SJSU CMPE 282 HW RESTful Web Svc SPRING 2016

Theme: Java-based REST server + NoSQL

Description

REST server side: please create Java-based RESTful Web Service server that retrieves/persists data from/to NoSQL according to the functional matrix specified later.

- NoSQL: choose either one of two most popular NoSQLs, Cassandra or MongoDB
- Install an instance of NoSQL for development purpose without relying on container (we will work on NoSQL on container in the next homework)

REST client side: you can use any REST client, such as RESTClient (Firefox), soapUI, curl, etc.

You may build your REST server in any way, such as using Servlet directly (and deploy to say Tomcat), or leverage Jax-RS (and then deploy to say Tomcat), or even use Spring (standalone or deploy to Tomcat), etc.

Data objects

There are two types of objects:

Employee

int id // user-generated unique value across all employees
String firstName
String lastName

Project

int id // user-generated unique value across all projects
String name
float budget

Format of HTTP/REST Request and Response

You can choose either XML or JSON. No need to have both.

For example, XML representations of these objects are as follows

```
<employee>
  <id>1</id>
  <firstName>John</firstName>
  <lastName>Doe</lastName>
</employee>
<project>
  <id>10</id>
  <name>projectX</name>
  <budget>1234.56</budget>
</project>
```

Environment

Use either Linux VM or Windows.

- To install a Linux VM, first install free VMware Player (Windows, Linux) or free Oracle VirtualBox (Mac) on your laptop, and then install a recent Linux such as Ubuntu desktop as guest OS.

- Linux VM is recommended since we will work on Linux VM only in the next homework. However, it is certainly possible to develop the REST server on Windows in this homework, and then move the code to Linux VM in the next homework.

Additional requirements

- REST resource (URI) must start with /cmpe282<YourName><L3SID>/rest/.. . For example, the entire URL would be http://hostname[:port]/cmpe282Demo123/rest/… The screenshots (explained later) must include the proper resource (URI).
- The database name (i.e., keyspace in Cassandra) of NoSQL must be cmpe282<YourName><L3SID>. The screenshots (explained later) must include the proper database name.
- Externalize the NoSQL configuration information, such as NoSQL IP, port, DB name, etc., as initial parameters, in say web.xml. This would become useful in HW2.

Questions

Q1.

a. List technologies and softwares (including version) for dev tools, REST client, and web/app component and (NoSQL) database component of REST server.
b. How did you build your REST serevr
c. If the NoSQL needs to create DB objects beforehand, such as Cassandra, include the script to create DB objects.

For example,

a.

Dev tools: Eclipse EE Luna SR1 on Ubuntu 14.04
REST Client: soapUI 5.2.1 on Ubuntu 14.04
REST Server: web/app component: Servlet, JAX-RS, Jersey, Jaxb, Tomcat 7.0.46, Jetty, etc. on Ubuntu 14.04
REST Server: (NoSQL) database component: Cassandra 2.2.4 on Ubuntu 14.04

b. REST Server: use Servlet directly without relying on JAX-RS and Spring. Then deploy to Tomcat.

c. Cassandra script: …

Q2.

Functionality matrix of the REST server:

HTTP method + URI	Description	HTTP status - successful	HTTP status –err, in addition to 400, 500
GET /…/rest/employee/m	Retrieve employee with id m. Resp body: XML or JSON	OK (200)	Not found (404)
GET /…/rest/project/n	Retrieve project with id n. Resp body: XML or JSON	OK (200)	Not found (404)
POST /…/rest/employee	Create a new employee. Req body: XML or JSON	Created (201)	Conflict (409)

POST /…/rest/project	Create a new project. Req body: XML or JSON	Created (201)	Conflict (409)
PUT /…/rest/employee/m	Update employee with id m. Req body: XML or JSON (partial properties to be updated is allowed)	OK (200)	Not found (404)
PUT /…/rest/project/n	Update project with id n. Req body: XML or JSON (partial properties to be updated is allowed)	OK (200)	Not found (404)
DELETE /…/rest/employee/m	Delete employee with id m	OK (200)	Not found (404)
DELETE /…/rest/project/n	Delete project with id n	OK (200)	Not found (404)
GET /…/rest/employee	Retrieve all employees. Resp body: XML or JSON (<employeeList>…</employeeList>)	OK (200)	Not found (404)
GET /…/rest/project	Retrieve all projects. Resp body: XML or JSON (<projectList>…</projectList>)	OK (200)	Not found (404)

For each of the function matrix,

- add the status column and specify its status (i.e., done, partial, none)
- If it's done
    - include before screenshot on DB side
    - include screenshot of REST request and response (method, URL, HTTP headers): error case
    - include screenshot of REST request and response (method, URL, HTTP headers): success case
    - include after screenshot on DB side, if it's PUT or POST
- if it's partially done, indicate its status

(Option) You can include the following info

- List known issues if any
- Any additional unique design or features you are proud of

Submission

Submit the followings as *separate* files to Canvas

- cmpe282<YourName><L3SID>.zip: zip the directory of .java sources, web.xml, and, if any, config files. Do *not* include .class, .jar, .war files.
- cmpe282<YourName><L3SID> (.war or .jar): a *deployable* .war file, or (mainly for Spring) a standalone .jar file.
- CMPE282_HW1_<YourName>_<L3SID> (.pdf, .doc, or .docx): the report consists of answers and screenshots to questions specified in **Question**.
    - You receive no credit if your report is not .pdf, .doc, or .docx.

- If a screenshot is unreadable, it will be treated as if you did not turn in that screenshot.
- If you do not follow requirements (including naming conventions), you will receive no credit.

The ISA and/or instructor leave feedback to your homework as comments and/or in Crocodoc of your submission. To access Crocodoc, click "view feedback" button. For details, see the following URL: http://guides.instructure.com/m/4212/l/106690-how-do-i-use-the-submission-details-page-for-an-assignment

Additional Info

- Get started early
- Design options
  - XML: jaxb vs. XPath vs DOM/SAX
    - jaxb: http://www.vogella.com/tutorials/JAXB/article.html
    - xml: http://www.vogella.com/tutorials/JavaXML/article.html
- Servlet: be able to handle GET, POST, PUT, and DELETE
  - http://www.tutorialspoint.com/servlets/
  - http://www.vogella.com/tutorials/EclipseWTP/article.html
  - In web.xml, url-pattern (within servlet-mapping) should be /rest/*
  - req.getPathInfo() returns string after /rest. E.g., it returns /employee/20 if URI is /rest/employee/20
- JAX-RS
  - http://www.vogella.com/tutorials/REST/article.html
  - http://crunchify.com/how-to-build-restful-service-with-java-using-jax-rs-and-jersey/
  - http://crunchify.com/create-very-simple-jersey-rest-service-and-send-json-data-from-java-client/
- Spring