Project Concept

*League of Legends*, by Riot Games, is the world’s most popular free-to-play game. In a 5-versus-5 match, each player takes control of a certain “Champion”. Champions normally need to be purchased using real money or through an earned in-game currency. However, Riot also offers a rotation of Champions that are entirely free that week.

Using Riot Games’ API, for which I have registered an account, I have a server periodically query for the current free to play champions, store the data in a MySQL database, and use this data generate a dynamic web page for the user when they visit my website.

The website is available at <http://www.stevesite.me>. The source is now publicly available on [Github](https://github.com/scb5304/FreeWeekChampions) as well.

Glossary

Champions: Characters in League of Legends that the player can control. In a given match, 10 players, divided into 2 teams of 5, each control a Champion and attempt to destroy the center of the enemy’s base. Each Champion is unique, even though there are currently a massive 123 Champions released.

Abilities: Each Champion has 5 abilities. They are unique to the Champion and are used to accomplish various tasks, such as dealing damage or healing.

Lore: Term used to refer to the in-game universe, and in this case, the given champion’s backstory.

Free Champion Rotation: Every week, 10 to 11 champions are provided for free by Riot Games. They typically require purchase through in-game currency or real money. Free champion rotations are *not* guaranteed to happen on a certain date or time ([source](http://forums.na.leagueoflegends.com/board/showthread.php?t=46045)).

Setting up the Project

This project was created using a variety of software, though development was primarily done through Eclipse and the command-line-interface. As I walk through each step of development, I will discuss the technologies that were used to accomplish a given task.

**Setting up Bitbucket**:

Bitbucket was used as version control in this project. I created a repository on Bitbucket and then cloned it on my desktop using the following command:

git clone git@bitbucket.org:scb5304/freeweekchampions.git

After I made changes to my project, I both committed these changes locally to my repository and then pushed them to Bitbucket. This allowed me to occasionally work on the project while at school using Cloud 9 ([c9.io](file:///C:\Users\Steven\Documents\Microsoft%20Office\c9.io)). They generally went as follows:

git add \*

git commit –am “Message regarding my changes”

git push

This was my first time using version control software such as Git, and I found it extraordinarily helpful in looking at older versions for pieces of code I want back, and keeping a history of my code changes just to see how far it has come. I also imagine that source control is crucial in the industry, so I think that acquiring experience with Git could give me a leg up in a job interview and on the job.

**Setting up Eclipse for use with Tomcat**:

Tomcat was used both as a web server and servlet engine in this project. In order to use Tomcat and create Dynamic Web Projects, I installed the Enterprise Edition of Eclipse on top of my current installation. I then added Tomcat to Eclipse by using the Servers tab and selecting “New -> Server -> Apache -> Tomcat v7.0 Server” and then navigating to my local installation.

Although I would eventually use a separate system for actually deploying the software, having Tomcat installed on this development machine was helpful for testing and debugging.

**Creating the MySQL Database:**

MySQL was used as the Database Management Software (DBMS) for this web application. Although I had some prior experience with SQL queries in general from IST 240, I had no experience with the MySQL software. By using some online tutorials and doing some tinkering, I was able to create the database required for my app. I used to the command-line interface just to get some more experience using it.

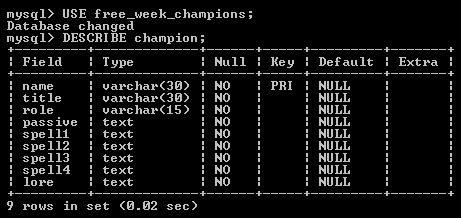


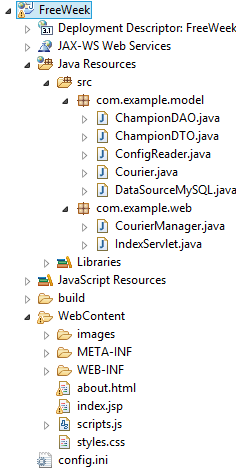
Figure 1: The MySQL database “free\_week\_champions” contains the “champion” table

I then used the “java.sql.DriverManager” with an instantiated JDBC driver to establish a connection to the database in my code.

Developing the Application:

My project went through much iteration as I attempted to perfect my design and also optimize the code. I attempted to utilize the MVC architecture as detailed in *Head First Servlets and JSP* such that the model layer of my code would be completely independent of the web aspect that utilizes it.

