TAMK University of Applied Sciences
Programme in Information Technology
Software Engineering
Jarno Välkki

Final Thesis

BSC Emergency Support Tool for Nokia Siemens Networks

Thesis supervisor: Lecturer Tony Torp
Commissioned by: Nokia Siemens Networks
Tampere 9/2010

TAMK University of Applied Sciences
Programme in Information Technology
Software Engineering

Author: Jarno Välkki
Name of the report: BSC Emergency Support Tool for Nokia Siemens Networks
Number of pages: 35
Graduation time: 9/2010
Thesis supervisor: Lecturer Tony Torp
Commissioned by: Nokia Siemens Networks

Abstract

The purpose of this thesis was to design and implement a web-based tool for Nokia Siemens Networks that can be used to insert emergency call cases to a database and view the cases. Other features of the tool include assigning duty periods to support personnel and sending automatic e-mails a few days before a duty period starts. The application also provides certain statistics regarding the calls.

The new tool was made as a replacement for an old Lotus Notes based database so a feature for importing data exported from the old database had to be implemented.

Google Web Toolkit was used to develop the web application, which allowed almost the whole application to be developed using Java as the programming language.

Keywords Java, JavaScript, SQL, Lotus Notes

Tampereen Ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Tekijä: Jarno Välkki
Työn nimi: BSC Emergency Support Tool for Nokia Siemens Networks
Sivumäärä: 35
Valmistumisaika: 9/2010
Työn ohjaaja: Lehtori Tony Torp
Työn tilaaja: Nokia Siemens Networks

Tiivistelmä

Tutkintotyön aiheena oli suunnitella ja toteuttaa Nokia Siemens Networks:lle web-pohjainen työkalu, joka mahdollistaa hätäpuhelujen kirjaamisen ja tarkastelun. Sovelluksen muina osina toteutettiin päivystysvuorojen jakaminen ja automaattinen sähköpostin lähetys muutamaa päivää ennen päivystyksen alkua sekä tilastotoiminto erinäisten tilastojen tarkastelua varten.

Työkalu korvasi vanhan Lotus Notes-pohjaisen ratkaisun, joten yksi osa työn suorittamista oli toteuttaa työkaluun tuonti-ominaisuus, joka mahdollistaa Lotus Notes kannasta viedyn datan tuomisen uuteen työkaluun.

Sovelluksen kehityksessä käytettiin Google Web Toolkitia, joka mahdollisti lähes koko sovelluksen ohjelmoinnin käyttäen Javaa ohjelmointikielenä.

Avainsanat Java, JavaScript, SQL, Lotus Notes

Table of Contents

List of abbreviations and terms

NSN	Nokia Siemens Networks, Employer Company
GWT	Google Web Toolkit, a development toolkit for building and optimizing complex browser-based applications.
RPC	Remote Procedure Call, a mechanism for interacting with a server across a network.
JSNI	JavaScript Native Interface, a feature of GWT allowing integration of JavaScript directly into the application's Java source code
AJAX	Asynchronous JavaScript and XML, a group of web development techniques used on the client-side to create interactive web applications
CSS	Cascading Style Sheets, a language used to style web pages
XML	Extensible Markup Language, a language commonly used to transfer data between software components
HQL	Hibernate Query Language, a language used to perform Hibernate queries
LDAP	Lightweight Directory Access Protocol, a protocol used to access the Enterprise Directory

1 Introduction

Nokia Siemens Networks (NSN) is one of the largest telecommunications companies in the world. At the moment NSN employs over 60 000 people in over 150 countries. NSN has over 600 Communications Service Providers around the world as its customers. /1/, /2/

Until now the company has used a Lotus Notes based system for logging BSC Emergency calls. The purpose of this thesis was to replace that system with a web-based solution. The new web-based application is developed using Google Web Toolkit, which is an open source SDK that enables developers to program modern AJAX web applications using Java as the programming language.

The application was developed mainly using Java but due to some incompatibilities with older browsers, some workarounds had to be implemented using JavaScript. The visual look of the application was enhanced by using CSS to style the application.

This thesis consists of three main parts. The first part (chapters 2-6) introduces the techniques and technologies used in developing the application with some basic examples. The second part (chapter 7) introduces some of the main phases of development and the most important parts of the development process. The last part (chapter 8) gives a brief look into the final application and what it looks like.

2 Google Web Toolkit

Google Web Toolkit (**GWT**) is an open source set of tools that allows web developers to create AJAX applications using Java. It simplifies the development of AJAX applications by allowing developers to quickly build and maintain complex JavaScript front-end applications in the Java programming language./3/, /4/

With the GWT SDK, the application developer can write the application's client side code in the Java programming language which GWT then cross-compiles into optimized JavaScript that works across all major browsers./4/

On the client side code the developer is pretty much restricted to the libraries that GWT offers and native JavaScript. On the server side however, any Java library can be used as the server side code isn't translated into JavaScript but runs as bytecode on the server instead.

The main advantage of both client and server side code being coded in Java is that objects can easily be passed between the client and the server using Remote Procedure Calls. The code is also significantly easier to debug and GWT automatically creates different variations of JavaScript for different browsers so the developer doesn't need to worry that much about compatibility with various browsers.

2.1 Prerequisites for using GWT

To make a GWT application, the following components need to be installed:

1. Java SDK version 1.5 or later.
2. Apache Ant

The Google Web Toolkit SDK can be downloaded from: http://code.google.com/webtoolkit/download.html

2.2 Creating a GWT project from command line

The SDK comes with a command line utility called webAppCreator that can be used to create a GWT project. To create a new project the following command in example 1 can be run from command line:

```
webAppCreator -out WebAppName com.company.webapp.WebAppName
```

Example 1 Creating a new GWT project from command line

The command in example 1 will create the project files under a folder named WebAppName.

2.3 Creating a GWT project using Eclipse

To make developing GWT applications more convenient, Google provides a plugin for Eclipse. Comprehensive installation instructions for the plugin are provided by Google at: http://dl.google.com/eclipse/plugin/3.6. The 3.6 should be substituted with the version of Eclipse used. As always, the latest version is recommended.

Once the plugin is installed to Eclipse, a new GWT project can be created by choosing to create a new “Web Application Project”. The “New” dialog for choosing the project is shown below in Figure 1.

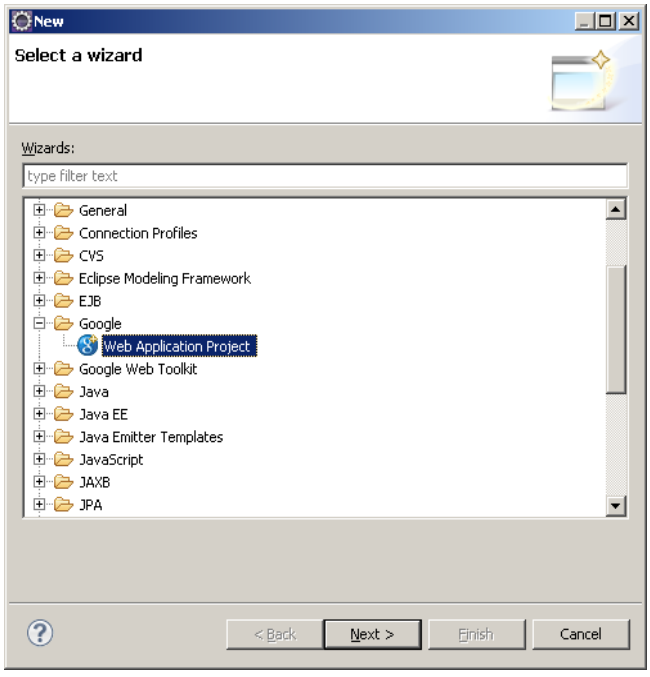


Figure 1 Wizard used to create a new GWT application

On the next screen a name can be chosen for the project and also for the package that will contain the code files. The aforementioned screen can be seen below in Figure 2.

