IMPLEMENTATION OF MAPLE, A BROWSER-BASED STAFF MANAGEMENT APPLICATION USING GOOGLE WEB DEVELOPMENT TOOLKIT AND SENCHA GXT APPLICATION FRAMEWORK

A graduate project submitted in partial fulfillment of the requirements
For the degree of Master of Science in
Software Engineering

By
Sevak Asadorian

May 2013
The graduate project of Sevak Asadorian is approved:

Prof. Peter Gabrovsky

Prof. Robert McIlhenny

Prof. Richard Covington, Chair

California State University, Northridge
DEDICATION

Dedicated to my family and many friends, who have provided so much support and encouragement over the years.
ACKNOWLEDGMENT

The author wishes to express gratitude to his family for supporting him with his studies throughout the years. To his former professor and current graduate chair, Professor Richard Covington for being supportive and patient. To his former professors and current committee members, Prof. Peter Gabrovsky and Prof. Robert McIlhenny, for their willingness to be on his committee. Lastly, to his Graduate Coordinator, Dr. Ani Nahapetian, for helping him stay on schedule with critical deadlines.
# TABLE OF CONTENTS

Copyright Page ........................ ii
Signature Page ........................ iii
Dedication ............................ iv
Acknowledgement ..................... v
List of Tables ........................ ix
List of Illustrations ................. x
List of Abbreviations ............... xii
Abstract ............................. xiii

Introduction .......................... 1
  GWT .................................. 1
  Sencha GXT .......................... 2
  Background .......................... 3
  Trade Study ........................ 5
  Literature ........................... 8
Requirements ........................ 11
  Features ............................ 11
  Create Staff Plan .................. 11
  Delete Staff Plan .................. 11
  Edit Staff Plan .................... 11
  Department Allocation Report .... 11
  Import ................................ 11
  Export Maple data .................. 11
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Future Enhancements</td>
<td>12</td>
</tr>
<tr>
<td>Mobile Alerts</td>
<td>12</td>
</tr>
<tr>
<td>Artifact List</td>
<td>12</td>
</tr>
<tr>
<td>Additional Reports</td>
<td>12</td>
</tr>
<tr>
<td>Source of Import/Export</td>
<td>12</td>
</tr>
<tr>
<td>Login</td>
<td>12</td>
</tr>
<tr>
<td>Access/Permissions</td>
<td>12</td>
</tr>
<tr>
<td>Action Items Database</td>
<td>12</td>
</tr>
<tr>
<td>Design</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>15</td>
</tr>
<tr>
<td>Database Schema and Tables</td>
<td>18</td>
</tr>
<tr>
<td>Software Design</td>
<td>21</td>
</tr>
<tr>
<td>UML</td>
<td>21</td>
</tr>
<tr>
<td>Class Diagrams</td>
<td>21</td>
</tr>
<tr>
<td>Use Case Diagrams</td>
<td>24</td>
</tr>
<tr>
<td>Sequence Diagram</td>
<td>25</td>
</tr>
<tr>
<td>Implementation</td>
<td>27</td>
</tr>
<tr>
<td>Implemented</td>
<td>27</td>
</tr>
<tr>
<td>Tools</td>
<td></td>
</tr>
<tr>
<td>Installation</td>
<td>30</td>
</tr>
<tr>
<td>Create First Project</td>
<td>33</td>
</tr>
<tr>
<td>Demonstration</td>
<td>43</td>
</tr>
<tr>
<td>Create Staff Plan</td>
<td>44</td>
</tr>
<tr>
<td>Update Employees</td>
<td>46</td>
</tr>
</tbody>
</table>
Update Programs 47
Criteria for Success 49
Lessons Learned 50
Development Error Troubleshooting 51
Conclusion 53
References 56
<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>19</td>
</tr>
<tr>
<td>Table 2</td>
<td>27</td>
</tr>
<tr>
<td>Table 3</td>
<td>35</td>
</tr>
<tr>
<td>Table 4</td>
<td>51</td>
</tr>
</tbody>
</table>
LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Image</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image 1</td>
<td>5</td>
</tr>
<tr>
<td>Image 2</td>
<td>7</td>
</tr>
<tr>
<td>Image 3</td>
<td>15</td>
</tr>
<tr>
<td>Image 4</td>
<td>16</td>
</tr>
<tr>
<td>Image 5</td>
<td>17</td>
</tr>
<tr>
<td>Image 6</td>
<td>18</td>
</tr>
<tr>
<td>Image 7</td>
<td>20</td>
</tr>
<tr>
<td>Image 8</td>
<td>22</td>
</tr>
<tr>
<td>Image 9</td>
<td>23</td>
</tr>
<tr>
<td>Image 10</td>
<td>25</td>
</tr>
<tr>
<td>Image 11</td>
<td>26</td>
</tr>
<tr>
<td>Image 12</td>
<td>33</td>
</tr>
<tr>
<td>Image 13</td>
<td>34</td>
</tr>
<tr>
<td>Image 14</td>
<td>35</td>
</tr>
<tr>
<td>Image 15</td>
<td>37</td>
</tr>
<tr>
<td>Image 16</td>
<td>37</td>
</tr>
<tr>
<td>Image 17</td>
<td>37</td>
</tr>
<tr>
<td>Image 18</td>
<td>37</td>
</tr>
<tr>
<td>Image 19</td>
<td>38</td>
</tr>
<tr>
<td>Image 20</td>
<td>39</td>
</tr>
<tr>
<td>Image 21</td>
<td>40</td>
</tr>
<tr>
<td>Image 22</td>
<td>40</td>
</tr>
<tr>
<td>Image 23</td>
<td>41</td>
</tr>
<tr>
<td>Image 24</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>42</td>
</tr>
</tbody>
</table>
LIST OF ABBREVIATIONS

app - Application

COTS - Commercial Off The Shelf

IDE - Integrated Development Environment (i.e. Eclipse)

Java- Java programming language

GUI - Graphical User Interface

GWT - Google Web Toolkit

GXT - Sencha’s application framework for GWT

SDK - Software Development Kit

API - Application Programming Interface

JavaScript - Interpreted computer programming language
ABSTRACT

IMPLEMENTATION OF MAPLE, A BROWSER-BASED STAFF MANAGEMENT APPLICATION USING GOOGLE WEB DEVELOPMENT TOOLKIT AND SENCHA GXT APPLICATION FRAMEWORK

