SCHOOL OF COMPUTER SCIENCE

PRIFYSGOL BANGOR UNIVERSITY

Java Technologies Exercises - Notes

Web Services

Dave Perkins

Introduction

In this document are some exercises to help you learn and investigate the use of SOAP(Simple Object Access Protocol) and REST(Representational State Transfer) based web services.

For the purposes of these laboratory exercises the best place to begin is by reading Chapter 31 in Deitel and Deitel's standard text book *Java™: How to Program* (9th edition). One legitimate way of accessing this material via the internet is to use **Safari Books Online** with a ten-day free trial subscription.

Other on-line tutorial support can be found at the following site:

http://netbeans.org/kb/docs/websvc/jax-ws.html

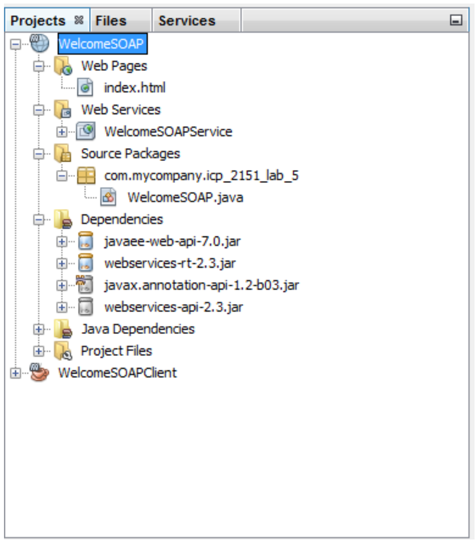
The **See Also** section on this site has a list of other useful tutorials.

The w3schools site has tutorials on the following topics all of which relate to the use of web services.

- SOAP
  http://www.w3schools.com/soap/default.asp
- JSON
  http://www.w3schools.com/json/default.asp
- Web Services in general
  http://www.w3schools.com/webservices/default.asp
- WSDL
  http://www.w3schools.com/wsdl/default.asp

Exercise 1: Defining a WelcomeSOAP Service

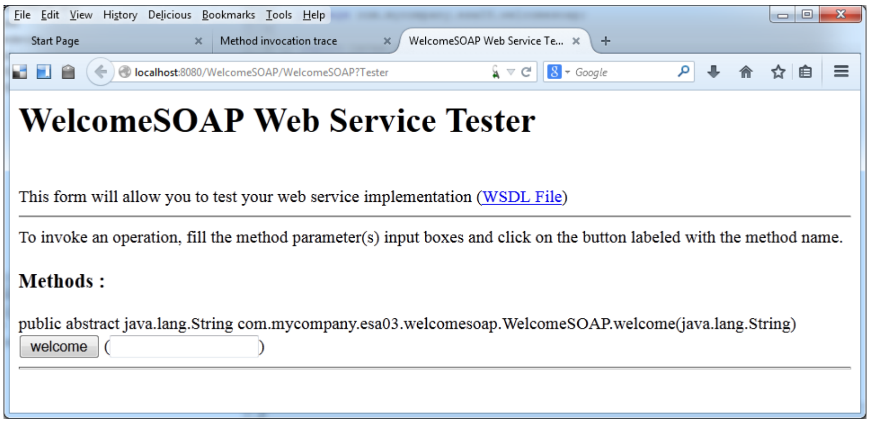
Create a SOAP web service to display a simple parameterised welcome message. The code to do this should be placed in a file called **WelcomeSOAP.java** which is stored in the **Source Packages** node. As a first step create a Maven project to build a Web Application called **WelcomeSOAP**. The figure below indicates what the project structure should look like:



```
WelcomeSOAP
  Web Pages
    index.html
  Web Services
    WelcomeSOAPService
  Source Packages
    com.mycompany.icp_2151_lab_5
      WelcomeSOAP.java
  Dependencies
    javaee-web-api-7.0.jar
    webservices-rt-2.3.jar
    javax.annotation-api-1.2-b03.jar
    webservices-api-2.3.jar
  Java Dependencies
  Project Files
WelcomeSOAPClient
```

Figure 1 Project Structure

Using Glassfish this simple web service can be tested independently of a client. To perform test generate the web page below with the aid of the web service test facility. To initiate the test open the **Web Services** node and right-click on the file **WelcomeSOAP.java**. Then select the **Test Web Service** menu item. The test page below should then be generated by the Glassfish application server.