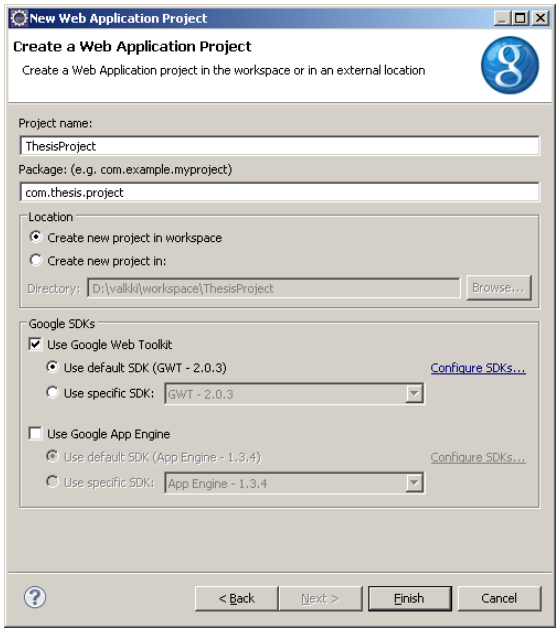


Figure 2 Configuring the project properties

For this project the “Use Google App Engine” can be unchecked as the application is not deployed to the App Engine. Clicking Finish will create the new project.

2.4 Testing the application

While developing the application, it is much more convenient to use the Development Mode to test the application rather than compiling the JavaScript files every time you want to test a new functionality. The application can be run in Development Mode by right clicking the project and choosing Debug as Web Application. This process is illustrated in Figure 3 below.

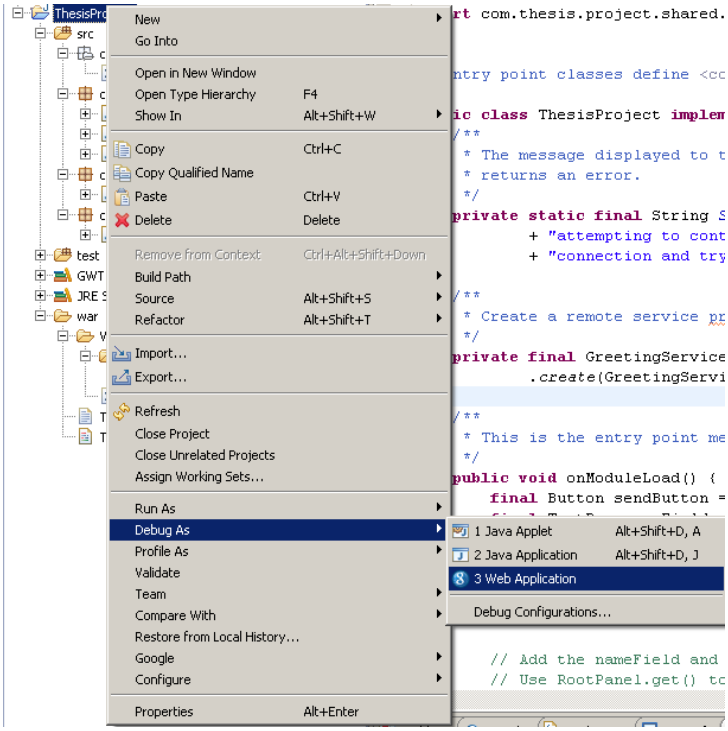


Figure 3 Testing the application in development mode

After a while there will be an URL in the Development Mode tab that can be used to test the application in a browser. The Development mode tab with the address is shown below in Figure 4.

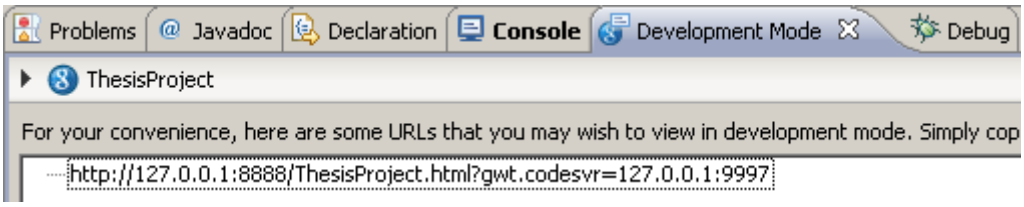


Figure 4 Development mode URL

If the browser hasn't been used before in Development Mode, the browser will prompt the user to install the Google Web Toolkit Developer Plugin. The plugin can be installed by following the instructions provided by the browser. /4/

2.5 Compiling the application for deployment to the server

In order to deploy the application to a server, JavaScript code has to be generated first from the Java code. This can be done by right clicking the project and choosing Google and GWT Compile. The compilation time will vary depending mostly on what browsers and locales the application supports. These can be defined in the module xml file.

If the developer chooses to support 4 browsers, and localizes the application in 4 languages, GWT will generate 16 different permutations of the application at compile-time. At runtime, GWT picks the appropriate version of the application to show the user. /6/

Browser and locale support can be defined by setting the following properties in the GWT module xml.

```
<set-property name="user.agent" value="ie6" />
<set-property name="locale" value="default" />
```

Example 2 Setting browser and locale

The above lines in example 2 will create a default locale version of the application for Internet Explorer 6. If these properties aren't defined in the file, the compiler will compile default locale versions of the application for all the browsers that GWT supports.

The following are the values that can be defined for the user.agent property: **ie6, ie8, gecko, gecko1_8, safari, opera**. Safari is basically any WebKit based browser and gecko1_8 means older versions of Firefox. The other values are pretty self explanatory.

Adding support for other locales than the default "en" can be done by adding the following lines (example 3) to the module xml file:

```
<inherits name="com.google.gwt.i18n.I18N"/>
<extend-property name="locale" values="fi_FI"/>
```

Example 3 Adding a locale

The first line in example 3 defines that the localization module should be used and the latter line defines that the application will support Finnish localization. “fi_FI” can be replaced with any supported locale and multiple lines like the latter can be added to support more languages. To use the localized version of the application, “?locale=fi_FI” should be appended to the end of the URL. Some Java objects like dates are automatically internationalized using locales and the developer can localize the applications messages and constants if they choose to.

When testing the application, it is useful to set the properties to use only one locale and the browser used to test the application with to greatly speed up the compilation process. Once the application is compiled, it can be deployed to a server.

2.6 Deploying the application on Tomcat

If the application uses RPC for communicating with the server, a servlet container has to be used to deploy the application. For the application produced for this thesis, Tomcat was used as the servlet container so that’s the one that will be used as an example on this document. To deploy the application on Tomcat, the contents of the application’s war folder should be copied to a folder in the webapps directory under tomcat’s root directory.

3 Communicating with the server-side code

This chapter introduces 2 ways to communicate with the server-side code when using GWT.

3.1 RPC

GWT provides a mechanism called Remote Procedure Call (RPC) for communicating with the server. The main advantage of using RPC is that a shared serializable class can be implemented that can be used in both client and server side code. This makes passing data between the browser and server very simple and efficient.

3.2 Using RPC

To use RPC a synchronous interface and an asynchronous interface needs to be created on the client side and the synchronous interface implemented on the server side.

3.2.1 Defining the client-side interfaces

Below is an example (example 4) of the synchronous interface class:

```
@RemoteServiceRelativePath("remote")
public interface RemoteServices extends RemoteService {
String insertData(CaseData data);
}
```

Example 4 Synchronous interface class

The synchronous interface class must extend RemoteService and list all the RPC methods. Note that a path must be defined that tells the compiler where the class that implements the interface resides using the RemoteServiceRelativePath annotation.

