Homework 3

Mattias Cederlund, mcede@kth.se

Server

The server is implemented using one resource on the path "/flightsReservationService" which contains all methods needed by the service. All application logic and storage is handled by a singleton class, FlightsDataService, just as in HW2.

The login is implemented as a GET request, using two query parameters, username and password. If validation succeeds it will return the secret token as plain text, otherwise it will give the unauthorized error as response. All other methods of the resource will take this token as a parameter of some sort and validate it and return the unauthorized error if it fails.

To get itineraries I used another GET request, providing token, departure city, destination city and date as path parameters. It will take the parameters, perform the itinerary finding algorithm and return a list of itineraries. Also a GET method to find a single itinerary was implemented and mainly used by the client application.

To book a ticket I used a POST request, passing the itinerary that should be booked (as application/xml), along with the secret token and a card number as path parameters. If booking succeeds it will return the ticket number for future reference.

To get a ticket (issue ticket) there is a GET method, getTicket, which takes the token and ticket number as query parameters and returns the full ticket.

To cope with the requirement of using all classes of requests I also added a method to change card number on a ticket. It's implemented as a PUT request, as it's editing an already existing entry with the new card number. The method requires the token, ticket number and the new card number. As PUT requests required an entry (not passed as parameters), I passed the new card number as an application/xml entry and the other parameters as path parameters. Although, I had the feeling that I probably should have created a xml-entity containing all the required information rather than using path parameters.

I also added a DELETE request, cancelBooking, which is used to remove a ticket entry from the storage. It takes the token and ticket number as path parameters.

Client

The client was implemented as a Web application, using the JSF framework. It has just one page, containing a simple interface to do requests to all of the methods in the REST resource.

To simplify the implementation, a REST client was auto-generated by Netbeans from the server project.

The JSF interface was implemented as a SessionScoped bean, because it needed to persist the secret token from the login request to be used by the other requests. Also the itineraries returned by the itinerary finding request is needed to make a booking.

To test the REST web service, first log in using a correct combination of username and password ("mattias" and "mattias). The token will be printed below the button if it succeeds.

After logging in you can use the other methods of the service. For example, try finding an itinerary from "Stockholm" to "London" on the "2015-02-07" (if unlucky try changing date in the range 2015-02-04 to 2014-02-08). The itinerary should be printed below the button if found.

To book a ticket for the last found itinerary, enter a card number and press the button. The ticket

number should be printed as a confirmation. Now it is also possible to see the full ticket by entering the ticket number just below.

To change card number enter the ticket number and the new card number. To confirm it worked, get the ticket again under “Issue ticket”.

To remove the booking enter the ticket number to cancel. To confirm it succeeded, try getting the ticket again by entering the ticket number under “Issue ticket”. It should no longer be able to find any ticket if everything works as expected.

Further explanation of the client JSF code is outside the scope of this course.

Running instructions

The projects are in Netbeans format (HW3Server.zip and HW3Client.zip) and can be imported in in Netbeans through File → Import project → From ZIP. After importing, first deploy HW3Server to an application server before deploying HW3Client. After deploying the client is available at http://localhost:8080/HW3Client/ .

The projects are also provided as war files (In /Executables, HW3Server.war and HW3Client.war) to be deployed to your favourite enterprise server.