Project 0: Cloud & Web Service Environment

Due by Midnight Thu Jan 22, 2015

0 points for assessment, but required to be completed

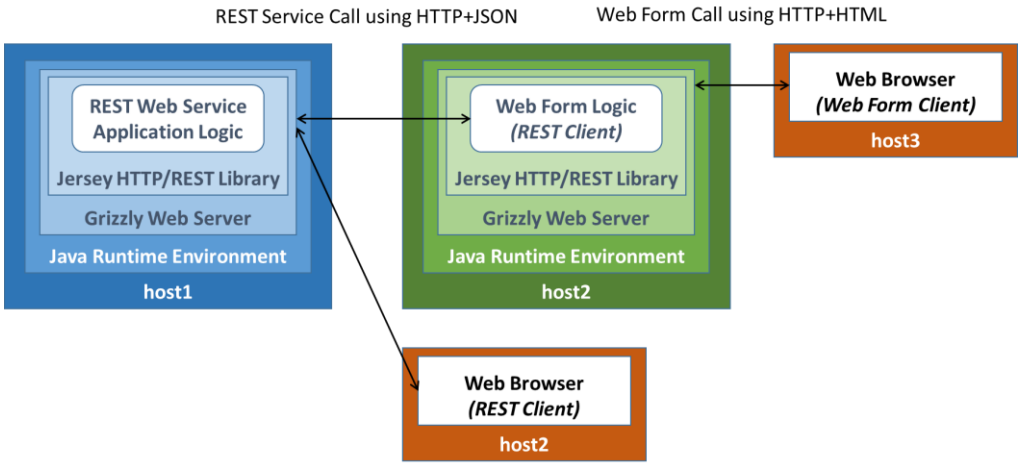
1. Goals

The intent of this project is to get the class familiar with the basics of building simple web services. We will REST web services with JSON as the specific technologies in the course project.

- At the end of Project 0, students should be able to *describe* and *demonstrate* (1) client-server pattern of interaction using synchronous execution, (2) remote procedure calls with marshalling and unmarshalling of parameters, and (3) coordinate execution of composite web services.
- They should also be able to *use* tool, libraries and languages to compose and invoke web services, and scripts to automate execution.

2. Project Tasks

- Download the project0-statup.zip file from the course webpage. Follow the instructions in the README.txt.



a. **Deploy & Test the REST Web Service:** Deploy the provided 'echo' REST web service whose application logic (`se252.jan15.calvinandhobbes.project0.EchoService`) implements the HTTP GET method using Jersey REST library while taking 'msg=$message' as a parameter. For e.g. **GET project0/echo?msg=Hello%20World HTTP/1.1** would be a sample request accepted by the service, where *$message* parameter is *Hello World*. The method responds with a JSON version of the text "I heard you say '$message'" as the response. E.g. `{"message" : "I heard you say 'Hello World'"}` would be the sample JSON response.

You should be able to launch the REST web service within the Grizzly Web Server using the provided `EchoServiceLauncher` on any machine (e.g., **host1**) from the commandline, as described in the

README.txt. Test this web service using a commandline or web browser client that can invoke this service from a different machine (e.g. **host2**) by taking the REST URL and the message as parameters. This client—service interaction should be demonstrated.

b. **Deploy & Test the Web Form:** A sample HTML web form (`EchoWebform`) that has a single *text box* to accept a message and a *submit button* that submits the form using a GET method is provided. The application logic of this web form will call the echo REST web service with the $message set to the contents of the text box, thus acting as a REST client. The received JSON response will be converted to an `EchoMessage` Java object by the Jersey REST library, and the form will then format and display the message in the object as an HTML web page.

   Once you launch the REST web service from step (a) above on **host1**, next launch the Grizzly web server for the web form (`EchoWebformLauncher`) on **host2**, passing commandline parameters as specified in the README.txt of the startup project. Access the web form from a web browser running on host3 (e.g., your laptop), submit the form and display the response in the browser. This interaction between browser (host3), web form/REST Client (host2) and web service (host1) should be demonstrated.

c. **Develop Composite Web Service:** Using the above service, form and client as a sample, you need to develop a new ***TripBuddy*** REST web service that takes two cities as source and destination for a particular traveller, and returns (i) the current weather in these two cities, (ii) the time zone difference between the cities, and (iii) Interesting activities in the destination. You will get the information required by (i-iii) by invoking external web services, such as [1,2,3]. The request to your REST service will be performed as a HTTP GET and the response should be in JSON returning just the information for (i-iii).

   You should provide and demonstrate a web form that takes the cities as input, invokes the REST service, and displays the response in a Web Browser. The web service invoked directly from a web browser to display the JSO response should also be demonstrated.

[1] Weather Underground Web Service, http://www.wunderground.com/weather/api/d/docs
[2] Google Timezone Service, https://developers.google.com/maps/documentation/timezone/
[3] Google Places, Bing Maps, etc.

3. Guidelines

- Read and follow the project tasks, guidelines and submission instructions carefully.
- You can implement these services and clients in any language. However, you are only provide a Java sample. If you are using a different language, you are responsible for developing the entire programs (including the echo services, web forms, travel service, etc.) for (a-c) and will not be able to use the sample startup code.
- Automate building, running and testing using scripts where possible. It will ease your life.
- There are many online resources. Use them wisely. Also use the mailing list to get doubts cleared. But do NOT post solutions or code there.
- *Start early. There are lots of "gotchas" when programming for and on the Cloud!* And, there may be additional issues when working from the IISc network. Plan to finish early.

4. Submissions Instructions

Submit a single file named **SE252:JAN2015:PROJ-0:$StudentName.tar.gz** containing all necessary source files required for running the demo. Also submit the **MD5 checksum** of the tar/gzipped file by email to 'simmhan@serc.iisc.in' with the subject 'SE252:JAN2015:PROJ-0:StudentName'. The total file size should be *<5MB*. You will have to demonstrate the project 0 by deploying from exactly this tar.gz file (hence the MD5 checksum) on three hosts which can have any standard web servers, libraries, tools, etc. of your choice.

Include a *README.txt* in the **zipped** submission with instructions on building and running all the project tasks. This should have enough information, including external software, servers, libraries and dependencies, to compile and run the project 0.

5. Deadline

Submit your project by email before midnight IST on Thu, Jan 22. Only a single submission will be accepted.