Webservices – Homework 2

Jyoti Yadav jyoti@kth.se
Larry Poon, larryp@kth.se

Design and Implementation of Flight ticket reservation service: We designed this using bottom-up approach. When the project is complied and build, it creates a war file named **webservice-hw2.war** which can be deployed in webserver or application server. We used jboss application server. Once the application is deployed, the web services are published at the end point **http://localhost:8080/webservice-hw2/FlightBookingService?wsdl .** With the top-down approach, we first implemented the wsdl file and the corresponding schema document. We then publish the wsdl our web services using the same method we used for bottom-up approach. After that, java and classes files are generated by using **wsimport** tool. Finally, we developed two client systems to test our web services created by both approach.

1) **Authorization of customers**: We are maintaining a fixed combination of username and password in memory. So only those users and password combination can be used. The systems authentication against those in memory fixed set of username and password. After successful authentication it generates a random token and the web service returns the token in response. On server side the token is stored in memory and is used for authentication for further requests. If user authentication fails, authentication exception is thrown.
2) **Checking possible itinerary**: System creates some fixed set of flight data in memory. When a user request comes it searches direct flights and returns the response. If no direct flight is found, then it tries to search for combination flight and if found returns in the response.
3) **Checking availability of tickets**: same as above step but here it also checks if the seat is available for a given date. We maintain a fixed set of inventory and on each booking made we reduce the flight inventory by 1.
4) **Output the price of available itineraries**: the response of above requests also returns the price in the response
5) **Booking tickets for requested itinerary**. Once the user provides correct request we book a seat for the person, a ticket is created in memory and in the response a booking number is returned which can be used to later generate/issue the ticket.

SOAP Headers:
All soap requests except login request need a token to sent as soap request parameter. An example for searchItinerary is following:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.bookingservice.jyoti.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:searchItinerary>
      <token>TOKEN-1835344290</token>
       <departureCity>stockholm</departureCity>
        <destinationCity>berlin</destinationCity>
    </ws:searchItinerary>
  </soapenv:Body>
</soapenv:Envelope>
```

we can put the same token information in the SOAP header also

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.bookingservice.jyoti.com/">
  <soapenv:Header>
            <token>TOKEN-1835344290</token>
 </soapenv:Header>
  <soapenv:Body>
    <ws:searchItinerary>
       <departureCity>stockholm</departureCity>
       <destinationCity>berlin</destinationCity>
    </ws:searchItinerary>
  </soapenv:Body>
</soapenv:Envelope>
```

Instruction to run: All the source code is in Intellij maven project format. In order to run it we first need to import it in Intellij and run **mvn clean install**. After the build for server part it creates a war file **webservice-hw2.war** which can be deployed on server. It also creates a client jar which can be run to demo webservice client.