Programming Web-Services Homework 2

KTH Royal Institute of Technology
School of Information and Communication Technology
Student:Fanti Machmount Al Samisti (fmas@kth.se)
Student:Aruna Kumari Yedurupaka (akye@kth.se)

February 25, 2016

1 Introduction

This week we were tasked to implement a flight ticket reservation service built upon SOAP and web services. The testing of these services was done by the clients who parsed a *wsdl* file.

2 Task 1

The first step is to build a user login system as we don't want them to go on a rampage without any control. In order to accomplish this we compare the user's credentials against some hardcoded userbase. If the check succeeds then we return a token(a random number) otherwise a "not-valid" token.

Included with the report, we have the *flightReservation-doc.wsdl* used to generated the *authentication* web service.

```
wsimport -keep authorize.wsdl
```

The rest of the clients used the remote wsdl files:

```
wsimport -keep http://localhost:8080/soap/itinerary?WSDL
wsimport -keep http://localhost:8080/soap/book?WSDL
```

The available web services are(all verified through the passed auth token):

- Authorize a user(*/soap/auth/login*)
- Search flights given a source and a destination(*/soap/itinerary*)
- Search itinerary by id(*/soap/itinerary*)
- bookTicketsForItinenary(*/soap/book*)
- issueTickets(*/soap/book*)

3 Task 2

The test clients are the following:

- AuthClient
- ItineraryClient(Checks both operations)
- BookTicket

To deploy the *.war* file we run the following command:

```
mvn clean package
asadmin start-domain
asadmin deploy target/foo.war
```

The *SOAPHandler* can be used for logging and metrics analysis but we didn't implement it in our project.

4 Conclusion

In conclusion, the task at hand was enlightening in the sense that we went through every step of the web service implementation and deployment. The semantic freedom we had *SOAP* messaging, along with the factory and XML manipulation libraries, played a big role in easing this process.
The ways that we could attack this problem was either bottom-up(*wsdl*→*java*) or top-down(*java*→*wsdl*). The positive of the second approach is that the developer can write java which is easier readable and maintainable than a *wsdl* document. On the other hand writing *wsdl* you don't become bound to a specific platform/programming language thus remaining free to choose or ship with multiple technologies.