By

Sevak Asadorian

Master of Science in Software Engineering

Software projects today are increasing in size and complexity. Planning and leading software projects can be challenging and require staff planning, monitoring and controlling. Software organizations can have multiple projects executed at one time and multiple people working on those projects. Software managers are responsible for managing the projects as well as the staff working on those projects. The web-based application, Maple, was implemented to help software organizations manage projects as well as the people working on those projects. The Maple application is capable of creating, editing and managing staff plans. The application keeps track of the staff and the number of hours allocated for each staff member for a specific project. Maple was developed using the Java programming language, Google’s Web Development Toolkit (GXT) and Sencha Application Framework for Google Web Toolkit (Sencha GXT). GWT is a free, open source Google product containing a development toolkit which is used for building optimized browser-based applications. GXT is a layer on top of GWT that provides high-performance widgets, richer templates, layouts, charting, themes, and
much more. Many contributions have been made toward GWT and Sencha GXT which will be discussed later in detail. The setup of the development platform is described, including the tools and packages used to develop the application.
INTRODUCTION

The purpose of this study is to produce a complex, rich and optimized GUI application called Maple. Maple is a management staffing tool that was developed using Google’s Web Toolkit (GWT) and Sencha’s Application Framework for Google Web Toolkit (Sencha GXT). GWT SDK includes the core Java API libraries and Widgets which allow you to develop client-side applications in Java and deploy them as JavaScript. JavaScript deployment will allow the application to run across all browsers (including mobile phones and tablets). GXT software application framework offers high performance UI Widgets which help accelerate development and increase productivity. In the following sections I will present the results of a trade study, which investigates commercial uses of GWT, a literature search of GWT development efforts, and more technical detail about GWT and GXT.

GWT

GWT is a free, open source Google product containing a development toolkit that is used for building and optimizing complex browser-based applications. The toolkit is targeted toward developers who are interested in building web applications without the knowledge of browser technologies such as JavaScript. Java is the one and only language required for building applications using GWT. Google applications like Google Wave and AdWords were developed using GWT.

The beauty behind GWT is that it allows web developers to create applications in Java and compile the source into highly optimized JavaScript. Once the application is converted from Java into JavaScript, it will have the ability to run across all browsers (Chrome, FireFox, Internet Explorer, Safari), including mobile browsers for Android and the iPhone.
Another advantage of using GWT from the application developer’s point of view is its ability to generate updated JavaScript on the fly. In a typical development cycle, the developer writes Java source, then the GWT tool translates the source into JavaScript which is then loaded into the browser. While in developer’s mode, within the GWT IDE, the developer can modify and save the Java source, which triggers the IDE to re-generate the JavaScript. Then, the latest changes can be viewed upon refreshing the browser. Because of features like this, GWT is an excellent application prototyping tool.

As mentioned before, the developer uses GWT to create optimized Web Applications. The term “optimized” is an overloaded term. Here, it refers to the execution speed of the resulting JavaScript code. GWT uses two main approaches to help with optimization. The first approach is a series of code optimization techniques, including:

- Method Inlining
- Removal of unused code
- String optimization

The second approach is based on optimizing the source code files by splitting them into multiple JavaScript fragments which can reduce application startup times.

**Sencha GXT**

Sencha GXT is a high-performance application framework. This framework acts as a software layer on top of the GWT components and widgets, enhancing their usability and performance. Sencha GXT provides high-performance widgets, richer templates, layouts, charting, themes, and much more. Another advantage of using GXT is that it provides fully themed and customizable widgets that are usable with little need for programmer customization. Some examples of the widgets provided by the GXT library are:
GWT has been available for a few years, while Sencha GXT is more recent. A quick survey of web-based apps will reveal many commercial GWT-based apps. Sencha GXT is still undergoing active development, and releases are not as solid and mature. Presently, there are not a lot of commercially available apps that combine GWT and Sencha GXT. Consequently, I felt that the development of Maple would be a good opportunity to explore how well these two technologies could work together.

**Background**

My current employer inspired me to develop the application Maple. For approximately two years, I worked for the Navigation Systems Division of Northrop Grumman in Woodland Hills, CA. The management staff in the Embedded Software Engineering department had a series of Microsoft Excel spreadsheets that were used to keep track of software metrics, defects, the staff and all active projects. One particular spreadsheet was used to keep track of all the staff members and their assigned projects. Each staff member was assigned to one or more projects and a certain development time was allocated to each project. A manager, a team leader, and one or more software engineers were also assigned to each of those projects.

There are several issues with using a Microsoft Excel spreadsheet and Visual Basic for Applications (macros) as if it were a desktop or web application. For example, the resulting application is bound to Microsoft and their licensing agreement. Macros are difficult to maintain and they are not normally considered an element in a formal
application development framework. Not to mention that source code is not very robust when written in VBA and the macros require permissions from the user to execute each time the spreadsheet is loaded.

Lastly, macros are not portable, since they only execute in the context of the Microsoft Excel application. Without licensing fees, your macros and so called “application” will be considered useless. There were times where Microsoft would release updates for Microsoft Excel and the macros would break as library functions became deprecated. At times, this would require a major redesign effort.

For the reasons stated above, I was inspired to design and develop a replacement to these spreadsheets that I named Maple. Moreover, I believed that the Excel spreadsheets lacked certain features that would have improved the staff plans. My company never had the opportunity, time, or budget to implement these features. I describe these features in detail later in the requirements section.

Realizing that Excel spreadsheets were an inadequate solution, I researched several alternative toolkits and application development frameworks that would be a good fit for Maple’s requirements. After some research, I came upon GWT and Sencha GXT. Both of these applications showed high potential in meeting Maple's requirements. GWT seemed like an interesting approach because, currently, there are several real world projects that have been developed using GWT. In addition to the Google applications that use it, I found Scenechronize (www.scenechronize.com), a production management company within the entertainment industry, some of whose web applications were developed with GWT.

