Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Datenmanagement und -exploration
Prof. Dr. T. Seidl

Proseminar

Google Web Toolkit

Stefan Dollase

May 2012

Betreuung: Prof. Dr. T. Seidl
Dipl.-Ing. Marwan Hassani

Hiermit versichere ich, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, den 01.06.2012

Contents

List of Figures

Listings

Abstract

In this paper an introduction to developing web applications with the Google Web Toolkit (GWT) in particular and a short summary about developing web applications in general will be given. Furthermore use cases of GWT will be explain and alternatives will be outlined.

Chapter 1

Introduction and Motivation

1.1 The problem with client-side scripting

When developing web applications, functionality on the client side needs to be added. This is necessary, because otherwise the page would be reloaded with every click on a link and there would be no possibility to include animations or AJAX requests or other dynamic behaviour. This means that code has to be written which the browser will execute, but here a fundamental decision has to be made. The options are Adobe Flash, Microsoft Silverlight, Java Applets and Javascript. Adobe Flash, Microsoft Silverlight and Java Applets offer the advantage that they work everywhere the same way if their plugin is installed. The disadvantage is that they need this plugin which is sometimes not available on a platform, e.g. for iOS. This means that if many different people with many different clients should use the web application, it has to be developed using Javascript. This is because Javascript is the only standardised scripting language which is supported by most browsers without an extra plugin. Naturally Javascript has a disadvantage, too. The code will be computed by many different browsers and each of them will interpret it in a slightly different way. They use different event models and each browser offers other features. That is the reason why the code has to branch in different cases to get the same behaviour in all of them. Luckily there exist libraries to abstract from this.

1.2 How the jQuery Project tries to solve it

The jQuery project[8] for example has the aim to provide a library which offers an abstract and browser-independent interface that can be used by the programmer. Instead of calling the original methods the library ones have to

be called. The library itself will execute the corresponding methods of the browser to accomplish the intended functionality.

1.3 The solution of GWT

Google chose to go another way. They neither wanted every website to download a large Javascript library nor wanted to send code to a browser that will never execute it, because it was written for another one. They wanted to minimize traffic and they wanted to be able to concentrate on the functionality of the application while developing it. Their solution is to develop a toolkit named Google Web Toolkit (GWT) which meets these requirements. The idea is that no Javascript code is written but a Java code, and a cross compiler will convert it to Javascript. It can obfuscate the resulting code automatically and it is able to compile it once for each of the major browsers to meet their particularities. Additionally all the benefits of developing in Java like using the integrated development environment Eclipse and the possibility to find errors at compile time because Java is type safe are achieved.

Chapter 2

Working with GWT

2.1 Installation of the Environment

To be able to take a closer look at this toolkit download and install Eclipse and the Google Plugin for Eclipse by following the guide[3]. After this a new Web Application Project can be created by following this guide:

2.1.1 Create a new Project

1. in the menu bar click *File* > *New* > *Web Application Project*
2. enter Project name *MyProject*
3. enter Package `com.example.myproject`
4. disable *Use Google App Engine*
5. disable *Generate project sample code*
6. click *Finish*

2.1.2 Create a new Module

1. in the menu bar click *File* > *New* > *Module*
2. enter Source folder *MyProject/src*
3. enter Package `com.example.myproject`
4. enter Module name *MyProject*
5. click *Finish*

2.1.3 Create a new EntryPoint

1. in the menu bar click *File* > *New* > *Entry Point Class*
2. enter Source folder *MyProject/src*
3. enter Name *MyProject*
4. click *Finish*

2.1.4 Create a new HTML Page

1. in the menu bar click *File* > *New* > *HTML Page*
2. enter File name *index.html*
3. click *Finish*

2.1.5 Connect the HTML Page with the Module

1. in *MyProject.gwt.xml* add `<module rename-to="myproject">`
2. in *index.html* add `<script type="text/javascript" language="javascript" src="myproject/myproject.nocache.js"></script>`

2.1.6 Install the Google Web Toolkit Developer Plugin in the browser

1. save all files
2. in the *Project Explorer* window right-click on *MyProject.gwt.xml* and click *Run As* > *Web Application*
3. in the *Development Mode* window double-click the link
4. (the browser will open)
5. download the *Google Web Toolkit Developer Plugin*
6. in the *Development Mode* window click the red square to stop the execution

