INFO-H-511: Web Services
TP 3 - Using & developping SOAP services

Lecturer: Stijn Vansummeren
Teaching Assistant: Francois Picalausa
http://cs.ulb.ac.be/public/teaching/infoh511

2011–2012

Part I: Consuming SOAP Services

In this first part, we will send messages to existing SOAP services. For this purpose, we use the `wsimport` tool[1] to translate wsdl service definitions into corresponding java code.

Exercise 3.1

In this exercise, we will use the DailyXmlFact service available at `http://www.xmlme.com/WSDailyXml.asmx?WSDL`.

- Using the `wsimport` tool, generate the caller code for the given service.
- Open the wsdl file in a browser, and compare it to the generated code. How are the supported methods defined? How are the input/output types translated?
- Write a program that uses the generated caller code to retrieve the daily fact.

Exercise 3.2

Amazon provides a SOAP API to find out the what books about Web Services are available. The skeleton code that is provided for this exercise interrogates Amazon's service to this end. However, this implementation is missing authentication [2] and therefore fails to execute. In this exercise, we will implement the necessary authentication mechanism.

- In the program entry point, add the `SOAPDebugger` to the resolver handlers of the service. Investigate the messages that are sent and received. What do they tell you about the execution?
- The `AwsHandler` aims to add the necessary authentication headers to outgoing SOAP messages. Implement the missing methods to include headers as specified in Amazon's documentation.

Part II: Designing SOAP Services

In the first part of this lab, we have explored ways of consuming SOAP services. We now turn on to implementing and designing SOAP services.

[1] `http://docs.oracle.com/javase/6/docs/technotes/tools/share/wsimport.html`
[2] `http://docs.amazonwebservices.com/AWSECommerceService/latest/DG/NotUsingWSSecurity.html`

Exercise 3.3

In this exercise, we will implement the Random service. This service will provide random quotes, and random numbers to its clients.

- The `Random` interface provides a description of the service. Add the necessary annotations to make it JAX-WS compatible. Create a new implementation of this interface. The utility class `Quotes` already contain a few quotes.
- Register your implementation of the `Random` service in the `ServiceServer` class.
- The `ServiceClient` contains the skeleton code to use your service from a client point of view. Test your service using this class.

Exercise 3.4

In the following exercises, we will implement a SOAP-based contact manager service. In this first part, you will define an interface that describes the service. To help you in this task consider the following scenarios and identify the methods that will be called.

- A new contact is created for "John Doe", with the email address `johndoe@company.org`.
- The last name of "John Smith" is changed to "Smithe"
- The "friends" group is rechristened the "party all night long" group.
- "Joe Bloggs" is removed from the contact manager.
- The "friends" group is rechristened the "party all night long" group.
- "Jane Doe" is removed from the "acquaintances" group and added to the "friends" group.

Exercise 3.5

Download the skeleton code provided on the labs web page. This code instantiates a Jetty server, with the necessary support for JAX-WS. The skeleton also implements a model for the contacts list.

Add an interface that corresponds to your service definition from Exercise 3.4. Also create an empty implementation of this interface and register it with the server.

Exercise 3.6

Based on the given contacts list model, provide an actual implementation for your contact manager service without support for contact groups. Test the implemented methods by adding to the client skeleton code.

Exercise 3.7

Add support for contact groups management to your contact manager service.