File Edit View History Delicious Bookmarks Tools Help
Start Page
Method invocation trace
WelcomeSOAP Web Service Te...
localhost:8080/WelcomeSOAP/WelcomeSOAP?Tester
Google

WelcomeSOAP Web Service Tester

This form will allow you to test your web service implementation (WSDL File)

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

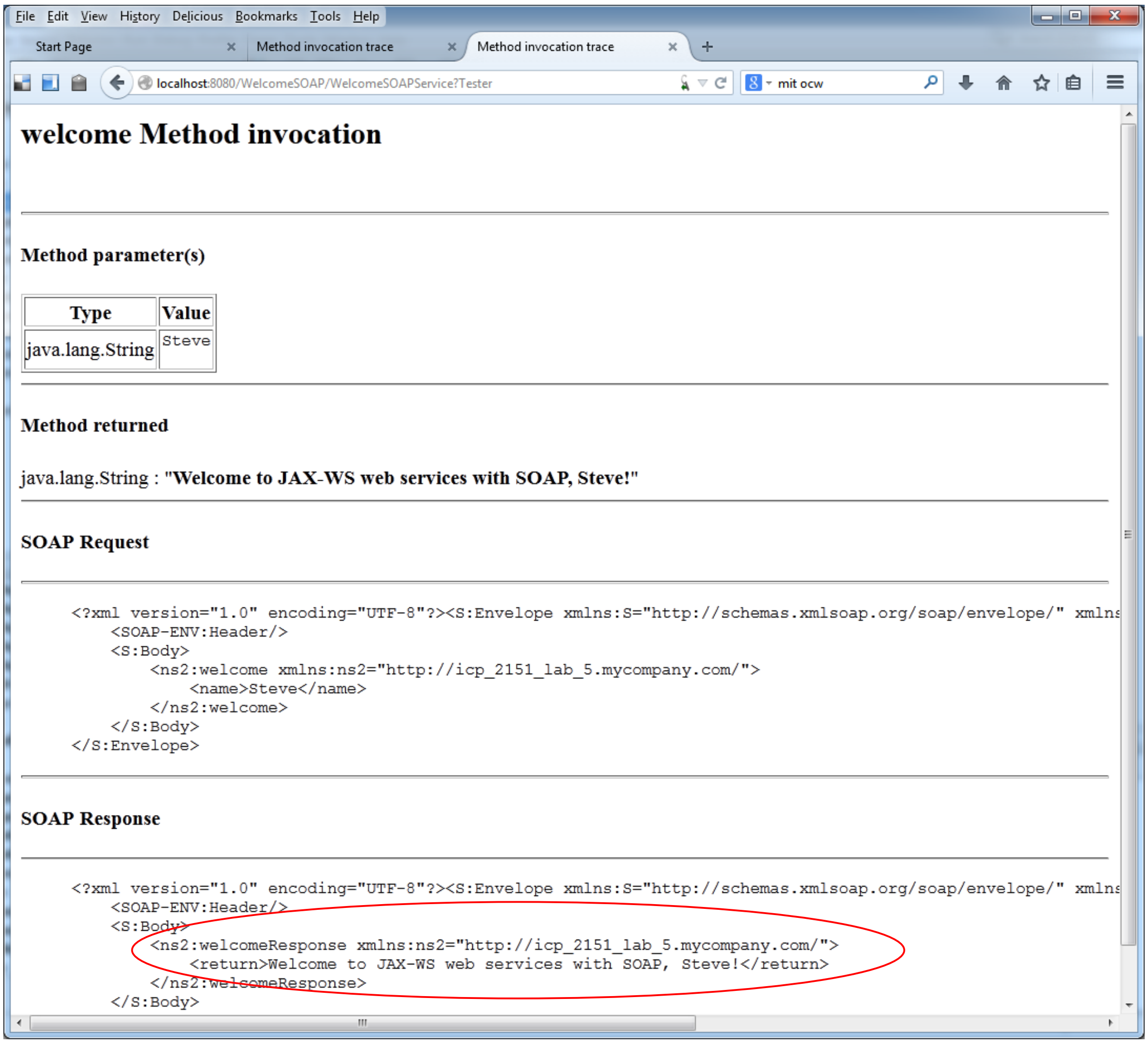
public abstract java.lang.String com.mycompany.esa03.welcomesoap.WelcomeSOAP.welcome(java.lang.String)

welcome ( )

Figure 2 Testing the HelloWorld Web Service

If everything is working correctly a SOAP Request and a SOAP Response document is then produced and displayed by clicking the button labelled **Welcome**. Make sure you supply your name as a parameter.

The result of clicking the Welcome button is shown below. You should get something similar. Study the display carefully and try to understand the contents of the SOAP Request and SOAP Response components.