2.2 The idea of GWT

2.2.1 The cross compiler

GWT essentially is a cross compiler from a Java to a Javascript code. That means the programmer writes Java code as shown in Listing 1 and GWT cross compiles it to Javascript code like in Listing 2. To minimize the traffic, GWT can automatically obfuscate the generated Javascript code.

```
public void modify(String author, String title,
    String text) {
  this.author = author;
  this.title = title;
  this.text = text;
  this.lastModified = new Date();
}
```

Listing 1: Java code of the `modify` method in Note.java

```
function $modify(this$static, author, title, text){
  this$static.author = author;
  this$static.title = title;
  this$static.text_0 = text;
  this$static.lastModified = new Date_1;
}
```

Listing 2: Javascript code of the `modify` method

2.2.2 Handling of different browsers

```
        unflattenKeylistIntoAnswers(['ie8'],
            '47056B0A16CF8F7C851D2E48541E54E3');
        unflattenKeylistIntoAnswers(['safari'],
            '4EAE8B444030A660BA1092A0674836FB');
        unflattenKeylistIntoAnswers(['gecko1_8'],
            '90D20EB14CCEAE76DA182A87F7E19BA3');
        unflattenKeylistIntoAnswers(['opera'],
            'EA8EA32B810B8A4ACE1996490899AC5F');
        unflattenKeylistIntoAnswers(['ie6'],
            'EE5DABF00BD9583A142AA8657D5E0FE7');
        unflattenKeylistIntoAnswers(['ie9'],
            'F50399EC9A9F81B7BB524E0C479D4858');
```

Listing 3: Mapping from the browser to the file name in notes.nocache.js

GWT compiles the code once for each of the major browsers, so it can meet the particularities of each browser. Additionally no code will be downloaded which was originally generated for another browser. Listing 3 shows the mapping from the browser to the file name, meaning if the current browser can be detected e.g. as 'safari' the file '4EAE8B444030A660BA1092A0674836FB .cache.html' will be loaded and executed. When comparing the files for 'ie6' and 'ie9' one can see that they are indeed different. The file for ie9 contains the function in Listing 4 called `$toJSONObject()` where the file for ie6 does not even contain the word 'toJSONObject'.

```
function $toJSONObject(this$static){
  var o;
  o = new JSONObject_0;
  $put(o, 'id', new JSONString_0(this$static.id));
  $put(o, 'author', new JSONString_0(this$static.author));
  $put(o, 'title', new JSONString_0(this$static.title));
  $put(o, 'text', new JSONString_0(this$static.text_0));
  $put(o, 'lastModified', new JSONNumber_0(toDouble(fromDouble
      (this$static.lastModified.jsdate.getTime()))));
  return o;
}
```

Listing 4: The function `$toJSONObject()` in the ie9 specific file

2.3 Emulation of the JRE

An advantage of Java is the large class library which comes with the JRE. With GWT Google provides a Javascript implementation of many classes of the JRE. They also provide a documentation what exactly is supported[4]. The advantage of having a Javascript implementation of the JRE is that Java code can easily be reused. For example if Java is used on the server side of the web application and GWT on the client side exactly the same code can be used for the data structure or common methods. The only difference will be that on the server side the real Java code will run where on the client side the cross compiled Javascript code will be executed.

2.3.1 Example - `HashMap` and `Map`

One example for a class and an interface from the JRE with a Javascript implementation of GWT is `java.util.HashMap` and `java.util.Map`, see Listing 5. In Listing 6 parts of the Javascript version of this can be found.

```
package com.example.notes.client;

import java.util.HashMap;
import java.util.Map;
...
public class NoteCollection {
  ...
  private Map<String, Note> notes;
  public NoteCollection() {
    this.notes = new HashMap<String, Note>();
    ...
  }
  ...
  public Note get(String id) {
    return this.notes.get(id);
  }
  public void remove(String id) {
    this.notes.remove(id);
    ...
  }
  public boolean isEmpty() {
    return this.notes.isEmpty();
  }
}
```

Listing 5: Java version of `HashMap` and `Map` in NoteCollection.java

```
...
function NoteCollection_0(){
  this.notes = new HashMap_0;
  ...
}
...
function $remove(this$static, id){
  $remove_4(this$static.notes, id);
  ...
}
...
```

Listing 6: Parts of the Javascript version of `HashMap` and `Map`

2.4 Two basic classes

2.4.1 `EntryPoint`

GWT defines the interface `com.google.gwt.core.client.EntryPoint` which specifies the method signature of a method called `onModuleLoad()`. If this interface is implemented the cross compiled Javascript code will start on the `onModuleLoad()` method of an instance of the implementing class. This is comparable with the main method in standard Java programs.