The asynchronous interface counterpart is shown in example 5 below:

```
public interface RemoteServicesAsync {
void insertData(CaseData data, AsyncCallback<String> callback);
}
```

Example 5 Asynchronous interface class

It is important that the name of the asynchronous interface is otherwise exactly the same as the synchronous interface except that it has the word **Async** appended to it. This is necessary because the GWT compiler depends on the classes being named like this in order to generate the proper code to implement RPC. The asynchronous interface should have all the same methods as the synchronous one but with the method's return value type as void and the actual return value type defined as a type for the AsyncCallback object.

3.2.2 Implementing the RPC interface

The RPC interface is implemented by creating a server side class that extends RemoteServiceServlet and implements the interface defined above in example 5. Below

is an example (example 6) of a server side class that implements the interface defined in the previous chapter:

```
public class RemoteServicesImpl extends RemoteServiceServlet
implements RemoteServices {

public String insertData(CaseData data){

Session session =
HibernateUtil.getSessionFactory().getCurrentSession();
session.beginTransaction();
session.save(data);
session.getTransaction().commit();
return "Data succesfully added to Db";
}
}
```

Example 6 Server side class implementing the remote service

3.3 Request Builder

If the server side code isn't written in Java or if the developer wants to access a regular Java HTTPServlet, GWT's Request Builder can be used which allows communicating with the server using XMLHttpRequests.

To use the RequestBuilder in a GWT application, the GWT HTTP module needs to first be inherited by adding the `<inherits>` tag shown in example 7 below to the module XML file: /4/

```
<inherits name="com.google.gwt.http.HTTP" />
```

Example 7 Inheriting the HTTP module

If the server response is in xml format the XML library can be used to parse it. In order to use the library, the XML module must be inherited by adding the following line (example 8) to the module XML file:

```
<inherits name="com.google.gwt.xml.XML" />
```

Example 8 Inheriting the XML module

3.3.1

An example of using Request Builder

This basic example uses Request Builder to get a generated xml from a Servlet and parses the data from the xml using GWT's xml library. To keep the example simple, the only data contained in the xml is the client's IP address. The example is presented below in examples 9 and 10:

Client side code:

```
public class ThesisProject implements EntryPoint {

private String ip,text;
private Label ipLabel;
private VerticalPanel panel;

public void onModuleLoad() {

panel = new VerticalPanel();
ipLabel = new Label();
panel.add(ipLabel);

RootPanel.get("ip").add(panel);
text = "Your IP address is: ";
ip = "";
String url = GWT.getModuleBaseURL()+"example?format=xml";
RequestBuilder builder = new RequestBuilder(RequestBuilder.GET,
URL.encode(url));

try {
Request request = builder.sendRequest(null, new RequestCallback() {
public void onError(Request request, Throwable exception) {
// No error handling for this basic example
}
public void onResponseReceived(Request request, Response response) {
if (200 == response.getStatusCode()) {
Document xml = XMLParser.parse(response.getText());
ip =
xml.getElementsByTagName("ip").item(0).getFirstChild().getNodeValue();
ipLabel.setText(text + ip);
} else {
// No error handling for this basic example
}
}
});
} catch (RequestException e) {
// No error handling for this basic example
}
}
}
```

Example 9 Client side code of Request Builder example

Server side code:

```
public class ExampleServlet extends HttpServlet{

protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {

String ip = req.getRemoteAddr();
PrintWriter out = resp.getWriter();

if(req.getParameter("format") != null ){
           if(req.getParameter("format").equals("xml"))
           {
           XMLWriter writer = new XMLWriter();
           writer.writeXMLHeader();
           writer.writeElement("","data",XMLWriter.OPENING);
           writer.writeElement("","ip",XMLWriter.OPENING);
           writer.writeText(ip);
           writer.writeElement("","ip",XMLWriter.CLOSING);
           writer.writeElement("","data",XMLWriter.CLOSING);
           out.print(writer.toString());
           }
}
else
{
out.write("<html>");
out.write("<head>");
out.write("<title>Thesis Example</title>");
out.write("</head>");
out.write("<body>");
out.write("This is an example Servlet<br>");
out.write("Your IP is: "+ip);
out.write("</body>");
out.write("</html>");
}
}
}
```

Example 10 Server side code of Request Builder example

The above Servlet has 2 features. If the user accesses it normally using the Servlet's URL they are presented with their IP address in HTML format. If however the Servlet is accessed with the parameter "format" set as xml, the Servlet will provide the user with an xml containing the user's IP address. In the example above (example 9), GWT's Request Builder is used to programmatically fetch the xml and parse the IP from it. The IP is then presented to the user using GWT's label widget.

4 Integrating JavaScript into Java code

Although GWT provides the developer with the ability to program the application using Java, it is still useful in certain situations to be able to write JavaScript code instead. For this purpose GWT offers a mechanism called JavaScript Native Interface (JSNI) which allows you to integrate JavaScript code into Java code.

4.1 Using JSNI

JSNI methods are declared `native` and contain JavaScript code in a specially formatted comment block between the end of the parameter list and the trailing semicolon. A JSNI comment block begins with the exact token `/*-{` and ends with the exact token `}-*/`. JSNI methods are called just like any normal Java method. /4/

The JSNI syntax is a directive to the Java-to-JavaScript Compiler to accept everything inside the comment block as valid JavaScript code and inject it inline in the generated GWT files. /4/

4.1.1 A basic example

Below is a basic example (example 11) of using JSNI to display an alert dialog.

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

Example 11 Using JSNI to display an alert dialog

The above code takes a message string as a parameter and displays it on an alert dialog. When using JSNI the browser's window and document objects can be referred to using $wnd and $doc.

4.2 A real life example of where JSNI was needed

Because one of the criteria for the application was that it has to work with Internet Explorer 6 (ie6), some of the functionalities had to be implemented using custom JavaScript code. One of those functionalities was adding a hover effect for the currently

selected row. For standards compliant browsers this can be done with the following piece of CSS (example 12):

```
.Table tbody tr:HOVER{
 background-color: rgb(200,255,200);
 }
```

Example 12 Adding a hover effect using CSS

For ie6, the following workaround (example 13) was implemented:

```
private native void addRowHover(Element table)/*-{
isIE6 = navigator.userAgent.toLowerCase().indexOf('msie 6') != -1;
   if(isIE6)
   {
   for(var i = 0; i < table.rows.length; i++)
   {
   var rowElem = table.rows[i];
   rowElem.onmouseover = function(){
   this.className='Table-Active';
   };
   rowElem.onmouseout = function()
   {
   if(this.rowIndex % 2 == 0){
   this.className='Table-OddRow';
   }
   else{
   this.className='Table-EvenRow';}
   };
   }
   }
}-*/;
```

Example 13 Hover effect workaround using JavaScript

4.3 Calling a Java method from JavaScript code

Because JavaScript uses dynamic typing and Java uses static typing, you a special syntax must be used to access Java methods from JavaScript. The syntax is demonstrated below in example 14:

```
[instance-expr.]@class-name::method-name(param-signature)(arguments)
```

Example 14 Syntax used to access Java methods from JavaScript

Calling a Java method from JavaScript is better explained with an example in the next chapter.