I also used the Database Access Object (DAO) design pattern when dealing with my database. As I understand it, a DAO is responsible for actually reading and writing to the database, and a Data Transfer Object (DTO) stores the data which is being transferred. If I had multiple databases, then I would have had my DAO implement an interface that details the necessary operations for interaction with the database. But, as I only use one MySQL database, I simply used a standalone DAO class. Also in order to keep the password for my database and my API key safe, I kept them elsewhere on the disk in a config.ini file which is accessed by the application at runtime.

I created a simple *Courier* runnable that is responsible for querying Riot’s API every given interval of time, in this case every hour. A java library I found on the internet, [Orianna](https://github.com/robrua/Orianna), aids in interacting with Riot’s API. The *Courier* uses the DAO for deleting the old champions and adding the new ones, if it is determined to be a new free week.

In order to ensure that this operation occurs as soon as the server starts automatically, I also created a *CourierManager* class which implements *ServletContextListener*. In this way, I can override the contextInitialized() method and place the instantiation and starting of my *Courier* here, so that it will start when Tomcat launches.

As per *Head First*, I tried to separate the controller and view entirely from the model I was just discussing. Although a visual explanation of how my classes interact and how the components work together in the application are present in the Appendix, I will briefly describe them here for some additional insight.

When the Server is started, the *CourierManager* launches the *Courier* who makes sure that the newest free week champions are stored in the database, and updates them if necessary. Then, the Server continues operation and waits for a user to actually visit the website. When they do, they arrive at the *IndexServlet*, thanks to the web.xml detailing what the welcome file should be. Here, this controller *IndexServlet* retrieves a List of the current free week champions using the DAO, creates a List of cleaned champion names (for use in file names, as one cannot use periods and apostrophes), and sets these two Lists as attributes in the Request. The request is then forwarded using a *RequestDispatch* to the View component of the application.

Figure 2: The Eclipse directory structure for the application.

The view consists of an *index.jsp*, which uses a CSS file and JavaScript file for display and functionality. Although I initially used scriptlets in my JSP for looping through the Champion data and outputting display as HTML, I upgraded the entirety of the JSP to use JSTL, a tag library for use in JSPs. In this way, there is no raw Java code in my JSP, only markup, which keeps it tidy and entirely a view component. I did this after researching scriptlets and reading far too many people insist that it’s an outdated and hard to maintain way of presenting data.

The HTML and CSS were coded entirely by hand without the aid of any additional libraries. Although I’m not a UI designer I’m actually rather happy with the outcome. I put a lot of work into making the page look suitable for the target audience of a gamer, and making it look more sleek than business-like.

JavaScript is used client-side in order to switch between each Champion’s information. The trick I used for sending all the Champion data but only showing one at a time was to send the Champion data in separate divs, but also have a blank div at the beginning. Depending on which image is selected (the first one by default), the blank div’s inner HTML is replaced with the div that corresponds to that image. This makes it possible to only query my server once when retrieving the web page initially, and then swap through the hidden HTML for display.

Deploying the Application:

I used my old laptop instead of my development desktop for deploying the web application. After some head bashing I was able to successfully configure my laptop to be able to boot either into Windows 7 or Ubuntu. I chose to use Ubuntu in order to garner some experience with Linux operating systems. I installed the necessary components using the Terminal and got Tomcat running.

In order to transfer my project from the development environment to the deployment environment, I used software called [PuTTY](http://www.putty.org/). Using some simple commands, I could transfer the application in the form of a Web Application Archive (WAR) file to my laptop:

cd C:\Program Files (x86)\putty  
pscp c:\users\steven\desktop\ROOT.war steven@192.168.1.109:/home/steven

I put this in a simple .bat file due to how often I was likely going to be updating the server.

On my laptop I also created a bash script for replacing the old WAR file with the new one using administrative privileges. By doing this, I can simply run the script update\_root.sh through PuTTY without ever actually using my laptop itself. The script is as follows:

sudo mv ROOT.war /var/lib/tomcat7/webapps  
cd /var/lib/tomcat7/webapps  
sudo rm –rf ROOT  
sudo service tomcat7 restart

Lastly, in order to make my application publicly accessible from the web, I acquired a domain name from <namecheap.com>. They offer one free “.me” address to students with a “.edu” e-mail address, so I didn’t need to spend any money. I configured the “stevesite.me” address I acquired to point to the external IP address of my router. Then, I port forwarded my router to take traffic sent to port 80 and forward it to the internal IP of my laptop’s server.