File Edit View History Delicious Bookmarks Tools Help
Start Page
Method invocation trace
Method invocation trace
localhost:8080/WelcomeSOAP/WelcomeSOAPService?Tester
mit ocw

welcome Method invocation

Method parameter(s)

Type	Value
java.lang.String	Steve

Method returned

java.lang.String : **"Welcome to JAX-WS web services with SOAP, Steve!"**

SOAP Request

```
    <?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns
        <SOAP-ENV:Header/>
        <S:Body>
            <ns2:welcome xmlns:ns2="http://icp_2151_lab_5.mycompany.com/">
                <name>Steve</name>
            </ns2:welcome>
        </S:Body>
    </S:Envelope>
```

SOAP Response

```
    <?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns
        <SOAP-ENV:Header/>
        <S:Body>
            <ns2:welcomeResponse xmlns:ns2="http://icp_2151_lab_5.mycompany.com/">
                <return>Welcome to JAX-WS web services with SOAP, Steve!</return>
            </ns2:welcomeResponse>
        </S:Body>
```

Figure 3 SOAP Response and Request Envelopes

These two documents are structured using XML and contain data passed from the client to the server and from the server to the client. Notice, in particular, that the SOAP Response document contains a return tag used to store the parameterised welcome message that the service returns to a client.

To get started on this exercise you are *strongly recommended* to read the previously cited reference from Deitel and Deitel's book *Java: How to Program*.

Exercise 2: Consuming the SOAP Service

Having defined the service you should now create another Maven project to hold a client to consume the service. The project should be called WelcomeSOAPCent. In the first instance the client should be created as a **Java Application**. Do not create the project with a main class: this will be provided later by a subclass of `JFrame` created using the GUI builder tool.

Once the application has been created add a *web service reference* to it. If no reference is provided the client cannot access the service provided. To establish a reference add a web service client to HelloWorldSOAPClient. The first step is to right click on the project name and then select **New > Web Service Client** this will generate the following dialog:

Figure 3 Web Service Client Dialog

Provide an appropriate URL for the WSDL URL field. To save typing and errors copy the URL from the output window of Glassfish. In the example above the URL entered is

http://localhost:8080/ICP_2151_Lab_5/WelcomeSOAPService?WSDL

If this is done successfully a number of new files are generated in the client application. Notice in particular the contents of the two folders:

- **Generated Sources** – contains client side artefacts;
- **Web Service References** – contains web service endpoint.

Check out the contents of the **Web Service References** folder – you should discover a service endpoint interface (SEI) labelled **WelcomeSOAPService**. See Figure 4 on next page.

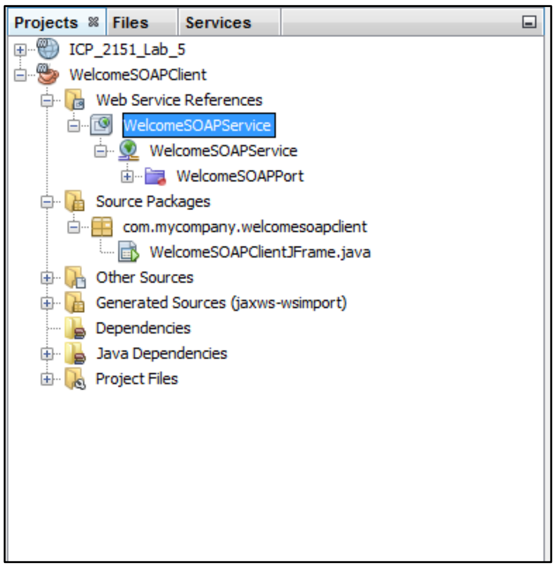


Figure 4 Adding a Web Service Reference

The client application should now be provided with a **JFrame** containing the three Swing components illustrated below:

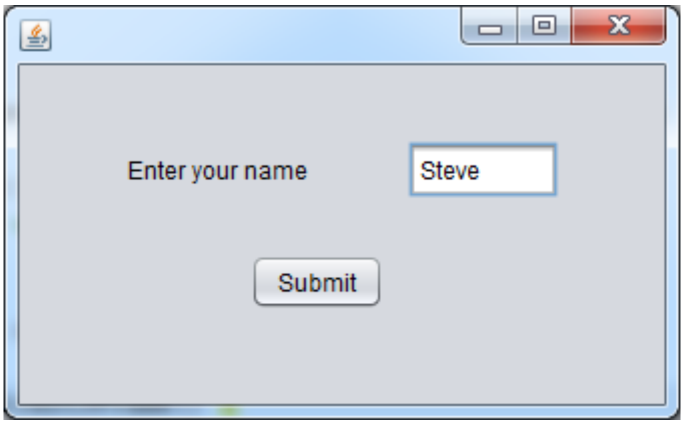
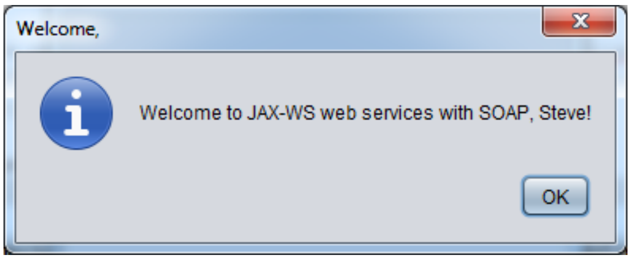


Figure 5 JFrame with Label and Button

Note that for the purposes of this laboratory session you are permitted to use a GUI builder tool. When the **`JButton`** labelled **Click** is clicked with the mouse pointer the web service called **`WelcomeSOAP`** should be invoked with the following results.



Welcome to JAX-WS web services with SOAP, Steve!

Figure 6 Service Response

If you get this response, congratulations are in order! You have just consumed your first web service?

Exercise 3: The Pig Web Service

PIg is a well known two player jeopardy style dice game, and is widely used in computing and mathematics courses to teach concepts of probability. The rules for the game of Pig are given below:

Players

The game has two players, one of which is human whilst the other is implemented by a program logic which you are required to develop. In effect, you are being asked to develop a limited form of gaming AI to create a virtual player (which I shall call Server). The game playing strategy you implement is entirely your decision but must obviously be consistent with the rules.

Equipment

The game is played with a single six-sided die.

Goal

Be the first player to reach 100 points.

Game Play

On a turn, a player rolls the die repeatedly until either:

- A one is rolled
- The player chooses to hold (stop rolling)

If a one is rolled, that player's turn ends and no points are earned.

If the player chooses to hold, all of the points rolled during that turn are added to his or her score.

Scoring Examples

Example 1: Sherri rolls a 3 and decides to continue. She then chooses to roll seven more times (6, 6, 6, 4, 5, 6, 1). Because she rolled a 1, Sherri's turn ends and she earns 0 points.

Example 2: Server rolls a 6 and decides to continue. Server then chooses to roll four more times (3, 4, 2, 6) and decides to hold. Server earns 21 points for this turn (6+3+4+2+6=21).

Game End

When a player reaches a total of one hundred or more points, the game ends and that player is the winner.

Your task is to code this game. For this exercise you are required to develop a SOAP web service which enables clients to play Pig using the Internet. To get started on this project you should study Deitel and Deitel’s example of a Blackjack web service[1].

[1] See pp. 1360-76, *Java: How to Program* (9th edition)

In developing your web service you should pay particular attention to the following points:

- Each game involves *only* two players – the client and the computer
- The client should be provided with a GUI

For the purposes of this exercise it is permissible to use a GUI builder tool to develop the interface. A sample interface is provided below. Note that this is intended for guidance only – feel free to improve on the design.

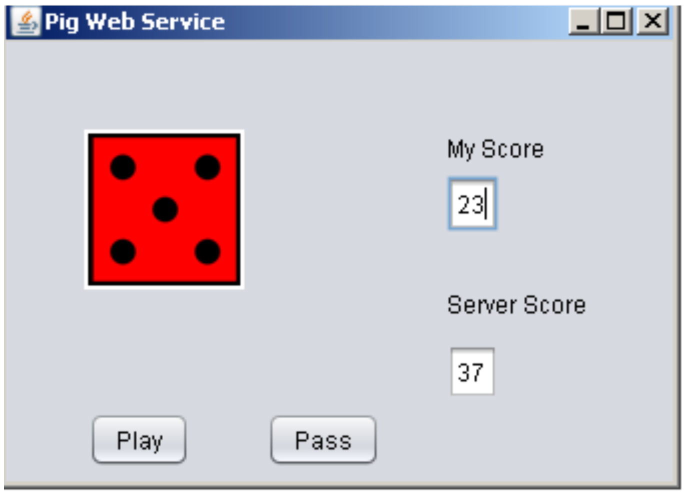


Figure 7 Pig Web Service

The interface below allows the player to roll the dice by clicking the Play button or to pass the dice back to the Server player by clicking the Pass button. At the end of each turn the relevant score is updated. When a player reaches the winning score a `JOptionPane` should be used to diplay an appropriate message.