4.3.1 An advanced example

The following example (example 15) is for demonstration purposes only. There's a much more convenient way to implement this functionality using GWT.

```
private native void addRowClickHandlers(Element table, CaseTable
obj)/*-{

for(var i = 0; i < table.rows.length; i++)
{
var rowElem = table.rows[i];
var cells = rowElem.cells;
for(var j = 0; j < 10; j++)
{
  cells[j].row = i;
  cells[j].onclick = function (e)
           {
target = e?e.target:window.event?window.event.srcElement:null;
if(target)
showTxt(target.row);
}
}
}

function showTxt(num){
obj.@com.nsn.emedb.client.CaseTable::showDetails(I)(num);
           }
}-*/;
```

Example 15 Calling a Java method from JavaScript

The above function in example 15 takes 2 parameters, a table element and an instance of the CaseTable class. It adds click handlers for the first 10 cells of all the rows. When one of the cells is clicked, the row number is passed to the showTxt function which calls a Java method called showDetails which takes a row number integer as a parameter. The param-signature value "I" means that the argument num is an integer. A complete list of available signature values can be found from the JNI Types and Data Structures documentation (5).

5 MySQL and LDAP

In the developed application, MySQL was used to store the input data. Lightweight Directory Access Protocol (LDAP) was used to access the company's Enterprise Directory in order to provide user authentication and searching of callers details.

5.1 MySQL

MySQL is the most popular Open Source SQL database management system. It is developed, distributed, and supported by Oracle Corporation. MySQL uses relational databases to store data in separate tables rather than putting all the data in one big table. This adds speed and flexibility. /8/

MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Several high-profile web sites (including Flickr, Facebook, Wikipedia, Google and YouTube) use MySQL for data storage and logging of user data. /8/, /9/

MySQL was chosen for the application due to its GPL license and fast performance. It was also familiar to me as I had used it in previous projects. MySQL server is available for both Windows and Linux which was helpful during the development process.

To get started with using MySQL, binary distributions of MySQL server for all major operating systems are available from http://mysql.com/. MySQL server installs itself as a daemon on Linux systems and as a service on Windows systems.

MySQL provides connectivity for client applications developed in the Java programming language via a JDBC driver, which is called MySQL Connector/J. /8/ Using Connector/J it is possible to perform SQL queries from Java code.

5.2 LDAP

LDAP (Lightweight Directory Access Protocol) is an Internet protocol used to look up information from a server. LDAP is particularly useful for storing information that is read often from different locations, but updated infrequently /11/. Most large companies have an enterprise directory containing information about their employees and their credentials. LDAP is often used for user authentication as it provides a mechanism to bind to the directory with the given username and password. The bind is successful if the provided username and password match the ones in the directory.

Directories have a tree structure. The top level usually consists of the company's domain components, for example dc=companyName, dc=com but there is no rule about the structure. In this situation the top level of the directory was o=companyName. Under that, the directory is usually divided by organizational units (ou), for example ou=orgUnit, o=companyName. As can be seen from the examples above, the directory entries are read from right to left. In a simple directory an employee's details might be stored under an entry like: uid=idOfEmployee, ou=orgUnit, o=companyName.

For the implemented application, LDAP was used for user authentication and searching for the caller's name. More information about using LDAP in the application can be found from a later chapter.

6 Hibernate

Hibernate is an Object/Relational Mapping tool for Java environments. The term Object/Relational Mapping (ORM) refers to the technique of mapping a data representation from an object model to a relational data model with a SQL-based schema. Hibernate takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types) and provides methods for querying and retrieving data. /7/

Hibernate is licensed under the LGPL license and is available for free from http://www.hibernate.org/.

6.1 Configuring the application to use Hibernate

In order to use hibernate with the application, **hibernate3.jar** and all required libraries from the distribution bundle should be copied to the project's classpath. Hibernate also requires the developer to configure some properties before it can be used. This can be done with a properties file, an xml file or in the Java code. For this document, an example of configuring Hibernate with an xml file will be provided as that is the most popular configuration method.

Hibernate checks for a file named hibernate.cfg.xml when creating a Session Factory, which is used to obtain instances of Hibernate sessions. The configuration file should be located in the project's classpath and should contain something along the lines of the example displayed below in example 16:

```
<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

        <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>

        <property
name="connection.url">jdbc:mysql://localhost/databasename</property>

        <property name="connection.username">user</property>

        <property name="connection.password">pass</property>

     <property
name="connection.provider_class">org.hibernate.connection.C3P0Connecti
onProvider</property> <!-- NECESSARY TO USE C3P0 -->
        <property name="c3p0.acquire_increment">1</property>
        <property name="c3p0.idle_test_period">100</property>
        <property name="c3p0.max_size">100</property>
        <property name="c3p0.max_statements">0</property>
        <property name="c3p0.min_size">10</property>
        <property name="c3p0.timeout">100</property>

      <property
name="dialect">org.hibernate.dialect.MySQLDialect</property>
```

```
<!-- Enable Hibernate's automatic session context management -->
<property name="current_session_context_class">thread</property>
<!-- Disable the second-level cache  -->
<property
name="cache.provider_class">org.hibernate.cache.NoCacheProvider</prope
rty>

<property name="show_sql">true</property>

<property name="hbm2ddl.auto">update</property>

<mapping resource="nameOfMappingFile.hbm.xml"/>

</session-factory>

</hibernate-configuration>
```

Example 16 Hibernate configuration file

The most important thing in the configuration file that needs to be configured is the connection pool. Most basic examples in the Hibernate documentation and in various other resources use Hibernate's built-in connection pool which is fine for development use but not for production use. There's a warning about this in the documentation but it doesn't really explain why the built-in pool shouldn't be used. During testing with the built-in connection pool, it was noticed that the connection to the database would work fine for the whole day but the next morning it wouldn't work anymore without restarting Tomcat.

For the above example, a c3p0 JDBC connection pool was used. The library required to use the c3p0 connection pool is included in the Hibernate distribution bundle. Other important values are the first four property elements, which define the connection properties that should be used and the mapping resource.

6.2 Creating a helper class to use Hibernate

To use Hibernate, a Session Factory should be created that manages Hibernate session instances. To make using Hibernate easier, it is useful to also create a helper class that makes accessing the Session Factory more convenient. The helper class is provided with the Hibernate documentation and can be seen below in example 17.

```
public class HibernateUtil {

private static final SessionFactory sessionFactory =
buildSessionFactory();

private static SessionFactory buildSessionFactory() {

try {
     // Create the SessionFactory from hibernate.cfg.xml
     return new Configuration().configure().buildSessionFactory();
    }
    catch (Throwable ex) {
    System.err.println("Initial SessionFactory creation failed." +
ex);
    throw new ExceptionInInitializerError(ex);
    }
    }

    public static SessionFactory getSessionFactory() {
    return sessionFactory;
    }
}
```

Example 17 Hibernate helper class

To initialize the SessionFactory on Tomcat startup and close it on shutdown, the following ServletContextListener can be implemented (example 18). /10/

```
public class HibernateListener implements ServletContextListener {

public void contextInitialized(ServletContextEvent event) {
HibernateUtil.getSessionFactory(); // Just call the static initializer
of that class
}

public void contextDestroyed(ServletContextEvent event) {
HibernateUtil.getSessionFactory().close(); // Free all resources
}
}
```

Example 18 A listener used to initialize Hibernate's Session Factory on Tomcat startup

To use the listener the following declaration (example 19) must be added to the web.xml file:

```
<listener>
<listener-class>[packageName].HibernateListener</listener-class>
</listener>
```

Example 19 Taking the listener into use

After the helper class is created, Hibernate can be used by accessing the SessionFactory using HibernateUtil.getSessionFactory() and getting the current session with the getCurrentSession() method.