GWT also supports HTML5 features. The following is a list of features that are supported by GWT:
- Client-side storage - use local storage, monitor StorageEvents
- Canvas - add visualization using canvas
- Audio - use audio widgets
- Video - use video widgets

The following link will take you to a webpage which will provide a demonstration of HTML5 canvas within a GWT application:

http://gwtcanvasdemo.appspot.com/

The following image is a snapshot of the HTML5 canvas demo:

![GWT Canvas Demo]

Trade Study

Before developing Maple, I did a trade study to find out about existing applications within the same application domain. One commercial off the shelf (COTS) software app that is similar to Maple is ResourceGuru (resourceguruapp.com). ResourceGuru is a more general resource scheduler that works for a wide variety of
resources (i.e. software, hardware, vehicles, conference rooms, etc.) for as long as
needed. Maple, on the other hand, is specifically designed for managing staff plans. It is
intended to provide more specific capabilities for an organization to monitor and keep
track of projects and staff members by the management staff. These features would not
be available in a more generic resource manager application.

Another application with the same domain as Maple is a open-source project
software for the collaboration of teams online called 2-plan (http://2-plan.com/). 2-plan is
web-based just like Maple and offers tools to help with project planning, project
execution and project control. Other features include:

- Project management
- Task management
- Calendar of events
- Team communication (message board)

The following is a snapshot of the “Dashboard” in 2-plan (includes features described
above):
2-plan is an alternative to Maple, however the application was developed using different technologies than Maple. Furthermore, 2-plan was not tested and evaluated closely. Hence there can be features in future builds of Maple that are currently not available in 2-plan.

Mule Enterprise Service Bus (ESB) from Mulesoft.org was also considered as a potential alternative development framework to GWT/GXT. MuleESB is an architecture for application integration and consists of a set of rules and principles for performing the integration over a bus-like infrastructure. This concept helps decouple applications from one another and instead have the applications talk to the service bus. For example, this allows the developer to implement any application using any language and have the application talk to any service to transform, transmit and manipulate data. The company ebay uses MuleESB as their single service integration solution. MuleESB was
considered for this project, but was rejected due to its complexity and size. It would take much longer to develop and deploy an application using MuleESB versus technologies like GWT/GXT.

**Literature**

A substantial amount of work has been performed by Brandon Donnelson, a web-developer and self-proclaimed GWT guru, using GWT to implement various applications. There are numerous examples, demos and snippets of code using GWT located on his website ([http://code.google.com/p/gwt-examples/](http://code.google.com/p/gwt-examples/)). The main aspect to note here is that Mr. Donnelson only used the GWT SDK in his examples and not the GXT layer. Also, some of his code examples are slightly outdated. In other words, some of the functions used in his source code have been deprecated, making some of the demos no longer functional. The main examples that I looked at on his web page are Demo File Uploader and MySql Feedback Application Demo. Both of these examples were used to develop some of the database functionality and file manipulation in Maple. The main difference is that Maple uses the Google DataStore to store data files. Also, instead of using MySQL database, Google Cloud SQL was used. These design aspects of Maple will be discussed in greater detail below.

As mentioned above, Maple uses Google Cloud SQL database to store application data. The steps required to integrate MySQL into a GWT application are not documented by Google, but have been developed separately. I discovered a guide compiled by Bastian Tenbergen during my research which demonstrated the process of installing GWT and step-by-step guidance on integrating MySQL database. The guide was a useful starting point, but it required additional research and development because it was outdated. Some of the steps were even unnecessary and misleading.
Now let us take a further look at the some of the key steps found in Tenbergen’s guide, which are required to perform the integration in more detail. An explanation of the individual items follows.

- Installation of CypalStudio
- Modify Web.xml for Servlet
- Create WAR File
- Install MySQL into Tomcat
- Install WAR file into Tomcat
- Launch Tomcat

CypalStudio is a free application that used to be a required part of the development environment for developers using GWT and Eclipse. Many of the CypalStudio features are now available in GWT directly, so current versions of GWT and Eclipse no longer require it. I originally installed it when following the existing guidelines but later removed it after realizing that it was no longer required. Later on, during the tools and installation section, you will notice that no modification of files like Web.xml (defines mappings between URL paths and the servlets that handle requests with those paths) and WAR file (essentially just like a JAR file with a special folder structure inside) were necessary to the extent that was described in the guide. Both the WAR and Web.xml are generated by Eclipse IDE and very little modification is required to get started with your application. Also, no installation of Tomcat (Web Server) and MySQL database were required for Maple. Instead, Maple uses Google App Engine as its web server and Google Cloud SQL as its database. More details will be provided in the later chapters.

There were additional app development sources that were studied during the development of Maple to gain experience in how GWT was used. One site was Google’s own Google Web Toolkit Showcase of Features webpage
The webpage page describes some of the main Widgets that are offered by GWT. A demo of each Widget, including sample source code, is also provided.

Similar to GWT Showcase, another important web site used to develop Maple is the GXT showcase site (http://www.sencha.com/examples/). This site demonstrates the main Widgets that are provided by the GXT Framework. Again, just like the Google’s showcase site, the GXT site provides a demo of each Widget with sample source code. What really interested me was the quality and look-and-feel of the GXT Widgets. The GXT Widgets include feature-rich optimized templates and layouts. The GXT has additional libraries to choose from, which include fully themed, customizable Widgets.

Maple has been developed with a combination of original code development, plus the inclusion of key code snippets and Widgets from sites mentioned above. For example, some of the database query statements were taken directly from Mr. Donnelson's site, and modified for use in Maple. The guide written by Bastian Tenbergen described the MySQL functionality for a typical GWT project, and it helped me develop equivalent database functionality for Google Cloud SQL that is used by Maple. If needed, Maple can be quickly modified to use a MySQL database. GWT is used for the main application layer of Maple. There are GXT Widgets that were found on the showcase site and used as is. There are other GXT Widgets which were modified in greater detail and used in Maple. In the next section, the requirements and design of the Maple application are described.
REQUIREMENTS

Features

This section will cover the features that will be included in Maple. Also, a list of features that will be implemented in the next build for Maple is provided. These requirements were developed based on my experience with the staff planning spreadsheets from my employer described earlier. I also added new requirements that I concluded were missing from the existing application.

