Java Web Services

Exercise book

Version 1.0

© Copyright 2003-2004 InterBit LTD.

Ex1-HelloWorld

HelloWorld Web service and clients demonstartion

Exercise description:

This first exercise demonstrates the creation of a simple web service including a single method:

```
String sayHello(String name)
```

and running different clients to test it.

This exercise will be completed with the instructor.

Exercise steps:

1. Browse through the web service code (HelloWorld.java) and understand it.
2. Deploy the web service using the Ant build script.
3. Browse through the static stub client code (HelloClient.java) & DII client (HelloClient2.java).
4. Run the clients and watch the result.
5. Use a web browser to watch the WSDL created for the web service.

Additional steps:

1. Us the VBScript client provided (hello.vbs) to test the web service.

© Copyright 2003-2004 InterBit LTD.

Ex2-Clients

Web service Java clients

Exercise description:

In this exercise you will write 2 web service JAX-RPC clients:

1. A static stub client
2. A DII client

The clients will be used to access a Weather web service which returns the current weather by getting the day of week as a parameter.

Exercise steps:

1. Browse through the web service code (WeatherService.java) and understand it.
2. Deploy the web service using the Ant build script.
3. Write a static stub client, remember that before writing the code, a stub must be created using the Ant script.
4. Test the client, get the weather.
5. Write a DII client, no stub is needed.
6. Test the client, get the weather.

© Copyright 2003-2004 InterBit LTD.

Ex3-Web Services

RPC and Message Web Services

Exercise description:

In this exercise, two kinds of web services will be written and deployed:

1. An RPC-based web service with a user-defined parameter type (JavaBean)
2. A message-based web service (advanced)

To test the services, two clients will be used.

Exercise steps – RPC:

1. Write the RPC-based AddressService. This service will hold a list of addresses attached to user names and will have two methods:
   ```
   void addAddress(String userName, AddressBean address)
   AddressBean getAddress(String userName)
   ```
   This service must be stateful.
2. Deploy the service using Ant. Check the WSDL, see the new <types> element.
3. Write the client, first create the stub. Check the classes that were created and look for an additional class representing the AddressBean.
4. Add some users and addresses.

Exercise steps – message (Advanced):

1. Write the Message-based MessageService. This service will get an XML message sent to it and return another XML using a single method:
   ```
   Element[] echoElements(Element[] elems)
   ```
   The message to send is provided in the file message.xml, check it out.
   The result will look like this:
   ```
   <result>
       Message sent successfully to {destination here}.
   </result>
   ```
2. Deploy the service using Ant.
3. Write the client as a DII web service client.
4. Check the web service.

© Copyright 2003-2004 InterBit LTD.

Ex4-SAAJ

Writing a SOAP client with SAAJ

Exercise description:

It this exercise, another client will be written for the WeatherService. This time, the client will be written with the SAAJ API, i.e., you will have to construct the SOAP request, send it and parse the SOAP response.
Remember that SAAJ is similar to DOM APIs, you must be familiar with the structure of a SOAP message to successfully complete this exercise.

Exercise steps:

1. We use the pre-deployed WeatherService, so no new service should be deployed.
2. Write the SAAJClient.
3. Run the client and see that you get the same response as before.

© Copyright 2003-2004 InterBit LTD.

Ex5-JAXR

Working with a UDDI registry from Java

Exercise description:

In this exercise we will connect to a public UDDI registry and query it for all the organizations with the 'ISR' letters as part of their name. We will then print the information about those organizations to the output console.

To connect to the registries, use the following URLs:
Microsoft UDDI: http://uddi.microsoft.com/inquire
IBM UDDI: http://uddi.ibm.com/ubr/inquiryapi
SAP UDDI: http://uddi.sap.com/UDDI/api/inquiry/

You can use only one of them, since they are synchronized every night!

If you are behind a proxy server, don't forget to set the host and port.

Exercise steps:

1. Write the JAXR client. Use '%ISR%' as the organization's pattern.
2. Test it with your favorite UDDI.
3. You should get some organizations (3 on the last count…).

© Copyright 2003-2004 InterBit LTD.

Ex6-JAXB

Object to XML mapping

Exercise description:

In this exercise we use the JAXB mechanism to unmarshal XML to Java objects, change the objects, validate the data and marshal it into a new XML document.
The JAXR APIs give you a great mechanism to work with MXL files instead of using the standard DOM parsing techniques or any other mechanism.

To work with JAXB, we need to use the JAXB compiler – xjc.
We will run it using an Ant build script.

Exercise steps:

1. An XML file (message.xml) and an XML schema file (message.xsd) are provided. Browse them and get to know them.
2. Run the xjc compiler using Ant. Check out the newly created classes.
3. Write the Main class, the following steps are needed:
    a. Unmarshal the message.xml file.
    b. Print the content of the message.
    c. Change the subject to null. This is invalid according to the XML schema!
    d. Validate the message object. You will get an exception.
    e. Change the subject to something new.
    f. Marshal the object into a new XML file.
4. Browse the new file, see that the subject has changed.

© Copyright 2003-2004 InterBit LTD.

Ex7-Security

Authentication and Authorization with AXIS

Exercise description:

In this exercise we will add some security aspects to a Bank web service. The purpose is to authenticate a user before he accesses the service and then also check that he is authorized.

Here, we use a very simple text based user repository (users.lst) and two pre-existing AXIS handlers:

```
org.apache.axis.handlers.SimpleAuthenticationHandler
org.apache.axis.handlers.SimpleAuthorizationHandler
```

Exercise steps:

1. Define the username/password pair. Find the file named users.lst in the AXIS WEB-INF directory and add a new user and password.
2. Write the bank service. The bank service has a single method:
   `String getAccountBalance()`
   which returns the username and his account balance.
   To get the current username, use the `MessageContext` class.
3. Add the security handlers to the axis deployment file (deploy.wsdd), and deploy the service using Ant.
4. Write the Bank client as a DII client. We need to specify the username and password for the Call. Use the following methods:
   `Call.setProperty(String properyName,String value)`
5. Check the client with a proper user and with an unauthorized user. See what happens with an invalid user.

© Copyright 2003-2004 InterBit LTD.