Practical 6

Theme

- adding a new collection resource to an existing JAX-RS service
- implementing a simple listener service to facilitate client notification

Key concepts: RESTful remote interface, Jersey libraries, JAX-RS resource annotations, proxy to RESTful service, notification services

6.1. Start up and essential configuration

a) *Log-in to Ubuntu and start a terminal.*

b) *Execute* `eeclipse.sh` *and locate projects* `lab5-server` *and* `lab5-client` *that you worked with last week.*

c) *Download and extract source code archives* `lab6-server.zip` *and* `lab6-client.zip` *within your* `CS3250-DS-1011` *folder.*

d) *Import the two extracted folders as two existing eclipse projects.*

e) *Perform steps 5.2(b), 5.2(h) and 5.2(i) from Practical 5 to project* `lab6-server`*.*
As a result, the project will become a Jersey-powered RESTful Web Service project.

f) *Deploy* `lab6-server` *on Tomcat, start Tomcat and ensure that* `lab6-client` *tests connect to* `lab6-server` *correctly.*

6.2. Implementing client notification — listener service.

a) *Perform steps 5.2(b), 5.2(h) and 5.2(i) from Practical 5 to project* `lab6-client`*.*
These service capabilities will be required on the client in order for the client to listen for server notifications.

b) *Add a class* `Listener` *to* `lab6-client` *project and annotate it as "messages" resource.*

c) *Add a POST method to the* `Listener` *class that simply prints the received chat message's sender, topic and content on the system output console (which will be the Tomcat's console).*

d) *Deploy the client project on the Tomcat server.*

e) *Write a new class* `NewMessageListenerProxy` *in the* `lab6-server` *project as a very simple proxy for a listener service with one operation* `newMessage`*.*
You can base this code on the chat server proxy in the `lab6-client` project.
Recall that the listener service has a single "messages" resource into which new chat messages should be posted by the chat server.

f) *Write a very simple class* `TestListener` *in* `lab6-server` *with a main method that sends a chat message to the deployed chat client using its listener service.*

g) *Test the program and fix any errors.*
You should see the chat message being printed by the listener on the Tomcat server log.

6.3. Implementing client notification — subscription service.

a) *Add a new class* `Subscriptions` *to the* `lab5-server` *project to represent a collection of subscriptions for some topic.*

When defining and using this class, you can use the fact that it is technically similar to the `Messages` class. Individual subscriptions will be identified by server-assigned numbers just like chat messages. Nevertheless, unlike chat messages, a subscription will not be exposed using an XML representation. Instead, a subscription will be represented by a simple plain text containing the URL of a client's listener service, to which the notifications will be sent by the chat server.

b) *In class* `Topics`*, add a locator method for the subscriptions sub-resource and adjust the inner class* `Topic` *to support this method.*

c) *Add a POST method in class* `Subscriptions` *and make it take a URL that is sent with the POST request and record it in a collection of subscriptions and correctly report back the URL of the created subscription.*

> **Hint.** *What type of collection should the subscriptions be held in?* Since subscriptions will be identified by integers, it may seem like a good idea to use a list as was done for the collection holding chat messages. Nevertheless, there is an important difference between chat messages and subscriptions: subscriptions get deleted whereas chat messages not. Keeping them in a list would not support deleting without losing an easy mapping between the items and their identifying numbers. It is therefore better to use a **map** from integers to the items. The map values could be the listener URLs or, much better, the proxies to the listeners.

d) *Add a DELETE method in class* `Subscriptions`*.*

Remember that you will need to use a *path parameter* to identify the subscription that should be deleted.

e) *Amend the method* `newMessage` *in class* `Messages` *so that all subscribers are notified of the message.*

You will need to add a new field in the `Messages` class referring to the appropriate instance of the `Subscriptions` class from which, after adding a new method, one will be able to get a collection of proxies to notify.

f) *Write a class* `TestSubscribe` *in project* `lab6-client` *with a main method that subscribes the current project's deployment URL to the server's subscriptions resource for topic "sport".*

You will also have to add a `subscribe` method to the `ChatServerProxy` class.

g) *Test the chat server-client interaction with notification and fix any errors.*

The following sequence should provide a reasonable test:

- restart the Tomcat server
- execute `TestSend`
- execute `TestSubscribe`
- execute `TestSend` again, observing the Tomcat server console for the listener reporting the "we won!" message.