Create Staff Plan

User shall be able to create a staff plan for a program by selecting a start date, end date, program, manager and a group of engineers. Then, the user will have the opportunity to input and allocate hours for the group for that project. The staff plan is then stored in a database.

Delete Staff Plan

User shall be able to delete previously created staff plans.

Edit Staff Plan

User shall be able to modify previously created staff plans.

Department Allocation Report

User shall be able to run at least one report, the Department Allocation Report.

Import

User shall be able to import data such as a staff list from an external source into the Maple database.

Export Maple data

User shall have the opportunity to export data out of the Maple database and into another source.
Future Enhancements

The following features have not been implemented and are considered to be future enhancements to Maple.

Mobile Alerts

An email notification is sent to the program team once a staff plan has been created, edited or deleted. The alerts can be expanded depending on customer needs.

Artifact List

Maple shall provide the user interface to allow artifacts such as SRS (Software Requirements Specification), SDD (Software Design Document), Source Code to be tied with a program’s staff plan. This allows the user to view a desired artifact for a specific program, while viewing the staff plan for that project.

Additional Reports

Customer shall be able to request additional reports to be added to Maple.

Source of Import/Export

User shall be able to have the option to import/export from multiple sources. Currently, Maple is able to export and import from Google’s Cloud Storage.

Login

Maple shall include a login page. Users can login using their current email providers login information.

Access/Permissions

Maple shall provide an interface where permission can be set for users of the application to allow filtration of content. For example, managers will be able to see certain contents that a regular employee will have no access (or limited access) to.

Action Items Database
Maple shall provide an interface where users can input action items for a group, team, project or other users in the department. Action items can also trigger an email notification to be sent out to the appropriate individuals. Furthermore, a mini toolbar menu item can be added to display open, closed, pending action items and have alerts popup and notify the user.
DESIGN

At Northrop Grumman, the Excel spreadsheet version of the staff plan tool used sheets to store application data (i.e. employees, programs, staff plan, etc.). Certain sheets which included information like list of employees were locked in order to prevent users from accidentally removing them. Each time a user created a staff plan, a new sheet would be created and the staff plan data would be populated onto the newly created sheet. Likewise, each time a user ran a report, a new sheet would be created and populated with the report data. The tool also included a GUI which was developed using VBA macros. Inspired by this basic design, I created an improved design, replacing the spreadsheets with a relational database back end and a GWT/GXT based front end.

The following is a high level context diagram of Maple which shows the relationship between Maple and other external services:
Based on that high-level view, we will now discuss the design of the major components in more detail.

**Database**

The main functionality of the Maple application is driven by data stored in the database (Google Cloud SQL). The Excel spreadsheet version of the staff planning tool used multiple sheets to store application data. In order to prevent them from being erased, the sheets were protected. The application data in the spreadsheets were organized into four main sheets: allocations, employees, programs, and staff_plans. For Maple, I used the same concept and reproduced the sheets by creating database tables which represents the same data.
The design of the database is described below in greater detail, including the database schema. Since the Maple application itself does not include any feature for browsing the database directly, we will show how to use the "Google APIs Console" to access and manipulate the database content within the Google Cloud SQL server.

The API console allows the user to configure and setup services provided by Google (i.e. Google Cloud SQL and Google Cloud Storage). The console provides a Dashboard which includes important project information and services status. The following screen capture shows the front page of Google API Console:

If you notice on the left hand side of the image you will find “Google Cloud SQL” and “Google Cloud Storage”, which are services used by Maple. On the right hand side of the image, under Services, you can see the status of the services. By clicking on the Google Cloud SQL, we will end up on the Dashboard page of that service. The Dashboard for Google Cloud SQL looks like this:
The dashboard provides useful information to the user but the most important link here would be the “SQL Prompt”. The SQL prompt will allow you to connect to any database and execute SQL statements. After executing the SQL statement “show tables”, the following tables are displayed:
The main tables that were used for the Maple database were allocations, employees, programs and staff plans. The decision to use the following tables was influenced by the original staff plan tool which was developed using Excel spreadsheets and VBA. As mentioned before, the main sheets in the spreadsheet contained data used by the macros, which were locked in order to protect them from being deleted. The sheets contained data just like a table would in a database. The fields in the following tables are similar to the fields which were in the sheets with some additions. The rest of the tables which are listed above and not described in the following section are there for future enhancement of the product. The following 4 tables are the main tables in Google Cloud SQL database used for Maple:

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>action_items</td>
</tr>
<tr>
<td>allocations</td>
</tr>
<tr>
<td>artifacts</td>
</tr>
<tr>
<td>employees</td>
</tr>
<tr>
<td>programs</td>
</tr>
<tr>
<td>staff_plans</td>
</tr>
</tbody>
</table>
### A. allocations

<table>
<thead>
<tr>
<th>COLUMN_NAME</th>
<th>COLUMN_TYPE</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>staff_name</td>
<td>varchar(20)</td>
<td>Name of employee (see table)</td>
</tr>
<tr>
<td>staff_plan</td>
<td>varchar(20)</td>
<td></td>
</tr>
<tr>
<td>m1</td>
<td>varchar(6)</td>
<td>Man-months for that month</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>m12</td>
<td>varchar(6)</td>
<td></td>
</tr>
</tbody>
</table>

In the allocations table the m in “m1” or “m12” stands for month. There are actually 12 months (m1 - m12) and the list has been truncated to save room. The following image is the schema of the database used in Maple:
The issue with the Maple database is that it inherits some of the data design deficiencies of the original spreadsheets, such as a lack of normalization. The fields and tables of the database have not been organized to reduce redundancy and dependency. If I had the opportunity, I would redesign the database and refine the relationships between key fields in the tables. For example, the empID would be the primary key in employees, and a foreign key in the allocations table.
In the section above we covered some of the high level requirements and database design for Maple. In the following sections, the software design of the Maple application is described including UML diagrams.

**Software Design**

In this section we describe the software design for the core functions provided by the Maple application to manage staff plans. Some of the functions included in Maple are:

- Maintaining a connection to the database
- Creating, Editing, Deleting staff plans
- Adding/Editing/Removing employee from the database
- Adding/Editing/Removing programs from the database
- Importing/Export data from Maple
- Generating a staff allocation report

**UML**

The following sections will provide some UML diagrams which will help demonstrate some components of Maple.