2.4.2 `Widget`

In GWT all classes which are displayable to the GUI extend the class `com.google.gwt.user.client.ui.Widget`. This is similar to `java.awt.Component` from AWT or `javax.swing.JComponent` from Swing. So a custom Widget can be created by extending the `Widget` class.

2.4.3 Example - `Button`, `RootPanel` and `EntryPoint`

```
public class Notes implements EntryPoint {
  private RootPanel createNotePanel;
  private Button createNoteCancel;
  public void onModuleLoad() {
    createNotePanel = RootPanel.get("createNote");
    createNoteCancel = new Button("Cancel", new ClickHandler()
        {
      public void onClick(ClickEvent event) {
        showEditNotePanel();
      }
    });
    createNotePanel.add(createNoteCancel);
    ...
  }
  private void showEditNotePanel() {
    ...
  }
}
```

Listing 7: Parts of Notes.java

In Listing 7 parts of the `Notes` class are shown which implements `EntryPoint`. The execution begins in the `onModuleLoad()` method where first a reference to an existing HTML element is created, second a `Button` with the text "Cancel" is created which will call the `showEditNotePanel()` method when clicked and third the `Button` is

added to the HTML element. Both, the `RootPanel` class and the `Button` class extend `Widget`, so they can both be displayed. It is noteworthy that no HTML tags appear in the Java code when working with GWT.

2.5 Feature detection at runtime

HTML and CSS are standards which browsers should implement. The problem is that not every browser implements every feature of the standard. So the possibility is needed to detect at runtime whether the current browser is implementing the feature the programmer wants to use. Then it becomes possible to fallback to a custom implementation of this feature or to deactivate the functionality which needs this feature to run correctly.

2.5.1 Example - HTML5 Web Storage

```
...
import com.google.gwt.storage.client.Storage;
import com.google.gwt.storage.client.StorageMap;
public class NoteCollection {
  ...
  private void loadFromStorage() {
    Storage storage = Storage.getLocalStorageIfSupported();
    Map<String, String> map;
    if (storage != null) {
      map = new StorageMap(storage);
      ...
    }
  }
  ...
}
```

Listing 8: Web Storage in NoteCollection.java

In Listing 8 an example of how to work with Web Storage is given. If the return value of `Storage.getLocalStorageIfSupported()` is null then the feature is not supported. Otherwise a `Storage` object will be returned to access the data. If the programmer prefers to use the storage like any other `java.util.Map` then `StorageMap` will fulfil exactly this task. Google offers a guide about Web Storage and GWT[2].

2.6 AJAX and GWT

AJAX (Asynchronous Javascript and XML) is a technique to exchange data between the client and the server without reloading the page, so it is an important part of GWT.

2.6.1 The server side

If an AJAX Requests should be made a class as shown in Listing 9 has to be defined which extends `RemoteServiceServlet`. An instance of this class will run on the server side to answer the requests. It has to implement an interface which extends `RemoteService`, see Listing 10. This interface defines the different calls that the server will accept. The `RemoteService` interface is an interface which does not defined any method. It has to be extended to signal which interfaces should be taken as definition of the remote calls.

```
import com.google.gwt.user.server.rpc.RemoteServiceServlet;
public class MyRemoteServiceImpl extends RemoteServiceServlet
    implements MyRemoteService {
  public String getEntry(String id) {
    return "The Entry with the id " + id;
  }
}
```

Listing 9: The class `MyRemoteServiceImpl`

```
import com.google.gwt.user.client.rpc.RemoteService;
public interface MyRemoteService extends RemoteService {
  String getEntry(String id);
}
```

