Homework 8: Stock Search using Yahoo Finance & Facebook Mashup

(AJAX/JSON/AWS/Java Exercise)

1. Objectives

- Become familiar with the AJAX, JSON & XML technologies.
- Use a combination of HTML, CSS, DOM, XMLHttpRequest, XML and Java Servlets.
- Get hands-on experience in Amazon cloud computing (AWS)
- Get hands-on experience on how to use YUI library to enhance the user experience
- Provide an interface to perform stock search from Yahoo! and post details to Facebook.

2. Background

2.1 AJAX & JSON

AJAX (**A**synchronous **Ja**vaScript + **X**ML) incorporates several technologies:

- Standards-based presentation using XHTML and CSS;
- Dynamic display and interaction using the Document Object Model (DOM);
- Data interchange and manipulation using XML and XSLT;
- Asynchronous data retrieval using XMLHttpRequest;
- JavaScript binding everything together.

See the class slides at **http://www-scf.usc.edu/~csci571/Slides/ajax.ppt** .

JSON, short for JavaScript Object Notation, is a lightweight data interchange format. Its main application is in AJAX web application programming, where it serves as an alternative to the use of the XML format for data exchange between client and server. See the class slides at

http://www-scf.usc.edu/~csci571/Slides/JSON1.ppt .

2.2Yahoo! Finance

Yahoo Finance is an online service to get free stock quotes and up to date news. See:

http://finance.yahoo.com/

In homework #6, a PHP script together with your Apache server provided the search functionality. In this exercise you will re-use your code to produce XML instead of HTML as you did in homework #6, plus you will enhance your PHP program to output a stock chart in

addition to the current stock quotes and news using the Yahoo API.

2.3 YUI Library

YUI is a free, open source JavaScript and CSS library for building richly interactive web applications. In this homework, you must use the YUI library (http://yuilibrary.com/) to enhance the user experience. You must apply its CSS style on the form elements and must also use the 'Auto complete' and 'TabView' widgets to display the stock information.

2.4 Facebook

Facebook is a global social networking website that is operated and privately owned by Facebook, Inc. Users can add friends and send them messages, and update their personal profiles to notify friends about themselves and what they are doing. Users can additionally post news feeds to their profiles, and these feeds may include images, besides text messages.

The Facebook homepage is available at:**http://www.facebook.com**

Facebook provides developers with an API called the **Facebook Platform**. **Facebook Connect** is the next iteration of Platform, which provides a set of API's that enable Facebook members to log onto third-party websites, applications and mobile devices with their Facebook identity. While logged in, users can connect with friends via these media and post information and updates to their Facebook profile.

Below are few links for Facebook Connect:

http://developers.facebook.com/blog/post/108/

http://developers.facebook.com/docs/guides/web/

2.5 Amazon Web Services (AWS)

AWS is Amazon's implementation of cloud computing. Included in AWS is Amazon Elastic Compute Cloud (EC2), which delivers scalable, pay-as-you-go compute capacity in the cloud, and AWS Elastic Beanstalk, an even easier way to quickly deploy and manage applications in the AWS cloud. You simply upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server PHP and Python, Passenger for Ruby, IIS 7.5 for .NET, and Apache Tomcat for Java.

The Amazon Web Services homepage is available at:

http://aws.amazon.com/

3. Description of the Exercise

In this exercise, you will write a web application that performs following functionality:

- The top-level interface consists of the following:
  - A form including a text area to enter the location and a “Search” button.
  - A dynamic area that displays the results.
- A form allows a user to enter a query(company symbol or company name) to retrieve information (quote information, news and stock chart) from Yahoo!. See the section titled “Autocomplete List” on page 11 for details on how to obtain a company symbol from a company name. Once the company symbol has been entered or retrieved and the “Search” button has been pressed, the form calls a JavaScript function for validation.

Figure 1 – Enter company symbol or company name

- Your JavaScript code will have to perform necessary validation for input query. E.g. On an empty input string you must show an alert message.

