Social Network Application based on Google Web Toolkit.

Piotr Pawlak, Bartosz Sakowicz, Piotr Mazur, Andrzej Napieralski

Abstract – **This article describes social network application based on Google Web Toolkit (GWT) as the new technology in creation of rich AJAX applications using only Java as the programming language, which is later on compiled into pure JavaScript and deployed as regular web site. Example application is a simple social portal as nowadays they are the most common representatives of Web 2.0 applications that require fast, rich interfaces as the ones created using GWT.**

Keywords –**Social network, GWT, AJAX, Java**

I. INTRODUCTION

Today, the Internet comprises mostly Web pages based on Web 1.0 technology which is the remnants of times when Internet was used only for selling real services online. The approach that was in use treated Web browsers as dumb terminals for displaying data prepared by the server. With the increasing popularity of the Internet, new term – Web 2.0 was introduced as a response to growing need of interactivity which is the core assumption of contemporary user collaboration over the network. It has become online service itself. The main problem that had to be addressed with Asynchronous JavaScript and XML (AJAX) technology concerned the need for resource-wasteful whole page refreshing after every user action regardless of whether the contents have actually changed.

Using this JavaScript technology, each page update with new data from server was accompanied by asynchronous request which made complex applications behave more like traditional desktop software with distributed architecture. New application structure had a clear division of pars: client side interface, business logic on server and data exchange mechanism.

With the increasing complexity of AJAX applications, new scalability problems arose. The biggest complication is the origin of AJAX technology. It is derived from web technologies, it was never meant to be used exclusively in large scale applications and therefore lacks tools (i.e. Integrated Development Environment (IDE), debugging etc.) that support development in the way desktop software creation is.

P.Pawlak, B. Sakowicz, P. Mazur, A. Napieralski
Department of Microelectronics and Computer Science,
Technical University of Lodz, Poland
Email: sakowicz@dmcs.pl

There have been many attempts to design frameworks alleviating the pain of AJAX, but still developers had to deal with cumbersome JavaScript [11].

Google came up with new approach: why create something new instead of just taking the advantages of already known, well-founded existing technologies, methodologies and tools?

Google Web Toolkit is an open source Java development framework that lets developers escape the matrix of technologies that make writing AJAX applications so difficult and error prone. With GWT, programmers can develop and debug AJAX applications in the Java language using generic Java development tools of their choice. When the application is deployed to production, the GWT compiler translates user written Java application to browser-compliant JavaScript and HTML[1-9].

GWT development cycle has the following steps:

1. Using developer's favorite Java IDE to write and debug an application in the Java language, using as many (or as few) GWT libraries as it is is considered useful.
2. Using GWT's Java-to-JavaScript compiler to distill application into a set of JavaScript and HTML files that can be served with any web server.
3. Confirming that application works in each browser that it to be supported, which usually takes no additional work.

As mentioned, GWT consists of libraries responsible for specific functionalities that may or may not be used (internationalization, communication, JavaScript Native Interface, visual interface components - widgets etc.) Most important elements are shown in Fig. 1.

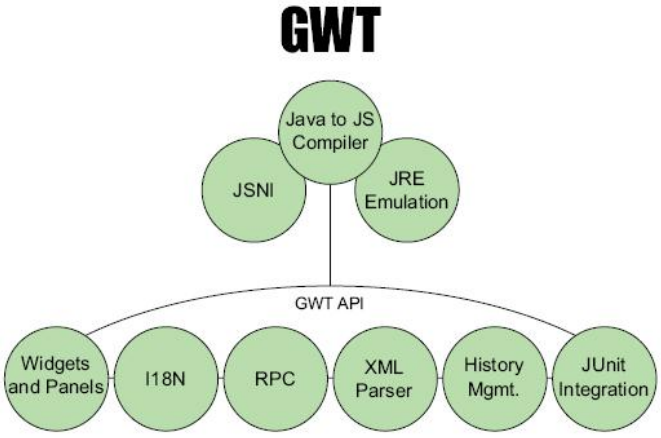

Fig. 1 - GWT component overview

II. Application Goals

Described application named Northgate is a social portal that allows registered users to create bets concerning each user's opinions and beliefs e.g. "Do you prefer Blondes or Brunettes?". Users join selected bets stating their point of view. Each participant has some number of virtual money that he can earn if his bet was in the majority of answers. Contestants that were in minority have their money pool shared equally amongst winning players. To increase attractiveness, during time when the bet is open users can view statistics based on charts that depict betting community grouped by factors like: education type, income, age, sex, country etc. This way they can try to predict possible bet outcome by analyzing participants' backgrounds. It also just gives interesting information about society so it can serve as both curiosity and gambling entertainment. Information required by betting functionality is provided by configuring profiles where users can publish additional description, picture galleries. Users can also communicate with portal's internal messages mechanism.