Listing 10: The interface `MyRemoteService`

2.6.2 The client side

The client needs a second interface as shown in Listing 11. It defines the same methods as the interface for the server, but the return type is void. Instead at the end of the parameter list a new entry is added with the type `AsyncCallback<T>` where `T` is the return type of the original method. There is no need to implement this second interface, because the method `GWT.create` will return an object of this type as shown in Listing 12. The call itself will be done by calling the corresponding method of this returned

object. As soon as the call returns successfully the `onSuccess(T result)` method will be called on the object which was passed as last argument. If the remote call fails the `onFailure(Throwable caught)` method will be executed.

```
import com.google.gwt.user.client.rpc.AsyncCallback;
public interface MyRemoteServiceAsync {
  public void getEntry(String id,
    AsyncCallback<String> callback);
}
```

Listing 11: The interface `MyRemoteServiceAsync`

```
import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.core.client.GWT;
import com.google.gwt.user.client.rpc.AsyncCallback;
public class MyClient implements EntryPoint {
  public void onModuleLoad() {
    MyRemoteServiceAsync service =
      GWT.create(MyRemoteService.class);
    service.getEntry("myid", new AsyncCallback<String>() {
      public void onSuccess(String result) {
        ...
      }
      public void onFailure(Throwable caught) {
        ...
      }
    });
  }
}
```

Listing 12: The class `MyClient`

2.7 The Model View Presenter Design Pattern

Google suggests MVP (Model View Presenter) as a design pattern when working with GWT[5]. The basic idea is to separate the application in three main parts. The Model manages all data. The View is responsible for the GUI, meaning all Widgets and the Layout but it does not know the Model, thus it has no idea what data it displays. The third part is the Presenter, which listens to events from the View and the Model to change the Model and the View as a result of the events. So the Presenter has to contain the program logic. Additionally every View has its custom Presenter. However the Presenter has not to know how the GUI looks like, it actually interacts

via an interface with the View. This helps the developer to test the Presenter and the Model code as he or she can simply implement a special View which contains the test cases. Another advantage of this design pattern is that it helps to split the code in a large number of nearly independent parts, so it becomes easier to work with many developers at one web application. Figure 1 shows a diagram of this pattern.

