Assignment 6: JavaScript and Cloud Computing

Release Date: November 22, 2016

Due Date: December 4, 2016 (11:59 pm Central Time)

(This is an individual assignment)

There are two important dates for this assignment: December 2, 2016 and December 4, 2016.

Midnight of December 2: Submit first version of the assignment by this date. The assignment does not have to work completely, but it has to be at the level where all startup related issues have been resolved (Eclipse, Tomcat, JavaScript, REST API, etc.). This submission will be graded for 3 points. This means if your first commit is on or before December 2nd , you will receive points out of 100. If you only have one submission (after December 2nd , you will receive points out of 97). The reason to have this deadline is to encourage you to get started on the assignment well ahead of the actual deadline.
Midnight of December 4: Deadline by which you should have completed your assignment.

1) Goal:
In this assignment you will learn about JavaScript and Cloud Computing

2) High-level Description:
Using HTML, JavaScript, and REST APIs build the following "analytics" display for *Solum's meetings*
The output should be as follows:

Year	Number of meetings
2013	<number of meetings in 2013>
2014	<number of meetings in 2014>
2015	<number of meetings in 2015>
2016	<number of meetings in 2016>

3) Details:

Part I: Calculating meeting counts
Solum team meeting logs are available at following *source:*
http://eavesdrop.openstack.org/meetings/solum_team_meeting/
For each meeting four different files are generated. For example, if you check this link:
http://eavesdrop.openstack.org/meetings/solum_team_meeting/2013/
You will notice there are four files (.log.html, .log.txt, .html, .txt) corresponding to a meeting date. For instance, for 2013-11-12, there are following files in the folder:
solum_team_meeting.2013-11-12-16.00.html
solum_team_meeting.2013-11-12-16.00.log.html
solum_team_meeting.2013-11-12-16.00.log.txt
solum_team_meeting.2013-11-12-16.00.txt
For calculating the counts, you can either use the meeting names, or the last modified timestamp. But you need to ensure that you count each meeting date only once in the overall meeting count. So essentially, the meeting on 2013-11-12 should contribute 1 to the overall meeting count, even though there are four files generated for each meeting.

Part II: HTML, JavaScript, REST API
The entry point to your web application should be a HTML file whose URL should be:
http://localhost:8080/openstack-meeting-analytics.html

This HTML page should have following text (you don't have to worry about making the text color blue):

Welcome to OpenStack Solum Meeting Analytics
Click here to see number of meetings for Solum OpenStack project

Clicking on "here" should show the above table.
You are required to use HTML and JavaScript for building the UI (the initial HTML page and the page that displays table).
You will need to build a REST API that can be invoked from the JavaScript to get the meeting count data, which will be rendered by the JavaScript.

Part III
Deploy your web application in the "cloud" and make it accessible over the Internet. You can use any of the several commercial cloud services for this, such as:

- Amazon AWS (http://aws.amazon.com/)
- Heroku (https://www.heroku.com/)
- Your-choice-of-cloud-provider

Once deployed, your web application should be available at:
http://<IP-address-of-your-VM>:8080/openstack-meeting-analytics.html or
http://<platform-specific-url>/openstack-meeting-analytics.html

4) Design details:
You can calculate the counts on the fly, every time your web application starts up. You will have to expose a suitable REST API that can be used by your web application's JavaScript AJAX requests. It is up to you to define a REST API for this purpose.

5) Grading criteria:
1) Correct counts displayed for each year
2) Appropriate REST API defined
3) Interaction between JavaScript and REST API happening correctly
4) Web app runs correctly and is accessible publicly
5) If #4 fails, then we will consider if the app runs correctly on a non-public setup. But you will not get points for cloud deployment (point split tbd).

6) Submission Details:
Create the following folder structure:
assignment6/src/main/java/<Java classes>
assignment6/src/main/webapp/<*.html>
assignment6/src/main/webapp/<*.js>
assignment6/WebContent/WEB-INF/web.xml
assignment6/pom.xml
assignment6/README
Feel free to add more packages inside the "src/main/java" to segregate work into different layers.
Use following format for README.yaml

```
---
name: SUBSTITUTE_WITH_YOUR_NAME
eid: SUBSTITUTE_WITH_YOUR_EID
appURL: <Public URL of your application>
```

```
cloud: <Name of the cloud provider you chose>
notes: |
   Notes
```

Push the entire assignment6 folder to your bitbucket account.
Grant “read” access to Eddy, Brandon, and me
  - Usernames: eddy_hudson, brandonhollowell, devdattakulkarni
We will use the latest commit sha for grading.

7) Helpful Material:
Lecture Notes: JavaScript, Cloud computing, Webapp-deployments-to-public-clouds

Relevant projects from github repository (https://github.com/devdattakulkarni/ModernWebApps/):
JavaScript-Example

Video showing how to deploy web app on cloud VM: https://vimeo.com/126759241