**Class Diagrams**

- class MapleDatabaseManager - The main purpose of this class is to maintain a connection with the Maple database and handle all user queries.
MapleGUI - This class is the driver class for building and launching Maple. This class constructs the Maple GUI (windows, panels, menu bar, toolbar, listeners, logging etc...), where users can interact with the different features the application offers. The following figure illustrates how the MapleGUI class sits at the center of the activity and interacts with the other classes:

- MapleEmployee
- StaffPlan (serializable class used to store staff plans which are queried from the database)
- MapleStaffPlan (main function is construct a staff plan based on user input).

The functionality of the primary classes used in Maple will be described in the Implementation section.
Use Case Diagrams

In this section, the most important use-case scenarios for Maple are described.

Navigation:
The following sequence diagram shows the main sequence of function calls that occur on the launch of the application. Maple starts off with the onModuleLoad()
function located inside class Maple. The app calls the constructor for the MapleGUI class to construct the Maple Application’s GUI, connect to the Maple database, load data (employees, programs, current staff plans), which is required for functionality, and available to the user. The connections and database queries are asynchronous calls to the MapleDatabaseManager. In other words, the GUI can be constructed by the MapleGUI class and returned back to Maple even if there are database issues such as bad connection or bad query.

Top level classes like MapleGUI and MapleDatabaseManager were described using UML diagrams. In the next section, the implementation of classes used in Maple will be described in detail.
IMPLEMENTATION

Implemented

The following table will provide details on the main classes which were implemented for the Maple application. Classes that were taken from the showcase sites and modified to integrate with other Maple code are labeled “Customized”. Classes that are automatically generated by the IDE and later customized for integration with other code are labeled “Auto+Customized”. Code that is new development specifically for Maple are labeled “New.”

<table>
<thead>
<tr>
<th>Java Class</th>
<th>Implementation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeEditingGrid</td>
<td>Customized</td>
<td>Calls the createGridEditing() function for the EmployeeGrid class. Taken from an example given on the Sencha’s tutorial page.</td>
</tr>
<tr>
<td>GreetingService</td>
<td>Auto + Customized</td>
<td>Class created by IDE. Name of class was reused and additional functions were added to the class. The function in this class are all the synchronous calls made to the server.</td>
</tr>
<tr>
<td>GreetingServiceAsync</td>
<td>Auto + Customized</td>
<td>Same as GreetingService, but these are all the asynchronous callbacks from the server.</td>
</tr>
<tr>
<td>Images</td>
<td>Customized</td>
<td>An interface to the images used for the Maple. Mainly used by the MapleGUI class. This class was taken from an example given on Sencha’s webpage and customized for Maple.</td>
</tr>
<tr>
<td>Maple</td>
<td>New</td>
<td>Initially generated by the IDE once the project was create, however it had a complete overhaul.</td>
</tr>
<tr>
<td>MapleChart</td>
<td>New</td>
<td>Creates the chart for the</td>
</tr>
<tr>
<td>Class</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MapleChartData</td>
<td>New</td>
<td>Data that goes in the chart for the Allocation Report.</td>
</tr>
<tr>
<td>MapleDetailsGridModel</td>
<td>New</td>
<td>The Model class for the Details section of the Maple GUI (located on the right hand side of the application)</td>
</tr>
<tr>
<td>MapleEmployee</td>
<td>New</td>
<td>The model class for the employee grid.</td>
</tr>
<tr>
<td>MapleEmployeeGrid</td>
<td>New</td>
<td>Creates the grid for the user to be able to update employee information.</td>
</tr>
<tr>
<td>MapleExport</td>
<td>Customized</td>
<td>Exports data out of Maple into Google’s Cloud Storage. The source code was directly off of the Google’s Cloud Storage tutorial page. Minimal changes to the code.</td>
</tr>
<tr>
<td>MapleGUI</td>
<td>New</td>
<td>Constructs the GUI of Maple and connects to the Maple DB. This class provides all functionalities of Maple available to the user.</td>
</tr>
<tr>
<td>MapleImport</td>
<td>New</td>
<td>Imports data from Google’s Cloud Storage into Maple.</td>
</tr>
<tr>
<td>MaplePrograms</td>
<td>New</td>
<td>The model class for the programs grid.</td>
</tr>
<tr>
<td>MapleProgramsGrid</td>
<td>New</td>
<td>Creates the grid for the user to be able to update program information.</td>
</tr>
<tr>
<td>MapleStaffPlan</td>
<td>New</td>
<td>Creates, deletes and lets the user modify existing staff plans.</td>
</tr>
<tr>
<td>MapleStaffPlanGrid</td>
<td>New</td>
<td>Creates the grid for a staff plan.</td>
</tr>
<tr>
<td>ProgramsEditingGrid</td>
<td>Customized</td>
<td>Calls the createGridEditing() function for the EmployeeGrid class. Taken from an example given on the Sencha’s tutorial</td>
</tr>
<tr>
<td>Class</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RowEditingGrid</td>
<td>Customized</td>
<td>Calls the createGridEditing() function for the Staff class. Taken from an example given on the Sencha’s tutorial page.</td>
</tr>
<tr>
<td>Staff</td>
<td>New</td>
<td>The model class for the staff plan grid.</td>
</tr>
<tr>
<td>StaffPlan</td>
<td>New</td>
<td>The serializable class to extract the stored staff plans that are located in the database.</td>
</tr>
</tbody>
</table>

The section above described some of the main classes that were used in Maple.

The following section will help demonstrate the setup and configuration of the tools used in the development environment and a demonstration of common usage scenarios of Maple.
TOOLS

Majority of the tools and libraries are free, but the correct installation and configuration of the development tools and development platform is not trivial. Adequate time must be allocated to properly configure and setup the environment. There are versioning issues (cross-compatibility between different versions of different tools), and the Eclipse IDE is the primary tool used in the development environment which must be carefully customized with the correct libraries and plugins. Previous developers have written guides to help with the installation process which are now out-of-date. My goal in this section is to give enough details to help the next developer understand the details necessary to setup and configure the development environment (even when my guide becomes out-of-date).