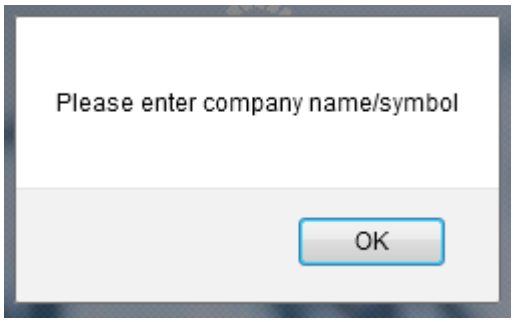

Figure 2 – Malformed Input Alert

- Once the validation is successful, the JavaScript function executes XMLHttpRequest to start *an asynchronous* transaction with a Java Servlet running under Tomcat, and passing the company symbol as a parameter of the transaction.
- The Java Servlet extracts the company symbol parameter and then it calls the PHP script on AWS, modified after Homework #6, to retrieve data from Yahoo. You should pass company symbol as a parameter of the transaction when calling the PHP script.

  For example, if your AWS server domain is called default-environment-randomstring.elasticbeanstalk.com and user enters ‘GOOG’ as the company symbol, then a query of the following type will be generated:

http://default-environment-randomstring.elasticbeanstalk.com/?symbol=GOOG

- The PHP script running on AWS would extract the symbol parameter from the request, query the Yahoo Finance web service, retrieve the results, and respond by returning the XML of the following format:

```
<result>
   <Name>Google Inc.</Name>
   <Symbol>GOOG</Symbol>
   <Quote>
      <ChangeType>-</ChangeType>
      <Change>3.5599</Change>
      <ChangeInPercent>0.29%</ChangeInPercent>
      <LastTradePriceOnly>1,215.65</LastTradePriceOnly>
      <PreviousClose>1,219.21</PreviousClose>
      <DaysLow>1,206.22</DaysLow>
      <DaysHigh>1,224.19</DaysHigh>
      <Open>1,220.34</Open>
      <YearLow>761.26</YearLow>
      <YearHigh>1,228.88</YearHigh>
      <Bid>1,213.50</Bid>
      <Volume>2,314,596</Volume>
      <Ask>1,214.94</Ask>
      <AverageDailyVolume>2,113,130</AverageDailyVolume>
      <OneYearTargetPrice>1,318.13</OneYearTargetPrice>
      <MarketCapitalization>408.5B</MarketCapitalization>
   </Quote>
   <News>
   <Item>
   <Title>Google cameras take rafting trip at Grand Canyon</Title>
   <Link>http://us.rd.yahoo.com/finance/news/rss/story/*http://fin
   ance.yahoo.com/news/google-cameras-rafting-trip-grand-
   202517039.html</Link>
   </Item>
   <Item>
   <Title>Google's New Store Sounds More Like A Tourist Museum</Title>
   <Link>http://us.rd.yahoo.com/finance/news/rss/story/*http://fin
   ance.yahoo.com/news/googles-store-sounds-more-tourist-
   000850968.html</Link>
   </Item>
   </News>
   <StockChartImageURL>
   http://chart.finance.yahoo.com/t?s=GOOG&lang=enUS&amp;width=
   300&height=180
   </StockChartImageURL>
</result>
```

- Notice that in Homework #6 your PHP script produced HTML. In this exercise, the

output must be changed to XML and the PHP code must run on AWS. Also note that, apart from the quote information and news, your PHP script must also return a stock chart image URL as a part of XML response.

You should use the following Yahoo Finance API to generate and retrieve the stock chart URL:

http://chart.finance.yahoo.com/t?s=GOOG&lang=en-US&width=300&height=180

Use the extracted company symbol instead of 'GOOG' in the above URL.

Also, make sure all your numbers should display the thousand operator (1500 → 1,500). And if the number has decimal points it should display only two decimal points (1500.562 → 1,500.56).

- Upon receiving the XML response, the Java Servlet extracts the information from this XML and converts it into a JSON string that is returned asynchronously to the original XMLHttpRequest.

  The format of the JSON string that needs to be generated is as follows:

```
{
   "result":{
      "Name":"Google Inc.",
      "Symbol":"GOOG",
      "Quote":{
         "ChangeType":"-",
         "Change":3.5599,
         "ChangeInPercent":"0.29%",
         "LastTradePriceOnly":"1,215.65",
         "Open":"1,220.34",
         "YearLow":761.26,
         "YearHigh":"1,228.88",
         "Volume":"2,314,596",
         "OneYearTargetPrice":"1,318.13",
         "Bid":"1,213.50",
         "DaysLow":"1,206.22",
         "DaysHigh":"1,224.19",
         "Ask":"1,214.94",
         "AverageDailyVolume":"2,113,130",
         "PreviousClose":"1,219.21",
         "MarketCapitalization":"408.5B"
      },
      "News":{
         "Item":[
            {
"Link":"http://us.rd.yahoo.com/finance/news/rss/story/*http://finance.y
ahoo.com/news/google-cameras-rafting-trip-grand-202517039.html",
```

```
"Title":"Google cameras take rafting trip at Grand Canyon"
            },
            {
"Link":"http://us.rd.yahoo.com/finance/news/rss/story/*http://finance.y
ahoo.com/news/googles-store-sounds-more-tourist-000850968.html",
"Title":"Google's New Store Sounds More Like A Tourist Museum"
            }
         ]
      },

"StockChartImageURL":"http://chart.finance.yahoo.com/t?s=GOOG&lang=en-
US&amp;width=300&height=180"
   }
}
```