III. Architecture

Application was intended to be designed according to good practice Model-View-Controller pattern to create portable application consisting of independent elements [12-14]. However, this pattern cannot be easily adapted in case of distributed AJAX application such as Northgate as communication is asynchronous. Fig. 2 shows correlation between elements.

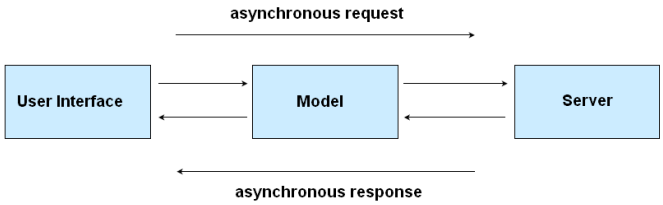

Fig. 2 Application pattern

Each request for data cannot be limited to directly operate only on nearest layer as it is only expected to receive response in some nearest future or not at all due to errors that can happen on the way in particular element (interface, network connection, business logic etc.). This creates more complex, distributed, abstract model that sits in the middle between actual components of client and server. Additionally, request must be followed by response in the form of server's callback request which then needs to be processed back in the calling method on the client side.

IV. Data Model

Application uses Data Access Object (DAO) design pattern to separate logic from implementation of operations on the database. So if the data persistence technology changes, source code of logic classes may be left unchanged.

The persistence layer is based on Hibernate 3. Hibernate requires metadata to create object-relational mappings. It uses XML mapping documents that hold information about each domain class and attributes [15,16].

Each domain class has corresponding DAO interface that specify find, save, update, and delete methods. Each implementation class extends abstract class BaseDAO. Some of the classes are introduced on Fig 4. The BaseDAO gets a reference to a Hibernate's Session object from the SessionFactory and makes sure that only one instance of session per thread is created. Fig. 3 illustrates relations between classes.

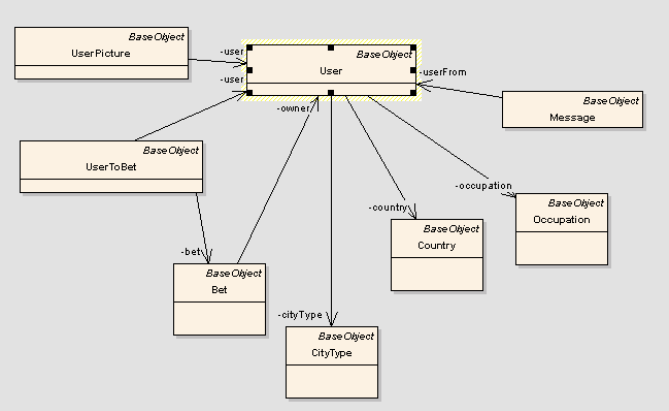

Fig. 3 Objects relationship diagram

PostgreSQL 8.2 is used as a database system. It is distributed as open source software, provides portability and high performance. Hibernate's configuration is stored in XML file. There is defined connection through JDBC to the database and SQL dialect, so that system specific metadata can be generated. Mapping files are created using XDoclet from POJO classes. A sample from mapping document of the UserToBet class is introduced on Fig. 4.

```
<hibernate-mapping>
<class
name="pl.northgate.server.dao.model.UserToBet"
table="usertobet">
<id name="id" column="id"
       type="java.lang.Long">
    <generator class="increment">
    </generator>
</id>
<version name="version" column="version"
       type="java.lang.Long"/>
<property name="joined" type="java.util.Date"
update="true"insert="true" column="joined"/>
<many-to-one name="bet"
class="pl.northgate.server.dao.model.Bet"
cascade="none"
             outer-join="auto"
            update="true"
            insert="true"
            column="betfk"
            not-null="true"/>
<many-to-one name="user"
class="pl.northgate.server.dao.model.User"
            cascade="none" outer-join="auto"
            update="true" insert="true"
            column="userfk" not-null="true"/>
<property name="answer" type="int"
       update="true" insert="true"
       column="answer"/>
```

```
</class>
</hibernate-mapping>
```

Fig 4. Hibernate mapping file for class UserToBet

References to Bets and User objects are mapped as many-to-one relationship to link tables with foreign key.