6.3 Using HQL to retrieve objects from the database

Hibernate provides its own language called Hibernate Query Language (HQL) for selecting certain objects from the database. The HQL syntax is pretty much like the normal SQL syntax, only slightly simpler with some convenient extra features. The following example (example 20) is an example of using HQL to retrieve the first 5 EmployeeData objects that match the search term from the database.

```
public ArrayList<EmployeeData> searchEmployeeData(String input) {

ArrayList<EmployeeData> myArr = new ArrayList<EmployeeData>();

Session session =
HibernateUtil.getSessionFactory().getCurrentSession();
session.beginTransaction();
Query q = session.createQuery("from EmployeeData where name
like :search)
.setString("search", input)
.setFirstResult(0)
.setMaxResults(5);

for(Object o : q.list()) {
EmployeeData data = (EmployeeData) o;
myArr.add(data);
}
session.getTransaction().commit();

return myArr;
}
```

Example 20 HQL query

One of the features of HQL is using named bind parameters. They are more convenient than positional parameters, because the same parameter can be present more than once in a query and they make the queries more readable. The above example uses the named parameter ":search" to insert the user's search term into the query. It is also easy to limit the amount of returned objects using the setFirstResult and setMaxResults functions. They also make it easy to implement pagination for the application.

6.4 Using Hibernate's Criteria API

Another way of filtering results with Hibernate is using its Criteria API.

```
Criteria crit = session.createCriteria(CaseData.class);
Disjunction dis = Restrictions.disjunction();
for(String a : values)
dis.add(Restrictions.like(a, searchTerm));
crit.add(dis);
```

Example 21 Using the Criteria API

The code example above (example 21) creates a new Criteria object. A disjunction is created that matches all CaseData objects that have a variable value that equals the "searchTerm" value in the variables defined with the values array. The disjunction is then added as a criterion for the Criteria object. As with HQL, the list function can then be called for the Criteria object to get a List containing the CaseData objects matching the criteria.

7 Implementing the application

Since the application is a replacement for an older tool, one of the requirements was that it should be quite similar to the one used before so it would be easy for people to adapt to using the new application. This chapter presents some key phases of the development process. Due to the more complex nature of the cases class, the code examples used in the chapter are taken from the shifts class. It is assumed you have read the earlier chapters and have a basic understanding of how the components used for developing the application work.

7.1 Designing the Database layout

The database layout was kept pretty similar to the one that was used before to make importing the data from the old Lotus Notes Database as easy as possible. Only a few unnecessary fields were removed and a couple useful fields like a separate field for a case id was added. The layout of the database is presented below in Figure 5.

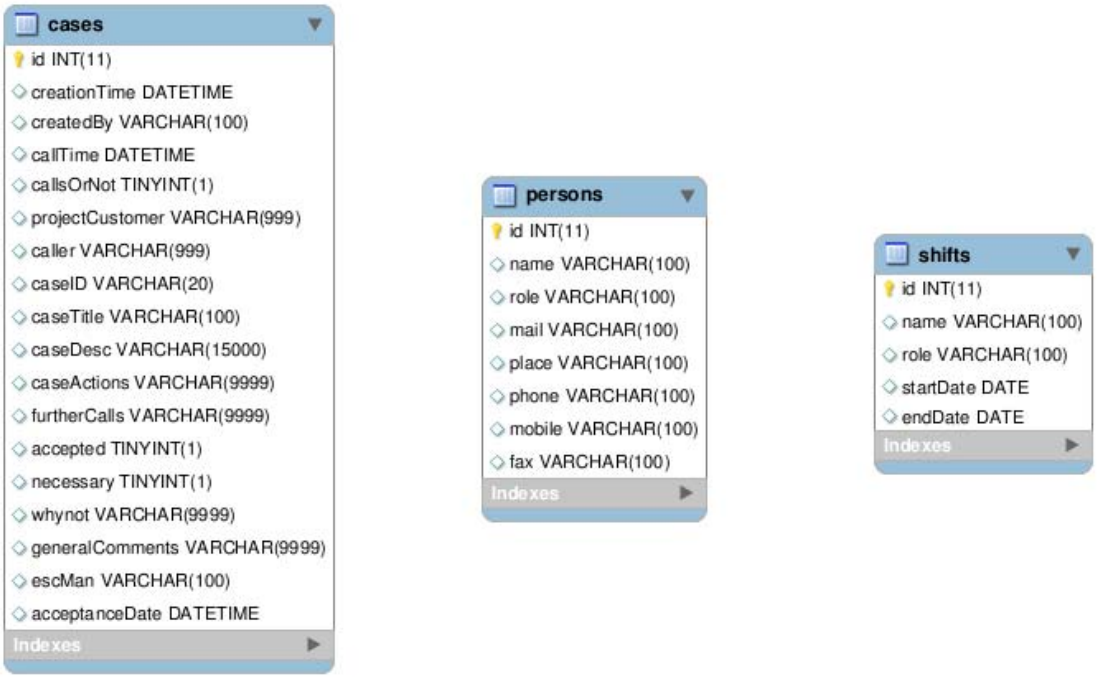


Figure 5 Database layout

7.2 Transferring the old data to the new database

The importing process for the data from the old database was pretty straightforward for support personnel and duties as the old data could be exported as comma-separated values (CSV). The importing of the data to the case table on the other hand required significantly more effort as some of the data in the old database was contained in Rich Text fields and Lotus Notes wasn't able to export those fields into the CSV format. The only format that had all the fields was "Structured text" and the output of the cases in that format looked something like on example 22 below:

```
ECRRole:  [Manager]
ECRProtection:  Yes
ECRState:  ACCEPTED
RepType:  CallOut Report
ECRDate:  06/28/2004 10:56:21 AM
ECRCreated:  CN=Name/OU=TR1/OU=NTC/O=CompanyName
ECRCdate:  06/24/2004
ECRCtime:
ECRCallsNoCalls:  Calls
ECRCustomer:  Customer
ECRCaller:  Caller
ECREscalationMgrAccepted:  Yes
ECRCallNecessary:  No
ECRGeneralComment:
ECREscalationManager:  EscMan
ECREscalationDate:  28.06.2004 12:22:54
$UpdatedBy:
CN=name/OU=TR1/OU=NTC/O=CompanyName,CN=name/OU=TR1/OU=NTC/O=CompanyNa
me
$Revisions:  06/28/2004 11:03:15 AM,06/28/2004 12:24:14 PM

All Rich Text Fields here. No way of telling
which field the text belongs to.
```

Example 22 Structured text output

The data in this form was much harder to parse. The dates and times had no consistency as they were in at least 4 different formats and even different formats were used on a particular case. Another problem was that all text from the “Rich Text” fields was located at the end of the case details and there was no way of knowing which field the text originally belonged to. For that reason all “Rich Text” data in the old database was dumped to the general description field of the case in the new database.

7.3 Creating classes used to transfer data between client and server

To make it easy to pass data between the client and server code, shared classes for cases, persons and duties were created. These classes contain private members for all the database field values and getters and setters for the members. In addition, a constructor is provided that can be used to set all the values. It is important to note that for these types of classes GWT requires the presence of a default zero parameter constructor. It can; however be just an empty constructor. In order for the classes to work with RPC, they have to implement the IsSerializable interface.