- After obtaining the query results from the callback of XMLHttpRequest, the JavaScript program displays the appropriate table in the "dynamic" area of the web page. **Also note that successive queries will clear the data of the dynamic area and overwrite it with new data.**
- A dynamic area displays top and bottom sections:

  The Top section consists of Headline, Stock Chart and Facebook button.

  - **Headline:** Shows company name, company symbol and three numbers. Table 1 shows the mapping between the JSON response and the result table. If the value of ChangeType is + you should display the second and the third numbers in the headline in green and should show the UP green arrow. Otherwise, the second and the thirds numbers should be displayed in red and the DOWN arrow should also be displayed. The images for the red and green arrows are available at:

    http://www-scf.usc.edu/~csci571/2014Spring/hw6/down_r.gif
    http://www-scf.usc.edu/~csci571/2014Spring/hw6/up_g.gif

Figure 3 – Headline

  - **Stock Chart:** The stock chart image using the "StockChartImageURL" obtained in the JSON response.

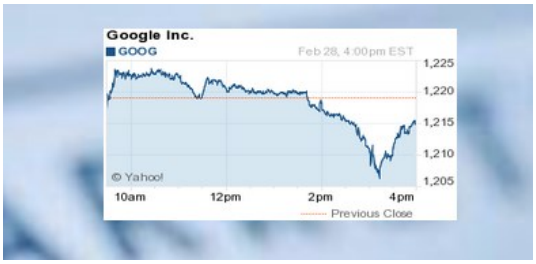

Figure 4 – Stock Chart

- **<u>Facebook Button:</u>** Below the headline, there should be a Facebook button, which allows the user to post Stock information of corresponding company. When the button is pressed, the web application does the following:

  - Authorizes the user to Facebook (i.e. logs him/her in) using the application and user credentials if the user is not already logged in to Facebook;
  - Posts an Update Status message to the feed.
  - The above two steps can be performed using the Facebook Connect API, using the JavaScript SDK, which provides a rich set of client-side functionality for accessing Facebook's server-side API calls. It is documented at: https://developers.facebook.com/docs/reference/javascript/

    The format of the feed to be posted is as follows:

    Company Name
    Stock Information of *Company Name(Company Symbol)*
    Last Trade Price: *XXXXX,* Change: *+/-XXXX*(*XXX%*)

    Enter appropriate values for Company Name, Company Symbol, Last trade price and Change. The Company Name (first line) should be hyperlink to Yahoo! Finance service page of corresponding company. The stock chart image must be displayed as a part of the post.

Figure 5 – Top Section(Headline, Stock Chart and Facebook button)

