Assignment 5: Hibernate
Release Date: November 8, 2017
First Deadline: November 17, 2017
Due Date: November 19, 2017 (11:59 pm Central Time)
(This is an individual assignment)

Objective:

In this assignment you will learn how to use Hibernate to interact with database for building data layer of a REST API.

High-level Description:

Implement following REST calls using Hibernate to interact with the database

- Create a project (POST)
- Create meeting for a project (POST)
- Update meeting information for a project (PUT)
- Delete a project (DELETE)
- Read a project (GET)

A project may have one or more meetings associated with it. You will implement a meetings sub-resource for a project resource. For interacting with the database through your Java code you need to use *Hibernate*.
The exact requests and responses are shown below:

1) Create project (same as assignment 4)
POST http://localhost:8080/assignment5/myeavesdrop/projects/

```
<project>
  <name>solum</name>
  <description>Project representing solum</description>
</project>
```

This should create project named solum in your system. You should create a unique projectId to represent the created project. The projectId will be used in subsequent calls. You should return the resource URL representing the newly created project as part of the location response header (similar to how you did this in assignment4).

2) Create a meeting for a project:
POST http://localhost:8080/assignment5/myeavesdrop/projects/<projectId>/meetings
Body:

```
<meeting>
  <name>m1</name>
  <year>2014</year>
</meeting>
```

This should create meeting named m1 and it should be associated with project with id “projectId”.
A meeting can be associated with only one project. You should create a unique meetingId to represent the created meeting.
Response:
If the request is successful:
Response status: 201 Created
Response body: Empty
Response Headers: Location Header with the URL of the meeting resource
If the request is unsuccessful based on validation requirements:
Response status: 400 Bad Request
Response body: Empty
If projectId does not exist in your database:
Response status: 404 Not Found
Response body: Empty

3) Get project details:
GET http://localhost:8080/assignment5/myeavesdrop/projects/<projectId>
Response XML:

```
<project id=``projectId">
  <name>cs378</name>
  <description>Project representing cs378</description>
  <meetings>
    <meeting id=``m1">
      <name>m1</name>
       <year>2014</year>
   </meeting>
    <meeting id=``m2">
       <name>m2</name>
       <year>2015</year>
  </meeting>
</meetings>
</project>
```

Response details:
If the request is successful:
Response status: 200 OK
Response body: XML representation of the requested resource (Response XML shown above)
If projectId does not exist in your database:
Response status: 404 Not Found
Response body: Empty

4) Update meeting:
PUT http://localhost:8080/assignment5/myeavesdrop/projects/<projectId>/meetings/<m1>
Body:

```
<meeting>
  <name>m1-name-changed</name>
  <year>2014</year>
</meeting>
```

This should change name of the meeting represented by meeting with id “m1” for project with id “projectId”.
Response:
If the request is successful:
Response status: 200 OK
Response body: Empty
If the request is unsuccessful based on validation requirements:
Response status: 400 Bad Request
Response body: Empty
If there is no project with id “projectId” or if there is no meeting with id “m1” then you should return a 404 Not Found response.
Response status: 404 Not Found
Response body: Empty

5) Delete project:
DELETE http://localhost:8080/assignment5/myeavesdrop/projects/<projectId>
This should delete the project with id “projectId” and all the meetings associated with it.
Response details:
If the request is successful:
Response status: 200 OK
Response body: Empty
If the request is unsuccessful based on validation requirements:
Response status: 404 Not Found
Response body: Empty

Design Details:

You need to use RESTEasy framework for building the API layer and *Hibernate* framework to interact with the database. You should organize your code into two layers, resource layer and a service layer. You can divide the responsibilities between these layers as follows. Use the resource layer to perform input validation and response generation. Use the service layer to implement database interactions using *Hibernate*.

Input validation:

Perform following checks on the input data:

•In the call to create a project (POST /projects/), ensure that name and description are not empty (“”) or does not contain only spaces.

•In the call to create a meeting (POST /projects/../meetings), ensure that name and year are not empty (“”) or does not contain only spaces. Moreover, year should be a valid year between 2010 - 2017 (inclusive).

•In the call to update a meeting (PUT /projects/../meetings), ensure that name and year are not empty (“”) or does not contain only spaces. Moreover, year should be a valid year between 2010 - 2017 (inclusive).

If the input data fails validation in any of the above cases, return HTTP 400 Bad Request response.

•In the calls to GET, PUT, or DELETE, if a project id or meeting id does not exist, return HTTP 404 Not Found response.

Database design:

You will need two tables to represent projects and meetings. For details on what columns to define, what should be the primary key, etc. refer to the project *Hibernate* on class github page.

You can install and run MySQL locally or you can use the one deployed for you by the CS department.

Tests:

You are not required to implement either functional tests or unit tests for this assignment. However, for the service layer, implementing test methods that execute the database calls would be helpful when you develop your Hibernate code.

Also, writing functional tests for the resource layer will be helpful when designing and debugging REST calls (specifically for response generation, content marshalling, etc.)

Submission Details:

1) Create the following folder structure:

```
assignment5/src/main/java/assign/domain/<Domain class>
assignment5/src/main/java/assign/resources/<Resource class>
assignment5/src/main/java/assign/services/<Service classes>
assignment5/src/main/resources/hibernate.cfg.xml
assignment5/README.txt
assignment5/WebContent/WEB-INF/web.xml
assignment5/pom.xml
```

2) Use assignment5/src/main/resources/hibernate.cfg.xml to define database credentials/parameters.

3) You will need to create the database that you will define in the “connection.url” string of hibernate.cfg.xml. Note that if you are using the CS database server then you don't need to create the database since it is already created for you. You can use it.

4) When submitting, set the url, username, password parameters in hibernate.cfg.xml as empty. We will add appropriate values for these when grading.

5) If you base your assignment on a sample project, before submission remove all the code in all the classes/files which is not relevant for the assignment.

6) 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 assignment5 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, Database-basics, ORM
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: Hibernate, REST, REST-JDBC
d) The ex03_1 example code from RESTful Java book
https://github.com/oreillymedia/restful_java_jax-rs_2_0

Hints on getting started:

1) Read the lecture notes on ORM.
2) Download and try the Hibernate example from course github page. 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)).
3) Convert the tables in the Hibernate example to the tables that are required for this assignment.
4) Convert the domain objects from Hibernate example to the domain objects required for this assignment.
5) Make sure that you are able to insert, update, query, delete items from the database using the test methods provided in that project.
6) Add RESTEasy specific things (resource definitions, marshaling, web.xml, etc.).
7) Modify the DBLoader to create new methods that you need for this assignment.
8) Modify the resource layer to use the new methods that you have created.
9) Test end-to-end.

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 November 17 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 (database connectivity, Eclipse, Tomcat, etc.) early and not postpone it until last minute.

Due date: Deadline by which your assignment needs to be completed (November 19 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.