Orad] Oracle Corporation. *MySQL :: Using MySQL With Java*.
URL: http://dev.mysql.com/usingmysql/java/.

[Ora03] Oracle Corporation. *Java Native Interface: Programmer's Guide and Specification*. 2003.
URL: http://java.sun.com/docs/books/jni/.

[Ora12] Oracle Corporation. *OpenJDK: Project Sumatra*. 13.12.2012.
URL: http://openjdk.java.net/projects/sumatra/.

[Ori02] Orientation in Objects GmbH. *Vergleich von Java Web-Frameworks wie GWT vs. JSF*. 27.02.2013.
URL: http://www.oio.de/public/java/java-web-frameworks-vergleich/jsf-vs-gwt-studie.htm#a14.

[Pro04] Prof. David MacQueen. *Liveness Analysis*. 21.04.2004.
URL: http://www.classes.cs.uchicago.edu/archive/2004/spring/22620-1/docs/liveness.pdf.

[Pro06] Prof. Dr. Martin Bertau. *Richtlinien für das Verfassen wissenschaftlicher Arbeiten*. 5.06.2008.
URL: http://tu-freiberg.de/sites/default/files/media/institut-fuer-technische-chemie-1876/richtlinien.pdf.

[R.] R. Fielding. *RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1 (RFC2616)*.
URL: http://www.faqs.org/rfcs/rfc2616.html.

[Refa] Refsnes Data. *CSS Reference*.
URL: http://www.w3schools.com/cssref/default.asp.

[Refb] Refsnes Data. *HTML DOM Nodes*.
URL: http://www.w3schools.com/htmldom/dom_nodes.asp.

[Refc] Refsnes Data. *HTML5 Web Storage*.
URL: http://www.w3schools.com/html/html5_webstorage.asp.

[Refd] Refsnes Data. *JavaScript HTML DOM*.
URL: http://www.w3schools.com/js/js_htmldom.asp.

[Refe] Refsnes Data. *JavaScript Tutorial*.
URL: http://www.w3schools.com/js/default.asp.

[Reff] Refsnes Data. *XML Introduction - What is XML?*
URL: http://www.w3schools.com/xml/xml_whatis.asp.

[Ros03] Ross Lodge. *Emulating JRE Classes In GWT*. 18.03.2013.
URL: http://concentricsky.com/blog/2011/mar/emulating-jre-classes-gwt.

[S.] S. Netesany. *gwt-versatile - GWT Features - Google Project Hosting*.
URL: http://code.google.com/p/gwt-versatile/.

[Sen04] Sencha Inc. *BlurEvent (Sencha GXT 3.0.0 API)*. 29.04.2012.
URL: http://dev.sencha.com/deploy/gxt-3.0.0/javadoc/gxt/com/sencha/gxt/widget/core/client/event/BlurEvent.html.

[Sen06a] Sencha Inc. *Ext GWT 2.2.6 Explorer*. 4.06.2012.
URL: http://www.sencha.com/examples-2/#borderlayout.

[Sen06b] Sencha Inc. *Ext GWT 2.2.6 Explorer - Advanced Toolbar*. 4.06.2012.
URL: http://www.sencha.com/examples-2/#advancedtoolbar.

[Sen08a] Sencha Inc. *Sencha GXT Desktop - 3.0.1*. 13.08.2012.
URL: http://www.sencha.com/examples/desktop.html.

[Sen08b] Sencha Inc. *Sencha GXT Explorer Demo - 3.0.1*. 13.08.2012.
URL: http://www.sencha.com/examples/#.

[Sta] StartCom Ltd. *StartSSL™ Certificates & Public Key Infrastructure - StartSSL™ Home*.
URL: https://www.startssl.com/.

[Ste] Stefan Haustein. *quake2-gwt-port - Quake II GWT Port - Google Project Hosting*.
URL: http://code.google.com/p/quake2-gwt-port/.

[The] The Sun. *Arsene Wenger tells fans: You'll miss me when I'm gone | The Sun*.
URL: http://www.thesun.co.uk/sol/homepage/sport/football/4801665/Arsene-Wenger-tells-fans-Youll-miss-me-when-Im-gone.html.

[The08a] The Apache Software Foundation. *The Apache Struts Project*. 13.08.2012.
URL: http://struts.apache.org/.

[The08b] The Apache Software Foundation. *Apache Wicket - Welcome to Apache Wicket*. 25.08.2012.
URL: http://wicket.apache.org/.

[Tim12] Tim Berners-Lee. *The World Wide Web project*. 3.12.1992.
URL: http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html.

[Uni09] University Frankfurt. *Statutory declaration*. 2.09.2010.
URL: http://www.wiwi.uni-frankfurt.de/fileadmin/user_upload/dateien_pruefungsamt/Formulare_Merkblaetter/Ehrenwoertliche_Erklaerung_MA_Billingual.pdf.

[Wir] Wireshark Foundation. *Wireshark · Go deep.*
URL: http://www.wireshark.org/.

[Wor06] World Wide Web Consortium (W3C). *W3C SVG Working Group*. 21.06.2012.
URL: http://www.w3.org/Graphics/SVG/.

[Wor11] World Wide Web Consortium (W3C). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 21.11.2008.
URL: http://www.w3.org/TR/REC-xml/.

Statutory declaration

I herewith declare that I have completed the present thesis independently making use only of the specified literature and aids. Sentences or parts of sentences quoted literally are marked as quotations; identification of other references with regard to the statement and scope of the work is quoted.[47]

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegeben Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.[48]

Freiberg, den 30.04.2013 _____

[47] Statutory declaration of University Frankfurt; source [Uni09]

[48] Eidesstattliche Erklärung aus der Promotionsordnung der Technischen Universität Bergakademie Freiberg vom 12. Dezember 1994; source [Pro06, p. 1]