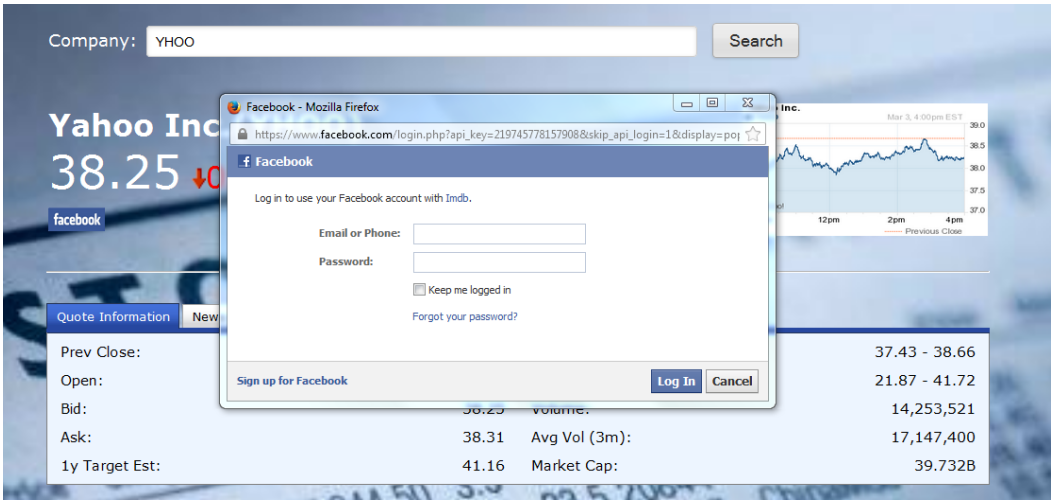

Figure 6 – Facebook Login

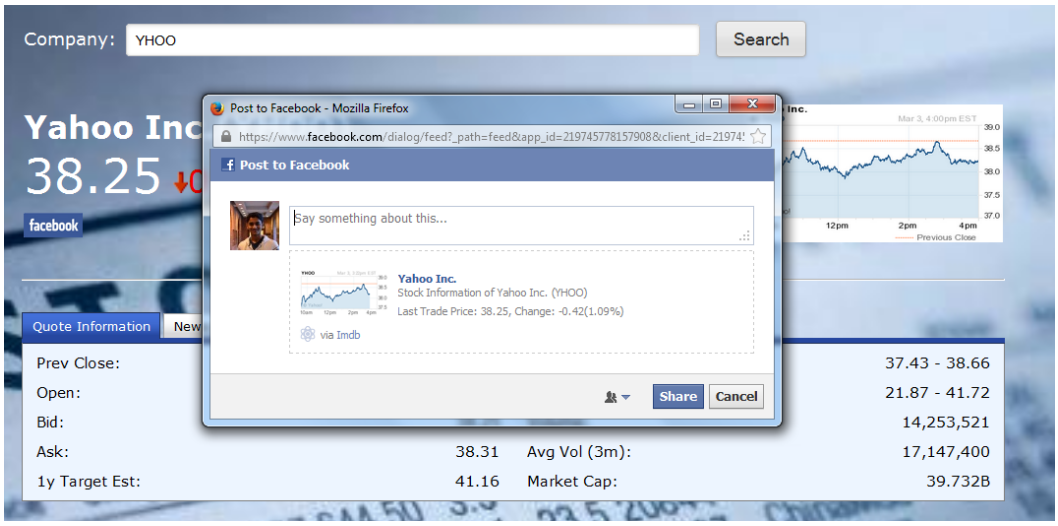

Figure 7a – Posting the stock information on Facebook feed

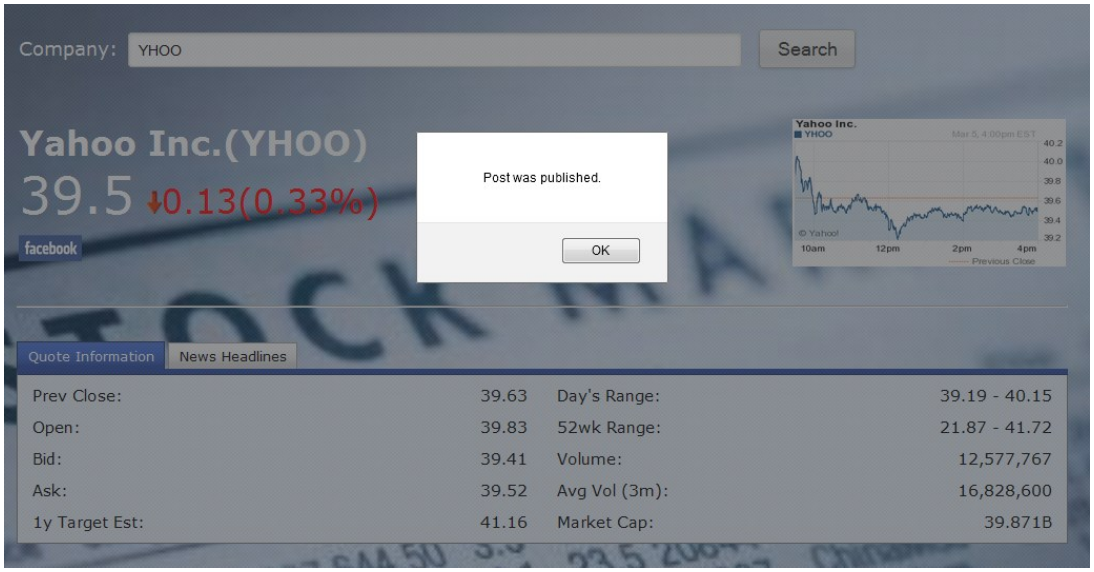

Figure 7b – Posting the stock information on Facebook feed

The Bottom section consists of a tab view for the remaining quote information and news

- **Quote Information and News Tabs:** Shows two tabs(Quote Information, News) using YUI's TabView widget. You can find more information on Tab View widget here: http://yuilibrary.com/yui/docs/tabview/.