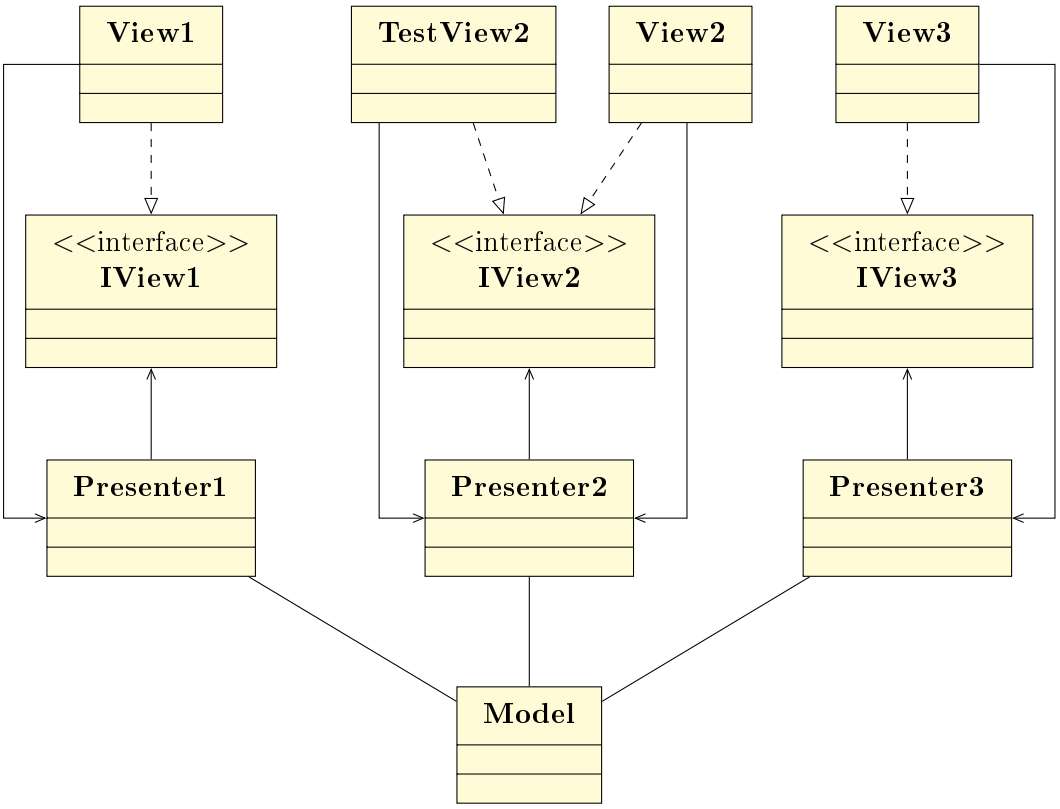

Figure 1: Diagram of the MVP Design Pattern

2.8 The complete example

For this paper a web application which allows a user to manage notes was written. Notes can be added, removed, read and modified. Each note comprises the name of the author, a title, a text and the date of the last modification. If available, it uses Web Storage to save the notes, so they can be loaded the next time the web application is opened, see 2.5.1. Additionally it uses `HashMap` and `Map` to manage the notes at runtime, see 2.3.1. After following point 2-4 in 2.1.6 the browser will open with the web application

running, see Figure 2. The surface in Figure 3 comes to front while creating a new note. If all notes has been deleted the surface in Figure 4 comes to front. The buttons can be used to add or delete a note, to discard the changes by clicking reset or to save the changes.