Below is the class used for transferring the shift table data as an example (example 23).

```
public class ShiftData implements IsSerializable{

private String name = "";
private String role = "";
private Date startDate = null;
private Date endDate = null;
private int id = 0;

public ShiftData() {

}

public ShiftData(String name, String role, Date startDate, Date
endDate) {
            super();
            this.name = name;
            this.role = role;
            this.startDate = startDate;
            this.endDate = endDate;
            }

            public int getId() {
                        return id;
            }
```

```
	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}


	public String getRole() {
		return role;
	}


	public void setRole(String role) {
		this.role = role;
	}

	public Date getStartDate() {
		return startDate;
	}

	public void setStartDate(Date startDate) {
		this.startDate = startDate;
	}

	public Date getEndDate() {
		return endDate;
	}

	public void setEndDate(Date endDate) {
		this.endDate = endDate;
	}
}
```

Example 23 Shared class used to transfer data between client and server side code

7.4 Creating Hibernate mapping files

In order to use hibernate to save objects to the database; mapping files must be created for the objects. The mapping files tell Hibernate what table in the database it has to access and what columns in that table it should use /7/. The mapping file for shifts is illustrated below in example 24.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
```

```
  <class name="com.nsn.emedb.shared.ShiftData" table="shifts">
   <id name="id" type="int" column="id" >
   <generator class="native"/>
  </id>
  <property name="name">
     <column name="name" />
  </property>
  <property name="role">
    <column name="role"/>
  </property>
  <property name="startDate">
    <column name="startDate"/>
  </property>
    <property name="endDate">
    <column name="endDate"/>
  </property>
 </class>
</hibernate-mapping>
```

Example 24 Hibernate mapping file

In the mapping file, the property name value is the variable name in the class and the column name is the name of the column in the database. For id, generator class is declared as native because the database takes care of auto-incrementing the id.

7.5 Inserting a new duty period to the database

This chapter demonstrates one of the features of the application, adding a new duty using Hibernate. When the user clicks the add button on the “Add new duty period” dialog in the application, a new ShiftData object is created using the supplied data. The dialog can be seen in Figure 6.

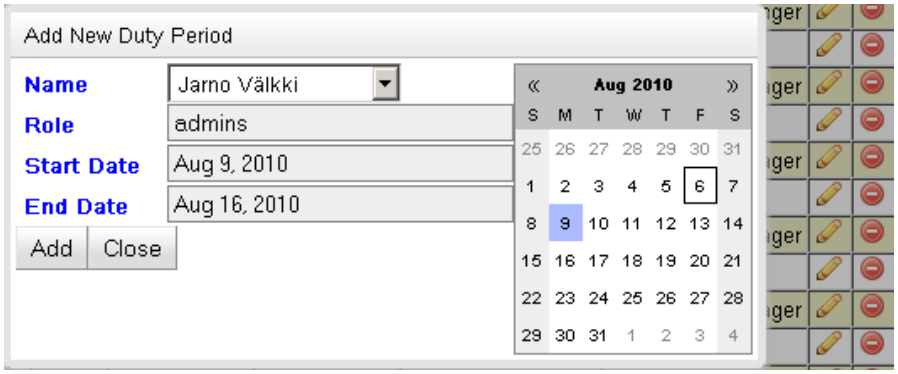


Figure 6 Inserting a new duty

After the object is created, it is sent to the server using RPC. Calling the RPC method is demonstrated below in example 25.

```
ShiftData dat = new
ShiftData(form.getEmployee(),form.getTextBox(0).getText(),form.getStar
tDatePicker().getValue(),form.getEndDatePicker().getValue());
new RemoteServiceCaller().getServiceObject().insertShiftData(dat, new
AsyncCallback< String >() {
public void onFailure(Throwable caught) {}
public void onSuccess(String result){
par.getDataFromShiftDB(par.getSearchTerm(),
par.getSortedBy(),par.getOrder());
}
});
```

Example 25 Client side code of adding a duty period

On the server side a method named insertShiftData is called with the ShiftData object as its parameter. The method is shown below (example 26).

```
public String insertShiftData(ShiftData data) {

Session session =
HibernateUtil.getSessionFactory().getCurrentSession();
session.beginTransaction();
session.save(data);
session.getTransaction().commit();

return "success";
}
```

Example 26 Server side code saving the duty period to the database

The above method gets the current Hibernate session and saves the data contained in the ShiftData object to the database.

7.6 User authentication

User authentication for the application is performed using LDAP to bind to the Enterprise Directory. If the bind is successful, the user is allowed to access the application. Tomcat provides a convenient way to use LDAP for authentication by using the JNDIRealm. To configure Tomcat to use the JNDIRealm, the following properties are defined in server.xml (example 27).

```
<Realm className="org.apache.catalina.realm.JNDIRealm"
   connectionURL="ldap://server.example.com:389"
   userBase="o=companyName"
   userSubtree="true"
   userSearch="(uid={0})"
   allRolesMode = "authOnly"
/>
```

Example 27 Configuring Tomcat to use LDAP for user authentication

The definition above defines a “connectionURL” value that is the directory server’s address. “userBase” is the branch used for searching for the user with the uid (login name) defined in the “userSearch” property. The {0} acts as a placeholder for the user inputted username. “userSubtree” instructs the search to search all sub levels for the user if it’s set to true. Finally, the “allRolesMode” value “authOnly” means that the directory is only used for authentication and that the user isn’t assigned to any groups from the directory.

To make the application use the JNDIRealm for authentication, the following must be defined in the application’s web.xml (example 28):

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Database App</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>*</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Database App</realm-name>
</login-config>
```

Example 28 Making the application use the authentication method defined in server.xml for authentication

The piece of xml above defines that accessing any file inside the application’s folder requires the user to be authenticated. The authentication method used in the example above is the basic authentication that pops up a dialog requesting username and password the first time a protected resource is accessed. This authentication is valid

until all browser windows/tabs are closed and there's no way to log the user out when using this type of authentication.

The implemented application uses form based authentication using JavaServer Pages. One page is used to provide the login logic and the other is used for logging out by invalidating the current session. When using form based authentication with login/logout options there are some important things that have to be taken into account. It isn't enough to just invalidate the session when the user logs out of the application, as the application can still be accessed by pressing the back button in the browser. The application itself should check if the user is authenticated by checking the session or the Remote User header on page load.

7.7 Assigning a role to the user

In order to determine what features of the application the user is allowed to access, the user is assigned a role after signing in to the application. This is done by checking the support persons list for an entry containing the details of the currently logged in user. If the user is on the list, additional functionalities will be enabled in the application based on the role. The basic user roles are Escalation managers and Technical support persons. Managers have the ability to add new support persons to the list.

7.8 Setting up automatic backup of database data

It was decided that due to the low number of new cases per week, it would be enough to perform weekly backups of the database data. Backing up of the data is performed using the mysqldump utility that comes with MySQL. Automation of the backup process was done using a linux shell script and cron (a time based scheduler for linux) was used to schedule the weekly execution of the script.

The script used is shown below in example 29:

```
#!/bin/sh
mysqldump --add-drop-table -uuser -ppass eme > /root/emedb/backup_`date
+%Y_%m_%d`.sql;
```

Example 29 Shell script used to backup database data

To add the script to the scheduler, in terminal the following command should be issued:

crontab –e

The above command will open the file used to define the scheduler's jobs in the default text editor. The file should be edited to contain something like shown in example 30:

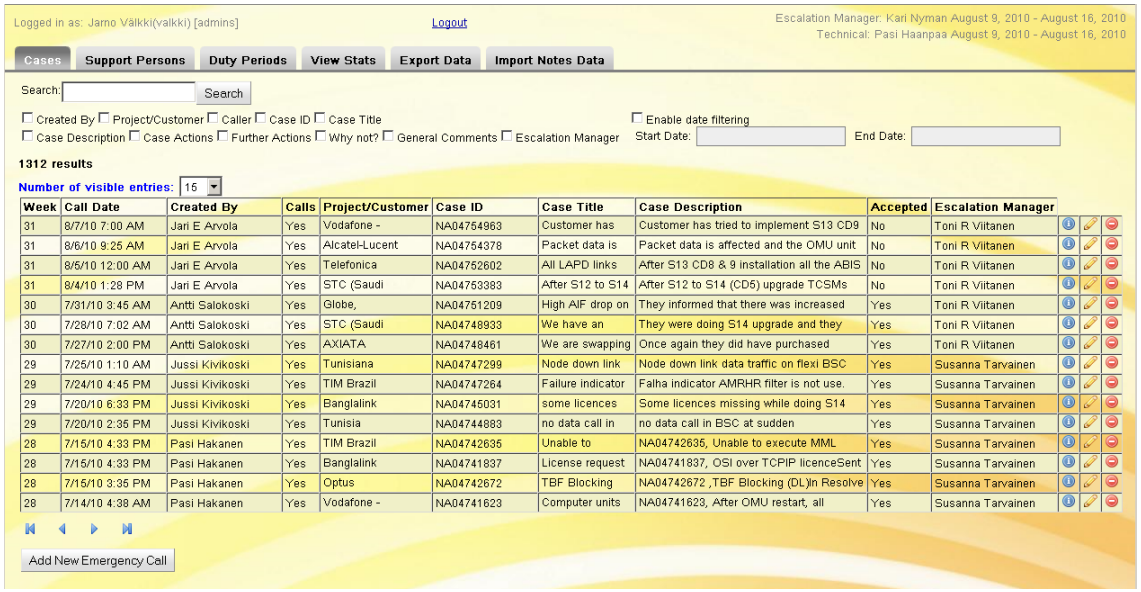
```
0 5 * * 0 /root/emedb/backup.sh
```

Example 30 Cron job used to schedule the weekly backup

The line above tells the scheduler to execute the script every Sunday at 5:00.

8 The finished application

The final look of the developed application is demonstrated below in Figure 7.



Week	Call Date	Created By	Calls	Project/Customer	Case ID	Case Title	Case Description	Accepted	Escalation Manager
31	8/7/10 7:00 AM	Jari E Arvola	Yes	Vodafone -	NA04754963	Customer has	Customer has tried to implement S13 CD9	No	Toni R Viitanen
31	8/6/10 9:25 AM	Jari E Arvola	Yes	Alcatel-Lucent	NA04754378	Packet data is	Packet data is affected and the OMU unit	No	Toni R Viitanen
31	8/5/10 12:00 AM	Jari E Arvola	Yes	Telefonica	NA04752602	All LAPD links	After S13 CD8 & 9 installation all the ABIS	No	Toni R Viitanen
31	8/4/10 1:28 PM	Jari E Arvola	Yes	STC (Saudi	NA04753383	After S12 to S14	After S12 to S14 (CD5) upgrade TCSMs	No	Toni R Viitanen
30	7/31/10 3:45 AM	Antti Salokoski	Yes	Globe,	NA04751209	High AIF drop on	They informed that there was increased	Yes	Toni R Viitanen
30	7/28/10 7:02 AM	Antti Salokoski	Yes	STC (Saudi	NA04748933	We have an	They were doing S14 upgrade and they	Yes	Toni R Viitanen
30	7/27/10 2:00 PM	Antti Salokoski	Yes	AXIATA	NA04748461	We are swapping	Once again they did have purchased	Yes	Toni R Viitanen
29	7/25/10 1:10 AM	Jussi Kivikoski	Yes	Tunisiana	NA04747299	Node down link	Node down link data traffic on flexi BSC	Yes	Susanna Tarvainen
29	7/24/10 4:45 PM	Jussi Kivikoski	Yes	TIM Brazil	NA04747264	Failure indicator	Falha indicator AMRHR filter is not use.	Yes	Susanna Tarvainen
29	7/20/10 6:33 PM	Jussi Kivikoski	Yes	Banglalink	NA04745031	some licences	Some licences missing while doing S14	Yes	Susanna Tarvainen
29	7/20/10 2:35 PM	Jussi Kivikoski	Yes	Tunisia	NA04744883	no data call in	no data call in BSC at sudden	Yes	Susanna Tarvainen
28	7/15/10 4:33 PM	Pasi Hakanen	Yes	TIM Brazil	NA04742635	Unable to	NA04742635, Unable to execute MML	Yes	Susanna Tarvainen
28	7/15/10 4:33 PM	Pasi Hakanen	Yes	Banglalink	NA04741837	License request	NA04741837, OSI over TCPIP licenceSent	Yes	Susanna Tarvainen
28	7/15/10 3:35 PM	Pasi Hakanen	Yes	Optus	NA04742672	TBF Blocking	NA04742672 ,TBF Blocking (DL)In Resolve	Yes	Susanna Tarvainen
28	7/14/10 4:38 AM	Pasi Hakanen	Yes	Vodafone -	NA04741623	Computer units	NA04741623, After OMU restart, all	Yes	Susanna Tarvainen

Figure 7 Case view of application

The picture above shows the case view of the application. The user can search for cases from the database using the controls provided and view their details by clicking on the case. Support personnel can also add new cases to the database. Editing and deleting of cases is available to the creator of the case until it has been accepted and to the managers. Cases can also be sorted by a column by clicking the respective column and the sorting can be inverted by clicking the column a second time.

Adding a new case can be done by clicking the appropriate button in the case view. After the user clicks the button, a form like the one illustrated in Figure 8 will be presented to them.