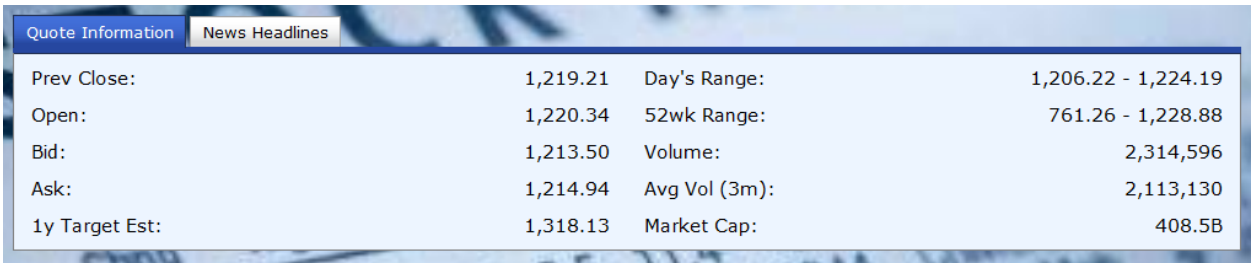

Figure 8 – Quote Information Tab

News are listed in an unordered list. Each list item should display the value of the "title" field which points to the link mentioned in the "link" field. In the unordered list, when clicking on any news item, a new tab must be opened.

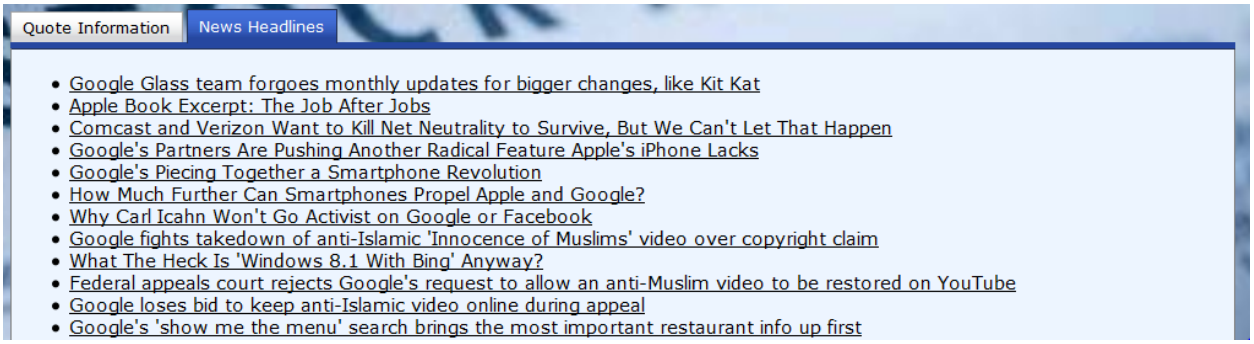

Figure 9 – News Tab

Result Table	Fields from JSON response
Company Name	Name
Company Symbol	Symbol
First Number in the headline	LastTradePriceOnly
Second Number in the headline	Change
Third Number in the headline	ChangeinPercent
Prev. Close	PreviousClose
Day's Range	DaysLow - DaysHigh
Open	Open
52wk Range	YearLow - YearHigh
Bid	Bid
Ask	Ask
Avg Vol	AverageDailyVolume
1y Target Est	OneyrTargetPrice
Market Cap	MarketCapitalization

Table 1. Mapping between result table and JSON response

- If the stock information cannot be found, an appropriate message should be shown as in the following snapshot.