V. User Interface

User interface is built using libraries and visual components provided by non-official GWT project GWT-EXT which offers more complex and attractive user interface controls like grids with paging and grouping, editable trees, layouts, progress bars and many more.

The layout in Northgate application is based on main menu bar on top of the page with dynamic content generated for each menu section that user visits with occasional usage of GWT windows displaying more detailed information on user, picture, bet, message etc. Fig. 5 shows screen where bet in progress can by viewed in a window on top of bets list.

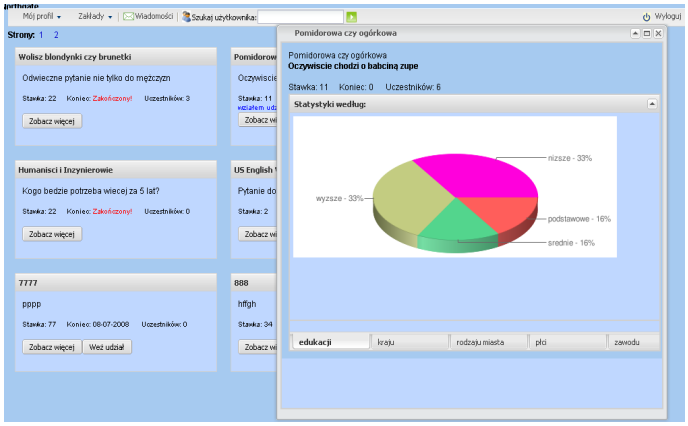

Fig.5 Application's screen for viewing bets

Additionally, Northgate utilizes external web service - Google Charts API to draw images of charts in a simple way, just by inserting *img* HTML tag with source set to appropriate URL address containing data to be displayed as parameters [10]. Sample source code for method responsible for making asynchronous request for statistics data and presenting pie chart is exhibited shown on Fig 6.

```
private void getPieChartPanel(final int chartType,
final int betId) {
BetsServiceAsync getBetData = (BetsServiceAsync) GWT
        .create(BetsService.class);
ServiceDefTarget endpoint = (ServiceDefTarget)
getBetData;
String moduleRelativeURL = GWT.getModuleBaseURL() +
"Bets";
endpoint.setServiceEntryPoint(moduleRelativeURL);
AsyncCallback callback = new AsyncCallback() {
public void onSuccess(Object result) {
String[][] factors = (String[][]) result;
String type = "cht=p3";String size = "chs=480x200";
String title = "chtt= ";String labels = "chl=";
String colors = "chco=FF5E5B,59FF54,4980FF,FFF242 ";
String data = "chd=t:";
for (int j = 0; j < factors.length; j++) {
       labels += factors[j][0] +factors[j][1]+"%|";
        data += factors[j][1] + ",";
}
labels = labels.substring(0, labels.length() - 1);
data = data.substring(0, data.length() - 1);
String url =
Format.format("{0}?{1}&{2}&{3}&{4}&{5}&{6}",
new String[] { CHART_URL_PREFIX, type, size, title,
labels, colors, data });
previewPanel.getBody().update("<img src =\"" + url +
"\"/>");
}
public void onFailure(Throwable caught) {
        MessageBox.alert("Error getting data");
        }
};
getBetData.getBetsChartData(chartType, betId,
callback);
}
```

Fig. 6 Source code for request displaying statistics pie chart

VI. Distributed Communication

Although GWT is mainly for building client side of the applications and there are no restrictions on server side technology to be used, the best solution that is also implemented in Northgate portal is to design application completely in Java with GWT in mind from the beginning. Described application takes advantage of the GWT Remote Procedure Call interface that is best optimized and simplified. Fig. 6 presents invocation of asynchronous request via `AsyncCallback` interface with two methods called as a response depending on result. Besides Java implementation code on server which returns array (for serialization purposes) of `String[][]` type containing key-value pairs of data to be displayed. There are also two stub interfaces that need to be created on client's side.

```
public interface BetsServiceAsync {
        public abstract void getBetsChartData (int
type,int betId, AsyncCallback callback);
}
public interface BetsService extends RemoteService {
        public abstract String[][] getBetsChartData
(int type, int betId);
}
```

Fig 7. GWT RPC interfaces source code

GWT RPC mechanism can be used to transfer simple data only. Northgate portal gives possibility to upload user pictures which has to be dealt with in a different way. To send picture to the server, client forms regular POST method headers using browser's binary file upload capability. On the server side there is a separate Java Servlet to handle file uploads. The servlet does not use any of the GWT code and is created as it would be in any other Java technology using *HttpServlet* class and it's *doPost* method. Server's response is sent in JSON format so that it can be interpreted by GWT user interface and GWT's *JSONResponse* class that is designed especially for this purpose when GWT RPC mechanisms cannot be used.