Addressing Questions

It’s my hope that laying out the information above about my project details how it relates to distributed programming, shows the challenges that I had to overcome, and also all the things I learned working on this project. However, in order to specifically address some things…

**How is it related to distributed computing?**

The entirety of the application doesn’t happen on one machine. It uses client-side JavaScript to parse through web pages that are generated server-side based off of data stored in a database. The database itself is managed using data retrieved from Riot’s remote server.

**Why did you decide to choose this project?**

I’m an avid player of League of Legends. I also was looking for an opportunity to use Linux, MySQL, and Java Enterprise Edition so this was an easy way of having information I find interesting to use for these purposes.

**What are the challenges you faced and how did you work through it?**

I encountered challenges all along the way. This includes creating a MySQL database through the command prompt, creating JSPs, deploying applications to a Linux system, and using the design patterns I read about. I worked through these challenges through countless Google searches, articles, and YouTube videos.

**Describe in detail three specific items you learned while doing the project**

Hopefully here I can just say that all of the detail I provided in the last two sections above shows *at least* three specific things I learned.

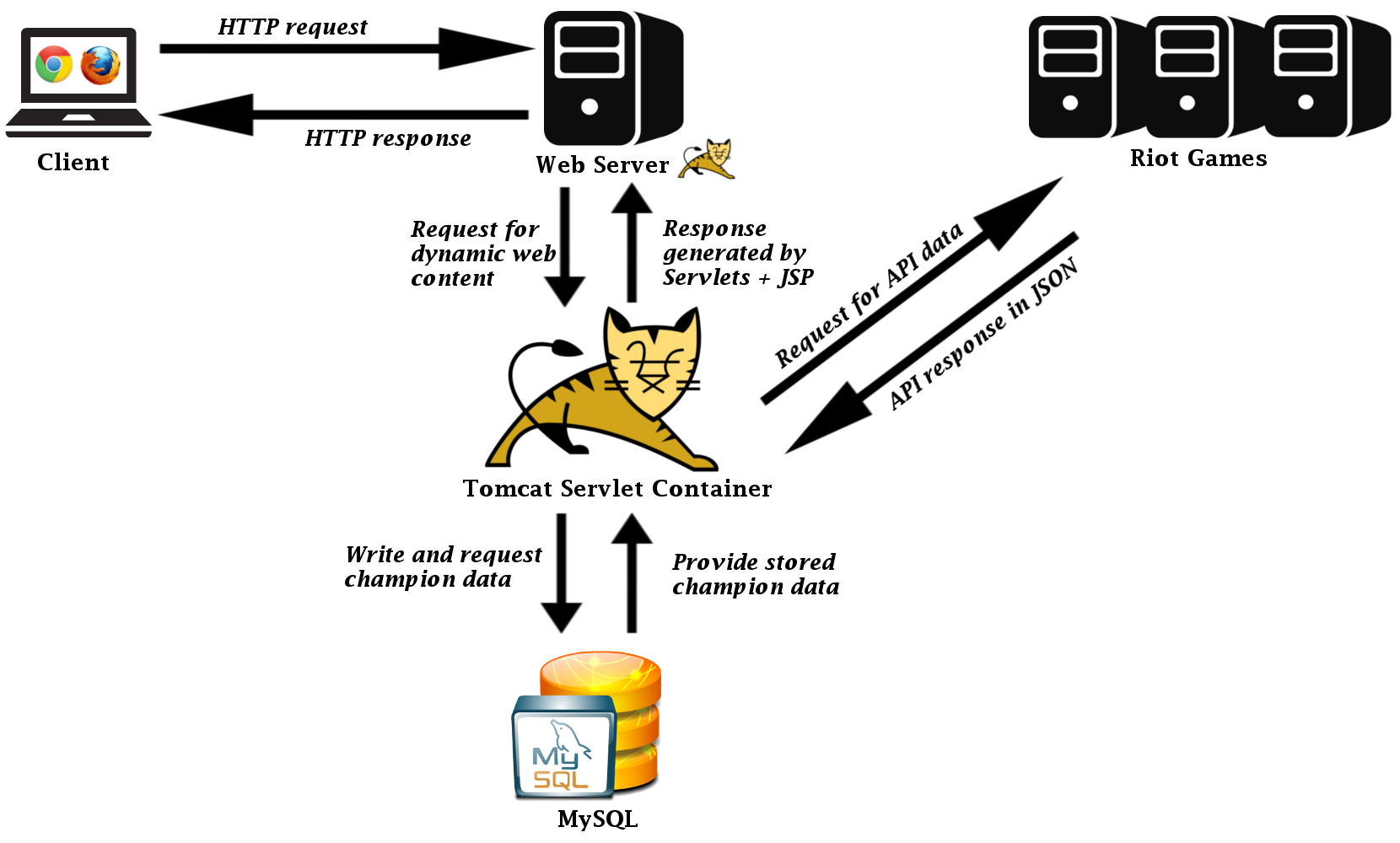
**How would you extend this project?**

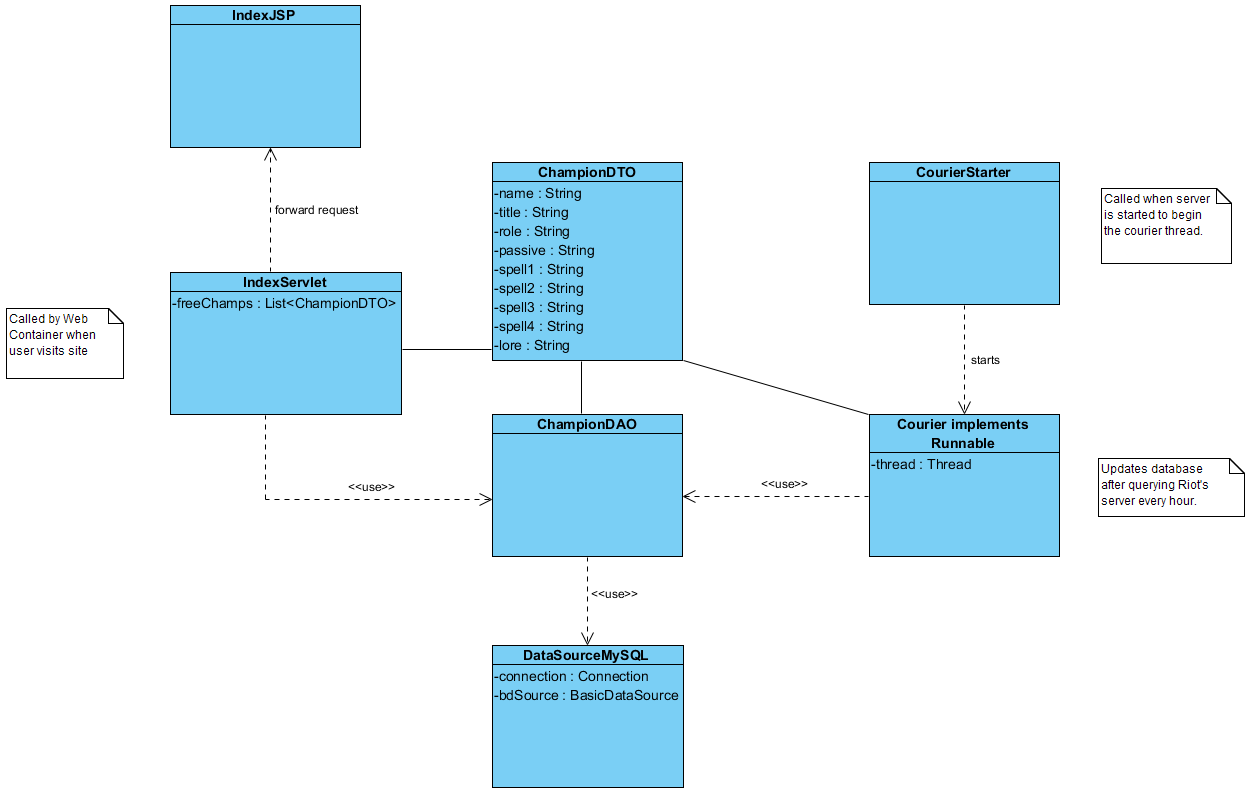
The project could also display data about Champions that aren’t currently free in a separate section. It could also utilize a mobile app, or at least be better suited to mobile devices. It could also link to data found about these Champions that are already stored on other sites.

**How could this help you in getting a job?**

I learned a number of valuable skills regarding SQL and distributed computing. I like to think that my upcoming FedEx internship will be much easier now that I’ve spent so much time familiarizing myself with the web aspects of Java, as that is the language I will be using. Hopefully my experience will both help in the project I’ll be working on and maybe put me ahead of other interns in the quest for a full-time, permanent position.

Appendix A: Components/Architecture



Appendix B: Class Diagram