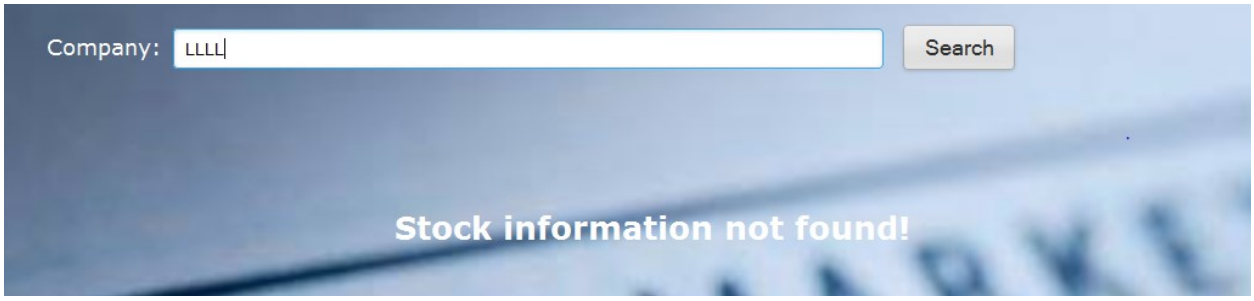

Figure 10 – Information not found output

The complete output of your application should look like in Figure 8.

Figure 11 – Stock Search Web Application

- **Autocomplete List**

Apart from the dynamic area mentioned above, you must implement the Autocomplete list feature for the input textbox using YUI's Auto Complete widget. Autocomplete involves the program predicting a word or phrase that the user wants to type in without the user actually typing it in completely. In our case, you should generate and show the autocomplete list of suggested company symbols based on the characters/phrase typed by the user in textbox.

You should populate the autocomplete list using the data obtained from the following Yahoo Finance API:

http://autoc.finance.yahoo.com/autoc?query=google&callback=YAHOO.Finance.SymbolSuggest.ssCallback

Replace "google" with the query entered by the user. Note that this is a JSONP API which returns the JSON data wrapped inside the callback function *ssCallback*.

Each row in the autocomplete list must be formatted in the following manner:

<Company Symbol>, <Company Name>(<Exchange>)

Figure 12 – Autocomplete List

You can find more information about YUI autocomplete widget here:

http://yuilibrary.com/yui/docs/autocomplete/

Once you have implemented the autocomplete feature, the user can request information by following any one of three methods:

1. type into textbox and click on the search button.
2. type into textbox and select from the drop-down list using mouse click selection.
3. type into textbox and press enter*(In this case the current active item on the list will be searched. By default the first item in the list must be the active item.)*

4. Implementation Hints

- *Populating the autocomplete list*
  Set the 'source' attribute of YUI's autocomplete plugin to a function to use that function as the result source. In the function make an asynchronous call to Yahoo's symbol suggest JSONP API.

- *Activating the first item in autocomplete list*
  To set the first item on the list as active item by default use "activateFirstItem: true" flag in YUI's autocomplete plugin.

- *Settingup an Ajax transaction*

  The JavaScript invoked by the Search button click event should do all of the following:

  - Assign the "callback" function;
  - Assemble the "url" parameter of the GET request as a reference to the Java

Servlet to be invoked:

Example url :http://cs-server.usc.edu:XXXXX/servlet/MyServlet?parameters

- Call the XMLHttpRequest method and create the request object.
- Prepare the GET XMLHttpRequest using the setRequestHeader method:

```
req.open("GET", url, true);

req.onreadystatechange = myCallback;

req.setRequestHeader("Connection", "Close");

req.setRequestHeader("Method", "GET" + url + "HTTP/1.1");
```

- *Executing the Ajax Transaction*

  The JavaScript should finally invoke the XMLHttpRequest`send` method (see Ajax slide 24).

  The "callback" function should check for completion of the transaction (request `readyState` equal to `4` and `status` equal to `200`(see AJAX slide 27 and JSON slide 5); use `eval()` and the `responseText` to retrieve the resulting JSON data (see JSON slide 5), and display the returned information properties to the "dynamic" area.

- *Using the Java Servlet to respond to XMLHttpRequest*

  The Java Servlet referred above (in step 1) as `/servlet/MyServlet` should be invoked using `doGet()`.

  The Java Servlet should initiate a connection with the modified PHP script(deployed on AWS) from Homework #6, to retrieve the appropriateinformation from Yahoo!.

- *Using the Java Servlet to retrieve the XML file content*

  You may have to use an XML parser (for example, JAXP or JDOM). The steps to retrieve XML file contents are as follows:

  Step a: Get the XML content based on the URL above.

  - You need to open a URL connection to get the file you want.

