Lucas Kitaev

Mr. Combs

Senior Projects

21 January 2018

Journal 3

Goals

- Register for Amazon Web Services and set up hosting
- Finish writing job shadow
- Configure webpack and npm
- Research how to use JavaServer Pages and create search functionality
- Research how to use a database with Java Server Pages
- Register for OMDb API to use for queries

Research

The first part of my research focused on learning how to use the main component of Java web applications: JavaServer Pages (JSP). As the name implies, JSP is meant to be run on a server and sent to clients in a dynamically generated form, rather than a regular Hyper Text Markup Language (HTML) file, which is a static resource. JSP actually uses HTML tags for page layouts, and the two appear identical in-browser, since server-side instructions are stripped from JSP once the page has been processed. Another similarity between the two is that JSP uses CSS for styling and JavaScript for running client-side scripts. However, JSP contains special

functionalities that allow it to dynamically generate content, unlike HTML. JSP is able to communicate with Java servlets (code located on a server) using "beans." JavaServer Page beans are created with the `<jsp:useBean>` tag ("Introduction to Developing Web Applications"). Beans must specify the class for which it is to communicate with using the Java Naming and Directory Interface (JNDI) to locate a Java file in the `src` directory on which the application is running. Source packages in Java are named like a reverse URL; a package located on this website (recsfor.me) would be prefixed with `me.recsfor`; however, JNDI does not accept any hyphens (which are replaced with underscores); additionally, a directory that starts with a number must have an underscore inserted before it ("Naming a Package"). Once the bean has been configured, it can be used to get and set properties of the corresponding servlet using the `<jsp:getProperty>` and `<jsp:setProperty>` tags ("Introduction to Developing Web Applications"). Properties can be obtained from user input in the shape of forms or buttons created with regular HTML. JSP also has more advanced capabilities that I would like to delve into as a part of further research.

My other research topic for this journal is learning about database usage in Java. Structured Query Language (SQL), by far the most popular relational database type in the world, seemed like the natural one to use. There are a number of different flavors of SQL: PostgreSQL and MySQL being the more popular free versions; there is also Oracle Database, which is a paid software more suitable for business applications. Microsoft offers two different database software: Microsoft Access (included in some editions of Office) and SQL Server. From all of these options, I chose to go with SQL Server, with which I got help setting it up from my advisor. All databases in Java are interacted with using Java Database Connection (JDBC), with

each type of database requiring its own driver that must be installed on the server the web application is being run on (Ganfield). The database itself can then be accessed by the server using either a standard deployment descriptor (`web.xml` or `web.config` file), or Persistence. The deployment descriptor implementation allows for `<sql>` tags to be embedded in the JSP, to access and modify the database from within the markup. The Persistence implementation requires its own Java "entity class" that represents the database itself, and code annotations to lay out relations ("Introduction to Java EE Technology"). I couldn't get either of these implementations working, so I'll need to look into that as a part of future research.

ACCOMPLISHMENTS

- Scripts in npm to compile JavaScript files and optimize image files for faster loading

```
"scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "webpack --progress -p",
    "watch": "webpack --progress --watch",
    "optimize": "for %f in (./web/img/*.png) do oxipng -o 4 -s \"./img/%f\""
},
```

- Readme in repository to explain the project
- Working usage of a JSP bean

```
<jsp:useBean id="search" scope="request" class="me.recsfor.search_engine.MediaQuery" />

<jsp:setProperty name="search" property="query" />

<h1><jsp:getProperty name="search" property="query" /></h1>
```

Reflection

My main holdup was that Amazon Web Services didn't approve my application for student benefits for nearly a month. Even now, I'm having issues doing anything with the account, as the student account is used different from a regular account in a way that is not very well documented. I may just look into other hosting options. I also had trouble getting my database to work with the web application, but in the interest of getting more done with the program itself, I may just make that a lower priority and not store any query data locally. This will have an impact on performance, because I'll need to submit an API call for every query,which is especially an issue because I'm limited to 1000 calls per day from OMDb. I chose to go with that service firstly because IMDb doesn't seem to have a true API, and secondly because several IMDb features appear to be going pro-only, and I really don't want to spend any money. On the bright side, I'm happy that I at least got a basic JSP configuration working for the web application. I'm hoping to at least expand its functionality a little bit before mid-year presentations.

Works Cited

Ganfield, Ken. “Getting Started with Java EE Applications.” *NetBeans IDE Tutorial*, Oracle, netbeans.org/kb/docs/javaee/javaee-gettingstarted.html.

“Introduction to Developing Web Applications.” *NetBeans IDE Tutorial*, Oracle, netbeans.org/kb/docs/web/quickstart-webapps.html.

“Introduction to Java EE Technology.” *NetBeans IDE Tutorial*, Oracle, netbeans.org/kb/docs/javaee/javaee-intro.html.

“Naming a Package.” *The Java™ Tutorials*, Oracle, docs.oracle.com/javase/tutorial/java/package/namingpkgs.html.