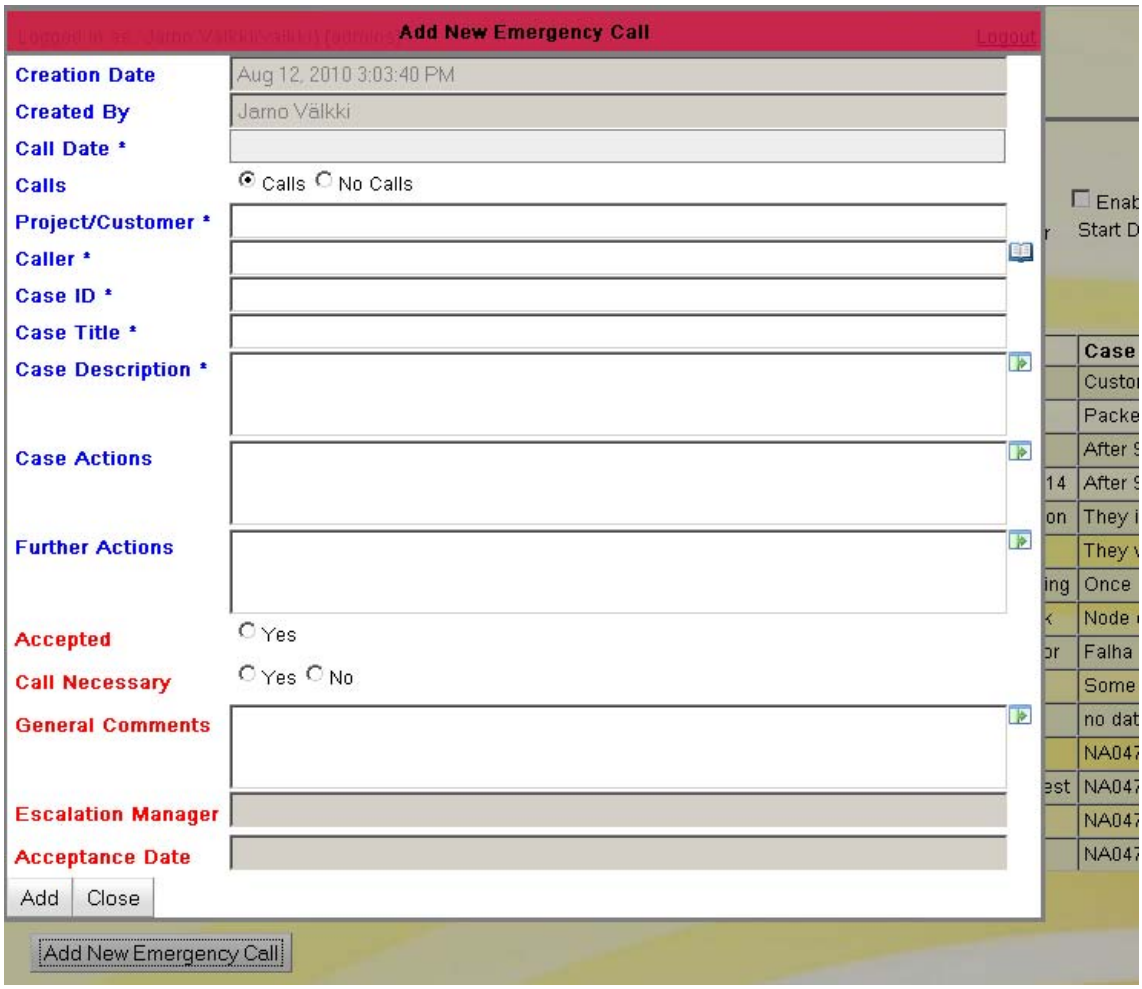


Figure 8 Dialog for adding a new case

Normal technical support personnel will only see the fields marked with blue labels and those are the fields that should initially be filled when creating a case. The fields labeled with red labels are for the Escalation managers to fill later when they review the case. When the user has filled all of the required fields (marked with an asterisk), pressing the add button will add the new case to the database and update the case view.

The stats view can be used to view stats regarding the cases. The stats view is illustrated below in Figure 9.

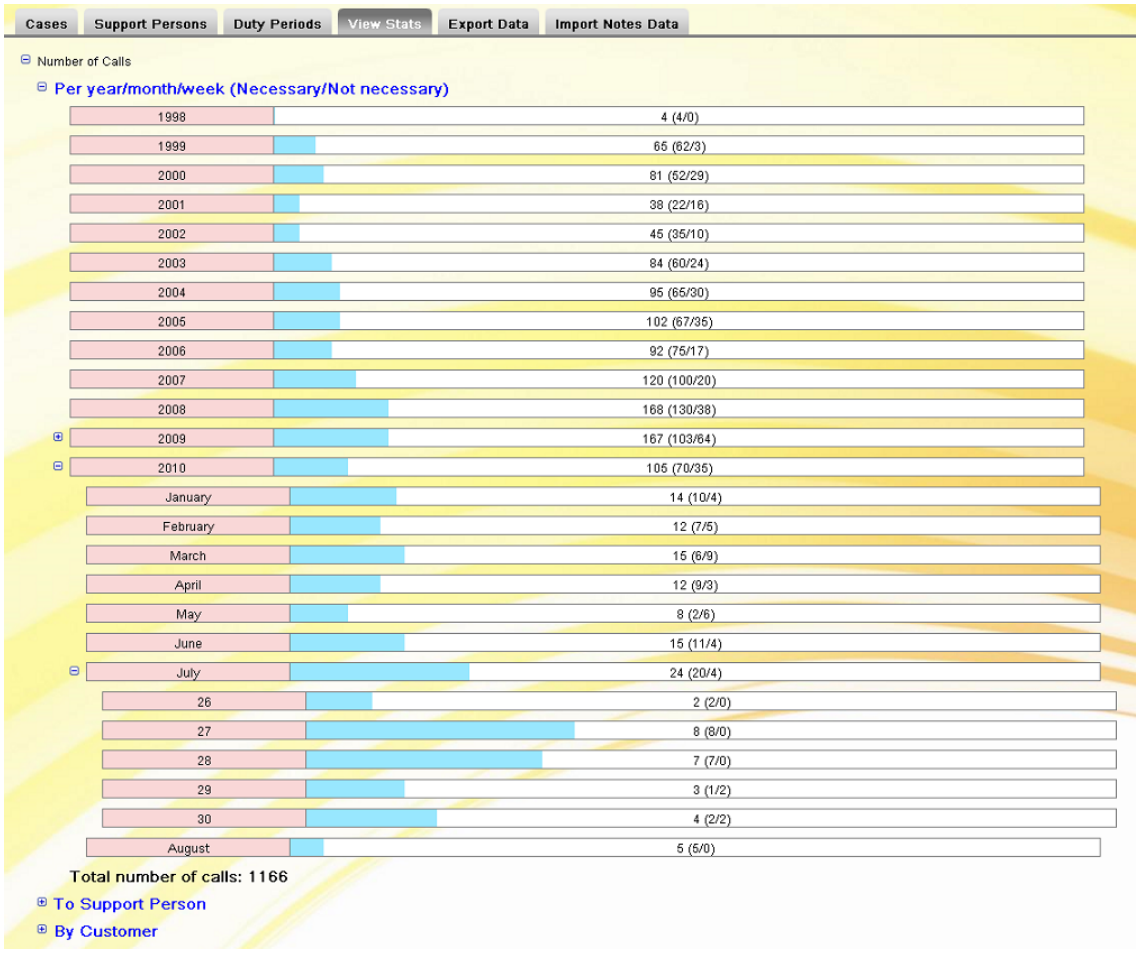


Figure 9 Stats view

A tree structure was used to display the statistics. Call statistics can first be seen by year. Then the user can choose a year to see monthly stats for that year. If the user clicks on a month, weekly stats will be displayed from that month.

9 Conclusion

Google Web Toolkit offers a convenient way to develop modern web applications like the one produced for this thesis. GWT is being actively developed by Google and has an active community providing custom widgets and extensions for it. The hardest challenge in developing web-based applications is that the developer has to take into account different browsers used by the users. GWT does a good job at maximizing compatibility with different browsers although it is self-explanatory that very old browsers will never be able to support the modern technology used by GWT.

Using Hibernate in conjunction with the GWT application was fairly straightforward and it provided a fast and reliable way to interact with the database. The most important thing with Hibernate is to make sure it’s configured properly; otherwise long-term reliability might be compromised.

Developing and deploying the application has given me a great deal of hands-on experience with web-based applications. The thesis work also allowed me to get familiar with the Red Hat Enterprise Linux operating system.

The application was taken into use in the beginning of August. It provides a good base that can easily be improved with more features if needed.

References

1. Nokia Siemens Networks Fact Sheet [PDF file] [referred to 2.8.2010] Available: http://www.nokiasiemensnetworks.com/sites/default/files/document/Fact_sheet_Jan_2010_final.pdf

2. Nokia [online] [referred to 2.8.2010] Available: http://www.nokia.fi/nokia/lehdisto/tiedotteet/tiedotteet?newsid=1116427

3. Google Web Toolkit (Wikipedia) [online] [referred to 2.8.2010] Available: http://en.wikipedia.org/wiki/Google_Web_Toolkit

4. Google Web Toolkit Official Documentation [online] [referred to 2.8.2010] Available: http://code.google.com/webtoolkit/

5. JNI Types and Data Structures [online] [referred to 3.8.2010] Available: http://download-llnw.oracle.com/javase/1.5.0/docs/guide/jni/spec/types.html#wp276

6. Introduction to GWT [online] [referred to 5.8.2010] Available: http://developerlife.com/tutorials/?p=80

7. Hibernate documentation [online] [referred to 6.8.2010] Available: http://docs.jboss.org/hibernate/stable/core/reference/en/html/

8. MySQL Reference Manual [online] [referred to 9.8.2010] Available: http://dev.mysql.com/doc/refman/5.1/en/

9. MySQL (Wikipedia) [online] [referred to 9.8.2010] Available: http://en.wikipedia.org/wiki/MySQL

10. Using Hibernate with Tomcat [online] [referred to 9.8.2010] Available: http://community.jboss.org/wiki/UsingHibernatewithTomcat

11. Introduction to LDAP [online] [referred to 10.8.2010] Available: http://www.ldapman.org/articles/intro_to_ldap.html