    To create a URL connection to AWS:

```
URL url = new URL(urlString);

URLConnectionurlConnection = url.openConnection();
```

```
urlConnection.setAllowUserInteraction(false);

InputStreamurlStream = url.openStream();

//read content
```

Step b: Parse the XML file using an XML parser

- Any XML parser can be used to parse the XML file. You can use methods like **getNodeName()** to access these elements. A good choice might be the JDOM library, which you get from:**http://www.jdom.org/downloads/index.html**

- *Using the Java Servlet to process the XML data*

As you parse the data, you will build an output string, converting the XML data into JSON format, as described in section 3.

Finally you will return the JSON as a single string to the calling JavaScript program. To easily create a JSON string, you might find useful the JSON-RPC library available at:

http://mirrors.ibiblio.org/pub/mirrors/maven/com.metaparadigm/jars/json-rpc-1.0.jar

The Java Servlet should handle exceptions such as MalformedURLException and IOException. The Java Servlet should also handle error responses sent from the PHP code running on AWS and reply with an appropriate error, a JSON message to the initial JavaScript XMLHttpRequest. This way, the JavaScript callback function will be able to inform the user that an error has occurred.

- *Authorizing Facebook User*

Once the user clicks on the Facebook button, the program invokes the Facebook Connect API and authorizes the user. The recommended API to use is FB.Init, which is documented at: **http://developers.facebook.com/docs/reference/javascript/FB.init/**

Also look at the code listed under "Loading" in the JavaScript SDK page at:

https://developers.facebook.com/docs/reference/javascript/

- *Post media information to Facebook News Feed*

There are several methods to post a message to the user's feed page (the "wall").

One such method uses the Fb.ui() API with the feed dialog, documented at:

http://developers.facebook.com/docs/reference/javascript/FB.ui/

The feed dialog is documented at:

https://developers.facebook.com/docs/reference/dialogs/feed/

Once the user is authorized, and an appropriate session token has been obtained, the text of the selected entry is posted to the user's news feed page (the "wall"). A subsequent posting will not require the user to log in again.

Additional information that may be useful to you is available at "Facebook for Websites" web page:https://developers.facebook.com/docs/guides/web/

5. Prerequisites

This homework requires the use of the following components:

- A servlet-based web server, Tomcat 4.1.27. Instructions on how to load Tomcat 4.1.27 can be found here:
  http://www-scf.usc.edu/~csci571/2013Spring/tomcatinstall.html.
  A tar version of Tomcat 4.1.27 can be found here:
  http://www-scf.usc.edu/~csci571/download/jakarta-tomcat-4.1.27.tar.
- The Java Servlet library (servlet-api.jar) to perform HTTP transactions using methods such as doGet() or doPost() from Java.
  You may find it here: http://repo1.maven.org/maven2/javax/servlet/servlet-api/2.5/
  or you can use the one located in **../jakarta-tomcat-4.1.27/common/lib/servlet.jar**
- A Java XML parser library. You may use the JDOM, an object model that uses XML parsers to build documents, available in the Download section of the jdom website
  http://www.jdom.org/downloads/index.html
- You may also use JAXP, the Java API for XML Parsing. A good tutorial on JAXP is available at http://www-106.ibm.com/developerworks/xml/library/x-jaxp1.html
  Orhttp://docs.oracle.com/javase/tutorial/jaxp/intro/index.html
- The YUI library, which can be downloaded from http://yuilibrary.com/.
- An AWS account. Signup and installation of your Apache+PHP server on AWS is available at http://www-scf.usc.edu/~csci571/2013Fall/hw7/aws.pdf. A $100 credit for AWS services will be provided to you by e-mail.
- You need to create a Facebook Platform application:
  To do that you will need to add the Facebook Developer application: go to https://developers.facebook.com/ and in the apps section, click **Create New App**. Once you've added the Facebook Developer application to your account, you can begin creating your application for Facebook. You should be getting an **API Key** and **Application Secret** that you will have to use with the JavaScript Client Library FB.init API.

6. Deployment Structure

To write your own Java Servlets program using Tomcat 4.1.27, you need to:

- Successfully install Tomcat 4.1.27 on your machine.
- Go to **$CATALINA_HOME/webapps/examples** directory.
- Place the HTML, CSS and JavaScript (**.js**) files in the Tomcat **servlets** subdirectory.
- Place your Java Servlets file (**.java**) in the**/WEB_INF/classes** folder. So the path of your Servlets file is **http://server_name:port/examples/servlet/your_servlet_name**
- Add appropriate sections to the **WEB-INF/web.xml** file, as in:

```
<servlet>
<servlet-name>finance_search</servlet-name>
<servlet-class>FinanceSearch</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>finance_search</servlet-name>
<url-pattern>/servlet/financesearch</url-pattern>
</servlet-mapping>
```

