Homework 2 | **Kim Hammar**
ID2208 | **Mallu Goswami**
Programming Web-Services | Due Date: 7 February 2017

Problem Statement

This report covers the work done in an assignment on programming WebServices. The given task was to implement flight-ticket reservation services as web services. Further more both the top-down and bottom-up approaches to developing web services should be practiced.

Main problems and solutions

- *Bottom-up approach* - This task included using JAX-WS library [2] in java to implement webservices by first developing java classes and interfaces and then using `wsgen` tool to generate WSDL and XML Schema files.
- *Top-down approach* - Opposite development chain compared to bottom-up. Start by developing WSDL and XML schema files and then use the `wsimport` tool to generate Java classes and interfaces.

Implementation

Webservice is implemented using JAX-WS library and Document-style SOAP communication over the default SOAP-HTTP binding.

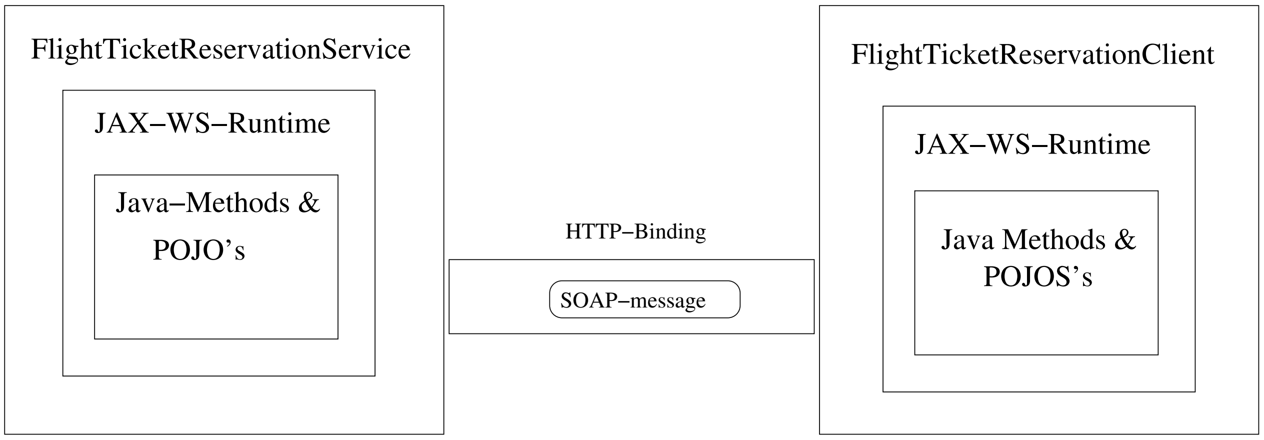

Figure 1: WebService Implementation with JAX-WS

SOAP

Using headers to add functionality to SOAP messages is known as vertical extension [1]. Perhaps the most canonical case to use SOAP headers to extend messages and protocols is to add security, since this is a typical requirement that might not be required at first deployment but as the service matures it might become a necessity. In our implementation of the flight-ticket reservation service we use a very ad-hoc authentication/authorization solution. First of all, credentials are sent in plaintext, and second the security token is sent as an *argument* to each invocation which clutters the interface and gives poor separation of concerns.

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:getItineraries xmlns:ns2="http://flight_reservation"
           >
            <arg0>Stockholm</arg0>
            <arg1>Mumbai</arg1>
            <arg2>ID2208_AUTH_TOKEN</arg2>
        </ns2:getItineraries>
    </S:Body>
</S:Envelope>
```

A better idea would be to employ a security solution utilizing SOAP intermediaries and WS-Security, however that is out of scope of this tutorial, but atleast we can achieve a better service-design by sending the security-token as part of the SOAP header instead of invocation-argument.

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header>
        <ns2:Token>ID2208_AUTH_TOKEN</ns2:Token>
    </S:Header
    <S:Body>
        <ns2:getItineraries xmlns:ns2="http://flight_reservation"
           >
            <arg0>Stockholm</arg0>
            <arg1>Mumbai</arg1>
        </ns2:getItineraries>
    </S:Body>
</S:Envelope>
```

There are many advantages with this approach:

- Separating the authentication from the method invocation means that we can later improve the security solution and change the API without having to change the interface of every operation.
- Having the authentication information in the header means that we could develop a solution where SOAP Intermedaries handles the authentication and the application don't have to be aware of it.

Conclusions

JAX-WS library allows you to develop web services in java without having to do much manual work with XML. However, to achieve good design for a web service you need still need a satisfactory understanding of the underlying technologies and formats. Two main approaches to developing web services nowadays when using java are top-down and bottom-up, which one to prefer depends on the situation. In general it requires more knowledge to do the top-down approach but it can also be more powerful

Attachments

Documented source code can be found in the attached zipfile. See README.MD in the root directory for instructions how to execute and build the program.

References

[1] Stephen Graham, Glen Daniels, Doug Davis, Yuichi Nakamura, Simeon Simeonov, Toufic Boubez, Ryo Neyama, Peter Brittenham, Paul Freemantle, and Dieter Koenig. *Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI (2Nd Edition).* Pearson Education, 2004.

[2] Oracle. Building web services with jax-ws. `http://docs.oracle.com/javaee/6/tutorial/doc/bnayl.html`, 2013. [Online; accessed 4-Feb-2017].