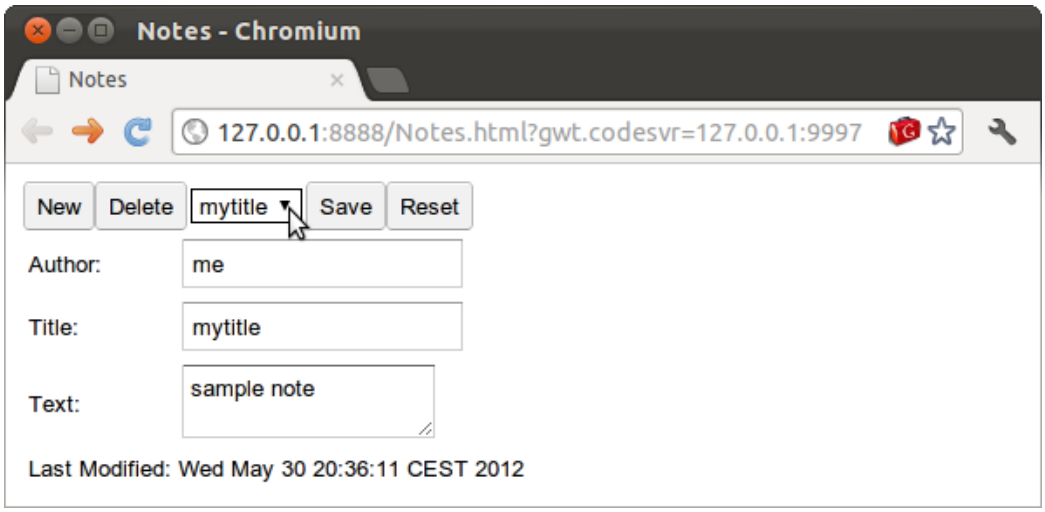

Figure 2: Notes - default surface

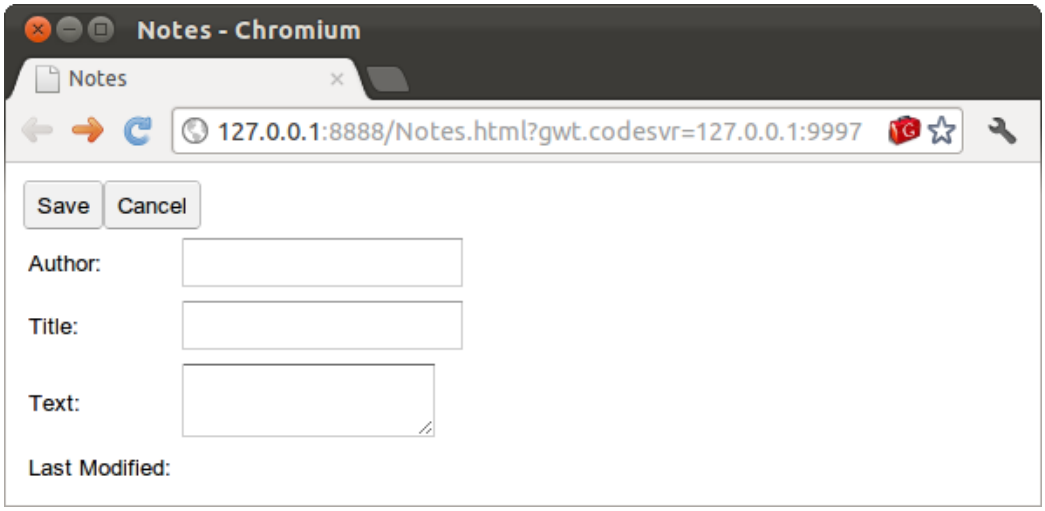

Figure 3: Notes - new surface

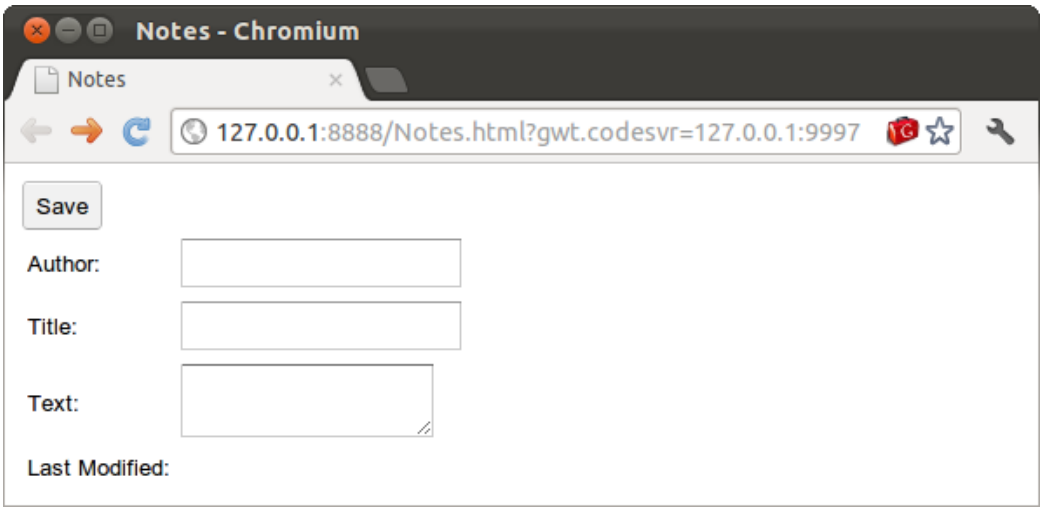

Figure 4: Notes - all deleted surface

Chapter 3

Validation

3.1 Where to use GWT

On the one hand it is a good idea to use GWT when developing a web application meaning a web page that one want to use more like a program than like a static document [6]. It provides a high level of abstraction and a great amount of optimization. Additionally frameworks as for example Sencha GXT [7] exist, which makes it even more easy to work with GWT. On the other hand jQuery may be the better choice when developing a web page with effects and only less functionality, because GWT does not really support effects. Furthermore GWT cannot be used when it is forbidden to use Javascript which is sometimes done for security reasons.

3.2 Who uses GWT

Three prominent examples are the VZ networks such as studivz, schülervz and meinvz[1]. Beyond that it is little documentation available about the usage of GWT.

Bibliography

[1] golem.de. Letzte Chance für die VZ-Netzwerke - Technisch ein großer Schritt. http://www.golem.de/1109/86675-4.html, May 2012.

[2] Google. Developer's Guide - Client-side Storage (Web Storage). https://developers.google.com/web-toolkit/doc/latest/DevGuideHtml5Storage, May 2012.

[3] Google. Downloading and Installing the Plugin. https://developers.google.com/eclipse/docs/download, May 2012.

[4] Google. JRE Emulation Reference. https://developers.google.com/web-toolkit/doc/latest/RefJreEmulation, May 2012.

[5] Google. Large scale application development and MVP. https://developers.google.com/web-toolkit/articles/mvp-architecture, May 2012.

[6] Seemann, M. *Das google web toolkit: Gwt*, page 9. O'Reilly Germany, 2007.

[7] Sencha Inc. Sencha GXT 3. http://www.sencha.com/products/gxt, May 2012.

[8] The jQuery Project. http://jquery.com/, May 2012.