Installation

The following section will describe the applications, accounts, libraries, extensions and necessary steps required to build a web-application such as Maple. It is important that you follow the directions on Google’s website carefully as the installation process may be complex.

The following list is a summary of all the tools and plugins that will be required to download and install:

- Java SDK 1.6
- Eclipse IDE Enterprise Edition (Juno)
- GWT SDK
- Sencha GXT Library
- Google Cloud SQL Account
- Google Cloud Storage Account
- Google Chrome
- GWT Plugin for Google Chrome

This chapter goes into each step in greater detail.
First and foremost, you will need Java SDK version 1.6 or later. Visiting Oracle’s web page at http://www.oracle.com/technetwork/java/index.html or simply Googling “Java SDK 1.6 Download” will help you find the correct files needed to install. Choose the right platform and following the prompts to complete the installation. The file used for this project was found here, http://www.oracle.com/technetwork/java/javaee/downloads/java-ee-sdk-6u3-jdk-6u29-downloads-523388.html, with the filename “java_ee_sdk-6u4-jdk-windows-x64”. The reason behind the exact link and filename is that there are so many versions and platforms to choose from. One may end up downloading the incorrect installer.

Next, download and install the GWT SDK. If you are using Eclipse (which I would highly recommend for GWT), download the GWT plugin for Eclipse straight from Google’s website. The download includes IDE support for the Google Web Toolkit and App Engine web projects, lightweight version of GWT Designer (which will not be used for this project, but it is available for use) and additional Eclipse tools.

Download and install Eclipse IDE (JUNO version). You must be careful to choose this version or the versions recommended on Google’s GWT web page. I provided the exact link to access the Eclipse version that worked for me:


Any other version of Eclipse may be risky and can lead to failure. Currently, GWT is only compatible with certain versions of Eclipse. Note that in the previous step, the installation of GWT SDK was described. There is an alternative method to install the GWT SDK and App Engine. You can use Eclipse installer as well. The directions on how to use the Eclipse installer is provided by Google here:
The GXT library can be found here:

http://www.sencha.com/products/gxt/download/

The file name which was used for this project is “gxt-3.0.1-GPL”. The zip file contains two main jar files that will be used for the GXT Widgets in the project. The main GXT library jar file and the chart jar file both must be added to the classpath of your project. This step will be covered later on in this section.

Next, you will need to create a Google Cloud SQL account. The account is not free. However, Google is offering a six month trial period which ends June 6th of 2013. An alternative, open source database like MySQL can be used as well, but additional tools like WAMP will be necessary to get MySQL up and running. Google Cloud SQL is more standalone self-contained solution. Follow the guide provided by Google here:

https://developers.google.com/cloud-sql/docs/before_you_begin

Another account required would be a Google Cloud Storage account. Follow the configuration and setup process provided on the following web page:

https://cloud.google.com/

Remember, at one point that Google will require payment information to be stored away in case you decide to pay for the service. However, you will not be charged until your trial period ends.

Finally, if you have not already done so, download and install the latest version of Google Chrome. It is not necessary to use Google Chrome, other browsers will work as well. I believe that I would avoid possible future compatibility issues by using Google’s browser, although I don’t know of any problems with other browsers. If you decide to use Google Chrome, you must install GWT Developer Plugin for Google Chrome. A
simple Google search will take you to the necessary link that will download the plugin. Without this plugin, your application will not load in developer mode.

Now that the tools have been downloaded, installed, and configured, the development platform is complete. We can now start the first development project using the platform.

Create First Project

The following section will provide a quick guide on how to create a new project in Eclipse. This is so we can make sure the necessary tools and packages were downloaded and installed properly on your system. If any of the steps here on fail or crash, this is an indicator that a step in the previous “Installation” sections was not followed properly. It is important to note that the following Eclipse IDE will have GWT plugin installed, which will have a different look-and-feel from a default Eclipse installation.

In order to get started we will need to load Eclipse. Once Eclipse has been loaded, click on File -> New -> Web Application Project:

The “Project name” and “Package” is necessary to continue. Make sure “Use Google Web Toolkit”, “Use Google App Engine” and “Generate project sample code” are checked (usually are by default) before clicking Finish.
In this example, the project was named “masters” and the package was named “thesis.” The reason we checked “Generate project sample code” is that we are going to let the IDE do some work for us for free. By that I mean the following core/important files are generated by the IDE within the “masters” folder (the folder for the project):
The following table will describe the main files in the project:

<table>
<thead>
<tr>
<th>File</th>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GreetingService.java</td>
<td>thesis.client</td>
<td>synchronous callback (specify the methods in this class that will provide DB queries for you)</td>
</tr>
<tr>
<td>GreetingServiceAsync.java</td>
<td>thesis.client</td>
<td>asynchronous callback from the server side (specify the methods in this class that will provide DB queries for you)</td>
</tr>
<tr>
<td>Masters.java</td>
<td>thesis.client</td>
<td>Your “main” file. This class implements the “EntryPoint” class. This</td>
</tr>
</tbody>
</table>
class includes the “onModuleLoad()” function, which is the entry point of your application (just like “main()” in C/C++ or “public static void main()” in Java.

<table>
<thead>
<tr>
<th>Class</th>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GreetingServiceImpl</td>
<td>thesis.server</td>
<td>DB queries happen here. This class is responsible to keep a connection with your DB and to perform queries requested from the client side by the user.</td>
</tr>
<tr>
<td>web.xml</td>
<td>war</td>
<td>All servlets (which provide dynamic content such as the results of a database query) information go in here.</td>
</tr>
<tr>
<td>Masters.html</td>
<td>war</td>
<td>This file contains a link to a JavaScript file which loads your compile application (module).</td>
</tr>
</tbody>
</table>

Before we launch this application, let us add two jar files to our class path. This is so the project is able to recognize GXT Widgets and Charts. In order to add a jar file to the project’s class path, we will need to do the following:

Right Click on your projects top level (root) folder (in this case called “masters”) and go to “Properties” (located at the bottom of the menu). Click on “Java Build Path” on the left hand side and click “Add External JARs...” on the right side. Locate and add both the GXT main library (in this case it was “gxt-3.0.1-GPL.jar”) and the Chart library. It should look something like this:
Click “OK” to exit the properties dialog box.

