Assignment 3: REST API (GETs) and Functional Testing

Release Date: October 10, 2017
First Deadline: October 20, 2017
Due Date: October 22, 2017 (11:59 pm Central Time)
(This is an individual assignment)

Objective:
In this assignment you will learn about RESTEasy REST framework. It provides functionality that allows creation of JAX-RS based REST APIs.

Introduction:
Design a RESTEasy based REST API to re-model the eavesdrop website:
http://eavesdrop.openstack.org

Your task in this assignment is to create a REST API, which will invert the relationship between meetings and projects. So instead of /…/meetings/<project-name> resource model, which is used on the eavesdrop website, your REST API will model the resources as /…/projects/<project-name>/meetings.
Specifically, you will support following resources and actions on them. The exact requests and responses are colored in gray below.

1) A "projects" resource with GET action on it as follows:

```
GET http://localhost:8080/assignment3/myeavesdrop/projects/
```

This should return all the projects listed on: http://eavesdrop.openstack.org/meetings
Response should be in XML format as follows:

```
<projects>
        <project>3rd_party_ci/</project>
          :
        <project>zun/</project>
<projects>
```

2) A "meetings" resource modeled as a sub-resource of an individual project resource. Support GET action on it as follows:
GET http://localhost:8080/assignment3/myeavesdrop/projects/{project}/meetings
E.g.:

```
GET http://localhost:8080/assignment3/myeavesdrop/projects/3rd_party_ci/meetings
```

Response should be contents from http://eavesdrop.openstack.org/meetings/3rd_party_ci/
formatted as XML format as follows:

```
<meetings>
   <year>2014/</year>
</meetings>
```

Another example request is:

```
GET http://localhost:8080/assignment3/myeavesdrop/projects/solum_team_meeting/meetings
```

Response:

```
<meetings>
    <year>2013/</year>
    <year>2014/</year>
    <year>2015/</year>
    <year>2016/</year>
</meetings>
```

You don’t have to worry about maintaining a specific order for “year” XML elements in the response. The order can be whatever the order is on the eavesdrop website.
If a project does not exist on http://eavesdrop.openstack.org/meetings
your program should return “Project <project-name> does not exist”. For example,
Request:

```
GET http://localhost:8080/assignment3/myeavesdrop/projects/non-existent-project/meetings
```

Response:

```
<output>
   <error>Project non-existent-project does not exist</error>
</output>
```

You should treat project names as case sensitive. So ‘Solum_team_meeting’ should NOT be treated as solum_team_meeting
Request:

```
GET http://localhost:8080/assignment3/myeavesdrop/projects/Solum_team_meeting/meetings
```

Response:

```
<output>
   <error>Project Solum_team_meeting does not exist</error>
</output>
```

Design Details:
You need to use RESTEasy framework for this assignment. You should organize your code into two layers, resource/controller layer and a service layer. You can divide the responsibilities between these layers as follows. Use the resource/controller layer to perform input validation and response generation. Use the service layer to implement the logic of calling the eavesdrop website and parsing the result.
Additionally, you need to implement *functional tests* for the resource layer. Specifically, implement functional tests for following four resources:
1) GET http://localhost:8080/assignment3/myeavesdrop/projects/
2) GET http://localhost:8080/assignment3/myeavesdrop/projects/solum_team_meeting/meetings
2) GET http://localhost:8080/assignment3/myeavesdrop/projects/3rd_party_ci/meetings
3) GET http://localhost:8080/assignment3/myeavesdrop/projects/non-existent-project/meetings
A functional test tests that output of the specified <method, REST resource> combination is as expected. So in above cases, the tests need to first figure out what the correct output will be and write *asserts* that the output is as expected. For this you will need to parse XML output that your API response generates.
For asserting the response of the GET on “projects” resource (test case 1 above), it is sufficient to assert that any 10 projects from http://eavesdrop.openstack.org/meetings exist in the response.

Submission Instructions:
Create the following folder structure:

```
   assignment3/src/main/java/assign/domain/<Domain class>
   assignment3/src/main/java/assign/resources/<Resource class>
   assignment3/src/main/java/assign/services/<Service classes>
   assignment3/src/test/java/assign/resources/<Resource test classes>
```

assignment3/README.yaml
Use following format for README.yaml

Include following information in README.txt
name: <your name>
eid: <your ut eid>
bitbucketid: <your bitbucket id>
comments: <Comments, if any>

Create a *private* repository on bitbucket (https://bitbucket.org/product/pricing?tab=host-in-the-cloud sign up using the free option if you don't have bitbucket account). Name the repository assignment2 and push/submit all the files that you create for your assignment to this repository.
Grant "read" access to me and Sailesh
- Usernames: devdattakulkarni, svsailesh

We will use the latest commit ID for grading. You don't need to submit anything on canvas.

Helpful Material:
a) Lecture Notes: REST API, Functional testing, XML-HTML-Parsing
b) Book Chapters: Chapters 1, 2, 3, 4, 5, 6, 17, 18, 19 of RESTful Java with JAX-RS 2.0 book.
c) Github (https://github.com/devdattakulkarni/ModernWebApps)
 Projects: REST/resteasy, Servlets/test-jsoup-1, XMLParsing
d) The ex03_1 example code from RESTful Java book
https://github.com/oreillymedia/restful_java_jax-rs_2_0

Note that for the ex03_1 example, you should change the JUnit dependency in pom.xml to use version 4.12. The JUnit test runner in Eclipse does not seem to work with the version that is there by default (4.1) in pom.xml of ex03_1 example.

Getting started:
1) Use following getting starter template:
https://github.com/devdattakulkarni/ModernWebApps/tree/master/assignment-templates/assignment3

2) Read the lecture notes on REST API.
3) Download and try the resteasy example from course github page (mentioned in Helpful Material section above). Your goal should be to get this example working as the first step (either in Eclipse or Intellij or directly from command line (using mvn)).
4) Convert the resteasy example to the resource model that is required for this assignment.
5) Integrate JSoup with resteasy example.
6) Convert the domain objects from resteasy example to the domain objects required for this assignment
7) Read lecture notes on Functional testing, XML-HTML-Parsing
8) Try out the ex03_1 example from the O'Reilly book
9) Try out XMLParsing examples from course github page
10) Write functional tests for your assignment based on the learning from steps 8 and 9.

Deadlines:
First deadline: Initial submission which is will be graded for 3 points. You don't need to have assignment implementation completed by this date. This means if your first commit is on or before October 20 11:59 pm, you will receive points out of 100. If your first commit is after that then you will receive points out of 97. This deadline is to ensure that you complete all the required setup (Eclipse, Tomcat, etc.) early and not postpone it until last minute.

Due date: Deadline by which your assignment needs to be completed (October 22 11:59pm)

Late penalty:
5 points for each late day after the due date.

Collaboration policy:
This is an individual assignment. You are allowed to discuss concepts and high-level implementation questions with each other. But you are not allowed to copy or share code with each other or students who might have taken this class before. Final submission should be your own code.