- To avoid UTFDataFormatException during file IO operation, you have to use JDK 1.3 or later for Tomcat. In the **.cshrc** file under your home directory, add the entries:

```
setenv JAVA_HOME /usr/j2se
setenv PATH /usr/j2se/bin:${PATH}
```

- Before you issue a request to your Java Servlet file, you need to compile it. You will need a Java Servlet class to compile your code, so open the .cshrcfile, and add the full path to the Tomcat file that implements the Servlet class APIs located in **../jakarta-tomcat-4.1.27/common/lib/servlet.jar**to your CLASSPATH variable.
- Then run "**source .cshrc**" and you are ready to compile your Java files.

7. Material You Need to Submit

On your course homework page, your link for this homework should go to a page that includes your JavaScript/HTML program. You should submit all source code files including HTML (.html), Images (.png, .jpg or .gif), Cascading Style Sheets (.CSS), JavaScript (.js), PHP scripts(.php) and Java Servlets (.java) electronically to the csci571 account so that it can be graded and compared to all other students' code via the MOSS code comparison tool.

**IMPORTANT Note:

- All discussions and explanations in Piazza related to this homework are part of the homework description. So please review all Piazza threads before finishing the assignment.
- In your Facebook application settings, you should go to the "Status & Review" section

and choose “Yes” for the question “Do you want to make this app and all its live features available to the general public?” as shown in the figure 13. If you answer “YES”, anyone will be able to log in through her/his Facebook account and post via your web interface. Otherwise, the developer will be the only person who is able to use the Facebook functionality in the web page. If graders try to test the functionality of Facebook button and they are not able to log in you will lose **all** points related to the Facebook component in your homework grade.

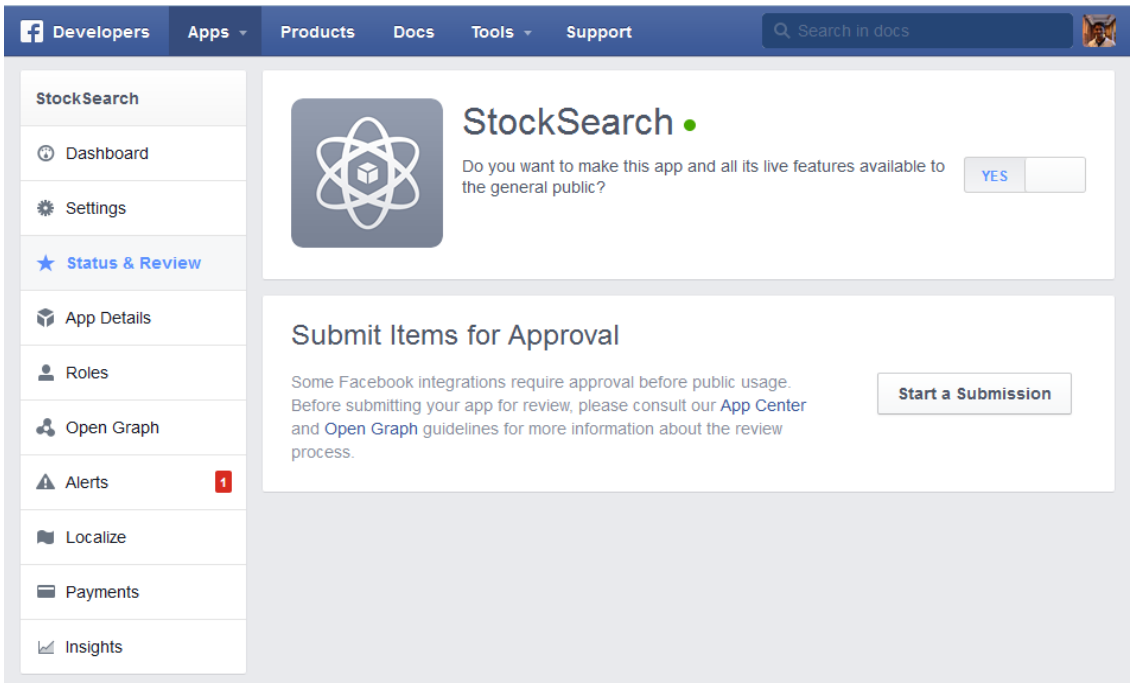

Figure 13: Making the app available to public (Disabling the Sandbox mode)