Now, locate <PROJECT>.gwt.xml file:

```
<inherits name="com.sencha.gxt.ui.GXT"/>

<inherits name='com.sencha.gxt.chart.Chart'/>
```

Now, most likely your project “root” folder has a small exclamation point on it like so:

```
- masters
  - src
```

This is due to the fact that the jar files are not going to be visible on the client side. There are two solutions to this. One would be to directly copy the jar file(s) to the root project folder. The second solution would be to take advantage of the IDE environment and allow Eclipse to fix the problem for you. If you open the “Marker” tab on the center-bottom of the application, you will see the issue and the IDE will provide a “Quick Fix” which you can take advantage of. The “Quick Fix” in this case will copy the missing jar files for you.
There is one more critical step to follow. The “JDK Compliance” level must be changed to 1.6 (in this case). Again, we must visit the “Properties” of the project and select “Java Compiler” from the left hand side of the properties dialog box and 1.6 from the drop down menu of the compliance level on the right hand side of the dialog box. Now, the project root folder will have a red “x” marked on it. Again, in order to fix this issue we can visit the “Markers” tab and select “Quick Fix” and have the IDE fix the problem for us. The error that you will most likely run into will look similar to this:
We will need to link Google Cloud SQL to the project properties as well. In order to do this, we will once again need to load the properties of the project by left clicking on the root folder and clicking on properties. Then, on the left hand side of the properties window, select Google -> App Engine like so.
Then you must select the “Enable Google Cloud SQL” check box and the “Use Google Cloud SQL instance” radio button. Then we must configure the database by typing the instance, username, and password. This information will be given to you once you setup the Google Cloud SQL account by following the link given above. Click on the last two Configure links on the bottom right hand side of the above image and input the same information in both boxes like so:
Once you click OK on the project’s preferences window, Eclipse will attempt to connect and store the information for your project. Then your project should be ready to interact with Google Cloud SQL.

Now we are ready to launch our application in the browser in developer’s mode. In order to launch the application, left click once on the project root folder (in this case “masters”). Now, from the menu bar, left click on the down arrow next to the green circular play button and select “Run As” and then “Web Application” like so:

![Image](image1.png)

The following link should get generated on the bottom of the screen. If it is the first time running this link, make sure we have the correct browser selected (or else we could have just double clicked on it). In order to check the default browser, right click on it and select “Open With” and make sure Chrome is selected:

![Image](image2.png)

After launching the link, the application should load in the browser (loading time varies based on system speed). If loaded successfully the test application should look like this:
Web Application Starter Project

The application can also be deployed into Google’s App Engine where it becomes public and anyone can have access to the application. If this is the first time the application is being deployed, you will need to fill in Application ID in the properties of your application like so:

After the properties have been updated, you are ready to deploy by clicking on the “spaceship” icon in the Google menu bar in Eclipse, “Deploy to App Engine...”.
The project should be accessible on the internet using the following link:

<your_project_app_id>.appspot.com

After you point your browser to the app’s URL, the application launches in your browser in production mode rather than developer mode. You will notice the application will not only load faster, but will also be responsive compared to running in development mode using your IDE on your computer. According to Google, when the application is in “Development mode”, it uses a special engine to run the application, because the application includes a mixture of both Java bytecode and native JavaScript. This is especially true when the application makes many calls back and forth between Java and JavaScript. Furthermore, UI code can slow down performance as well. This is not the case when the application is deployed into Google App Engine and running in “Production mode.” Application performance and behavior improve once deployed into the Google App Engine.

Now that the installation of the tools and the creation of the project has been described, we are ready to present a walkthrough of the complete application running in production mode.

**Demonstration**
In this section, we present some walkthroughs of common usage scenarios of the finished application Maple.

Create Staff Plan

The following section will describe steps to take in order to create a new “Staff Plan” in Maple.

A. go to “sasadorian.appspot.com”
B. You will view the following, once the application is launched in the browser.

C. Click on “Create” on the left hand side under “Navigation” section, “Home” heading, “Staff Plan” folder
D. Select “Start” and “End” date and click “Next”

![Create Staff Plan](Image)

E. Select “Manager” and “Program” and click “Next”

![Create Staff Plan](Image)

F. Select a team of engineers and managers and click “Next”

![Create Staff Plan](Image)
G. You will notice a newly created tab and staff plan in the center of the app.

Allocate

hours for each staff member in the staff plan and click “Submit”. The staff plan is then stored away in the Maple DB.

**Update Employees**

The following section will describe steps to take in order to add and update employees in the Maple application:

A. Click on “Update Employees” on the left hand side of the Maple application under “Staff”:
B. You can “Add Employee” or click on an existing Employee and modify fields as needed. Once you have completed your entry, click on “Update” located on the bottom of the “Update Employee” tab.

**Update Programs**

A. Click on “Update Programs” on the left hand side of the Maple application under “Programs”:
B. You can “Add Program” or click on an existing program to rename or remove the program. Once you have completed your entry, click on “Update” located on the bottom of the “Update Programs” tab.

This concludes the setup and configuration of the development environment, and demonstrations of common usage scenarios of Maple. The following sections will discuss success criteria for this project and lessons learned including a list of software bugs which were discovered during the development phase of the project.
CRITERIA FOR SUCCESS

The goal of the project was to take an existing practical Excel-Spreadsheet-based staff management application in use at my current employer and re-implement it using modern technologies such as Java, GWT and GXT. Had there been enough time, I was hoping to define an experiment where my employer would have used the reimplemented app in parallel to their current app in order to give me feedback. Unfortunately this part of the plan was not implemented, but it may still be possible to revisit this in the future.

A separate goal was to use newer GUI technologies such as GWT and GXT to expedite the development process from the use of the themed and customizable widgets. GWT is an interesting emerging technology that promises to simplify the process of developing truly cross-platform web-based front ends. Sencha GXT helps us by providing the rich, advanced, and powerful layer which takes GWT to the next level. Finally, I had the goal to implement a final version of the Maple application that was able to perform most of the features described in the requirements section of this document.
LESSONS LEARNED

In this section, I will share some lessons learned based on my experience of performing the implementation of the application, including detail on the setting up of the development platform.