VII. Conclusions

The main aim of this project was to show how AJAX technology is promising if approached with the right tools, especially for the applications in increasingly popular Web 2.0 trend just as described Northgate social portal is.

Google Web Toolkit at the moment is not “mature” enough technology to become standard in rich internet applications.

To create Northgate portal authors used most commonly known tools, Firefox & IE internet browsers and IDE software like Eclipse working under Windows operating system. One should not forget about many other software products that are in use by both developers and users. This variety still renders mainly compatibility issues to be addressed but at the time the Northgate portal project was started there was far less libraries, documentations, user contributions and materials available for GWT developers than it is now.

Authors observed some difficulties in adapting traditional java development thinking to asynchronous distributed web model rules imposed by GWT but this may be just a personal impression that goes away with increasing practical experience with this framework.

Social network is a very broad term and therefore there could be many different example applications covering this subject with GWT technology. Northgate is just one of these examples that could evolve almost endlessly by including more functionalities and gadgets from vast pool that is already provided.

ACKNOWLEDGEMENTS

This research was supported by the Technical University of Lodz Grant K-25/1/2008/Dz.St.

REFERENCES

[1] R. Hanson, A. Tacy, “*GWT In Action”*, Manning, 2007
[2] Official Google Web Toolkit project page - http://code.google.com/webtoolkit/
[3] R. Dewsbury, “*Google Web Toolkit Applications”*, Prentice Hall, 2007
[4] E. Burnette: *“Google Web Toolkit: Taking the pain out of AJAX”,* The Pragmatic Programmers, 2006
[5] OnGWT: http://www.ongwt.com
[6] GWT Site: http://www.gwtsite.com
[7] B. W. Perry, “*Google Web Toolkit for Ajax”,*O’Reilly, 2006
[8] P. Chaganti **,** *“Google Web Toolkit: GWT Java AJAX Programming”,* Packt Publishing, 2007
[9] Project GWT Ext http://www.gwt-ext.com
[10] Google Chart API http://www.code.google.com/apis/chart/
[11] B. Sakowicz, M. Wójtowski, P. Zalewski, A. Napieralski: "Problems of Standardization in Web Technologies”, XI Konferencja „ Sieci i Systemy Informatyczne, Łódź, Polska, październik 2003, pp. 111-114, ISBN 83-88742-91-4
[12] M. Zywno, B. Sakowicz, K. Dura, A. Napieralski “*J2EE Design Patterns Applications*” 12 th International Conference MIXDES 2005, Kraków, Poland, 23-25 June, pp. 627 - 630, vol. 1, ISBN 83-919289-9-3
[13] B. Sakowicz, J. Wojciechowski, K. Dura. :„*Metody budowania wielowarstwowych aplikacji lokalnych i rozproszonych w oparciu o technologię Java 2 Enterprise Editon*” Mikroelektronika i Informatyka, maj 2004, KTMiI P.Ł. , pp. 163-168, ISBN 83-919289-5-0
[14] J. Wojciechowski, B. Sakowicz, K. Dura, A. Napieralski: ”*MVC model struts framework and file upload issues in web applications based on J2EE platform*” TCSET’2004, 24-28 Feb. 2004, Lviv, Ukraine, pp., 342-345 , ISBN 966-553-380-0
[15] P. Ziemniak, B. Sakowicz, A. Napieralski: "*Object oriented application cooperation methods with relational database (ORM) based on J2EE Technology*"; CADSM'2007; pp. 327-330, ISBN 978-966-553-587-4,
[16] J. Wojciechowski, J. Murlewski, B. Sakowicz, A. Napieralski, "*Object-relational mapping application in web-based tutor-supporting system*", CADSM, Lviv-Polyana, Ukraine, Feb. 23-26, 2005,pp. 307-310, ISBN 966-553-431-9