As with any complex implementation project, I recommend documenting the process with a project diary that can capture all tasks or issues that come up on a daily basis. It often happens that you cannot realize the significance of a small problem until later in the process. I started documenting my progress daily and at one point stopped. Documentation can help during the implementation phase and when writing your project thesis. A project diary helps you review the final write-up for completeness.

Another lesson learned during this project is that “showstoppers” can really affect the project schedule. In this instance, showstoppers means IDE issues, or issues implementing some of the complex GUI components. One recommendation would be to start early on the project. Another recommendation would be to search the issues online and try to find a resolution for them as early as possible. Posting to public forums is a great way to reach out to the community for help on pretty much every topic you can think of.

Although the research contribution of the work was to develop a staff management application and to show how to use technologies like GWT/GXT, it’s a fact of life that no useful research can be conducted until the development platform is working. The time and attention to detail required to install and configure the tools turned out to be a larger resource sink than anticipated. Existing but out-of-date guides and tutorials were extremely important starting points, but gaps or mistakes in these guides occasionally brought development to a halt. As a result of the experience with this
project, I believe I could publish an updated version of some of the guides that I used to simplify the process for any other researchers who want to develop similar projects.

Lastly, spend as much as time possible on the requirements of your project. Software Engineers know how important requirements can be and how detrimental it can be if the requirements change during the implementation phase of the project. It is costly and it can drastically impact the schedule and in real world cases, the project budget. The requirements and design for this project were modified a few times and did impact the schedule for this project. One requirement for Maple was to Import and Export data from and to the application. However, during development phase, I discovered that GWT has no support for accessing file system on the client side. The research, design, and implementation of a new method impacted the project schedule. The solution to import and export data from Maple was to use Google Cloud Storage. As you can imagine, it can be difficult and time consuming to setup and integrate a new service with an existing project.

The following section will cover some of the errors I encountered during the development of Maple which also had an impact on the project schedule.

**Development Error Troubleshooting**

The following table includes some of the most significant error messages that appeared during development, with a description of the error and the fixes that were discovered while developing Maple:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
<th>Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unable to build your project. When trying to build, nothing happens. The IDE just sits there on idle.</td>
<td>The project was developed using my laptop. There were times when I would have no internet connection. After loading</td>
<td>Connect to the Internet.</td>
</tr>
<tr>
<td>Issue</td>
<td>Solution</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Unable to see the latest changes after updating the server side code.</td>
<td>As mentioned in a section above, you can update your source code and hit the refresh button on your browser to see the latest changes. This is without stopping, building and restarting the entire process every time there is a code change. However, this does not work when the code on the server side gets updated.</td>
<td>Must stop the currently running service and restart/rebuild the project.</td>
</tr>
<tr>
<td>Application loses connection to the IDE within the browser.</td>
<td>The IDE and your loaded application in the browser have a constant connection and communication. This connection can go stale.</td>
<td>Must shut down and kill the Eclipse and the Browser process. Another solution would be to temporarily load your application using a different browser.</td>
</tr>
</tbody>
</table>
CONCLUSION

Fortunately, Maple was developed successfully using Java software development augmented by the technologies GWT and GXT, as well as links to or dependencies on Google Cloud SQL and Google Cloud Storage. The steps and details given in this document were gathered from multiple sources online, some of which required significant updating and correcting. I am confident that the next individual who is interested in building a web-application using these technologies will benefit greatly by using my updated guides as a starting point. The table of bugs alone should save some research time and help reduce impact on the project schedule.

If additional time were given, the future feature list (covered in the Requirements section earlier) would have been implemented as well. However, due to the multiple showstoppers, the project schedule was impacted to the point where these features had to be left for the next release of this product.

After completing this project, I am more familiar with the GWT and GXT libraries and their usage. I would have implemented certain things differently if I knew some of the concepts in advance. If the Maple application were to be tested and accepted as a production-level application, I would redesign some of the key functionalities to produce a more modularized and component based architecture, as well as greatly increase the number and scope of the features.

For the future Software Engineers who would be interested in using the technologies from this project, I would suggest starting early and having your requirements baselined. Change in requirements can be costly and may affect the project schedule.
I will attempt to commercialize this product sometime in the future. I will use social networks and public forums to publish some of the features of the product and get feedback from the community. Another idea I had was to pitch the product to Northrop Grumman and see if they would like to continue using the spreadsheet version of the staff plan tool in parallel with Maple as a controlled evaluation experiment. This way they can experience a similar tool with improved performance and user interface and provide me with valuable input from how the application performed in a real deployment. Hopefully, Northrop Grumman will abandon the old Excel spreadsheet tool and move in the direction of more modern applications, if not Maple, then something similar.

I started the project not knowing which technologies I was going to adopt for my application. Both GWT and GXT were a success for this project and helped me develop a rich enterprise class application. The features that I wanted Maple to offer were implemented successfully. Due to time constraints, some additional features were omitted from this version of Maple. Majority of the time was spent researching the tools and configuring the development platform. The project schedule was also impacted during redesign efforts. As mentioned before GWT had no support for client side storage access. Thus, additional research effort took place during the development phase of the project. The research led to Google Cloud Storage, which is a method to import and export data from and to Maple.

If I had the opportunity to develop yet another web-application, I would revisit GWT and GXT mainly due to my success and experience with both technologies. Now that I am familiar with setup and configuration of the development platform, I can concentrate mainly on the application requirements and design rather researching
solutions for errors and crashes. I believe this paper has enough information to lead the next developer in the right direction for their next big project.
REFERENCES

https://developers.google.com/web-toolkit/overview

http://www.sencha.com/products/gxt

http://code.google.com/p/gwt-examples/


http://www.w3schools.com/sql/

https://developers.google.com/appengine/docs/java/cloud-sql/developers-guide

https://developers.google.com/web-toolkit/doc/latest/tutorial/appengine


https://developers.google.com/web-toolkit/examples/


http://gwt.googleusercontent.com/samples/Showcase/Showcase.html#!CwCheckBox