

Installing Middle-ware: Apache Axis

Web Languages Course 2009
University of Trento

Lab Objective

- Install Tomcat 5.5 and Axis 1.4 in Windows and Linux
- Getting started with a simple example
 - Direct deployment of a Java Web Service (jws)
 - Test the service with a simple client



Lab Outline

- Ant Installation
- Tomcat Installation
- Web Service Middleware: an overview
- Axis Installation
- Deploying a Java Web Service (JWS)
- Running a Client (DII method)



Apache ANT

- Software tool for automating software build processes
- Java based build tool.
- Open source
- Full portability of pure java code.
- Downloads available at <http://ant.apache.org>



Tomcat Installation

1

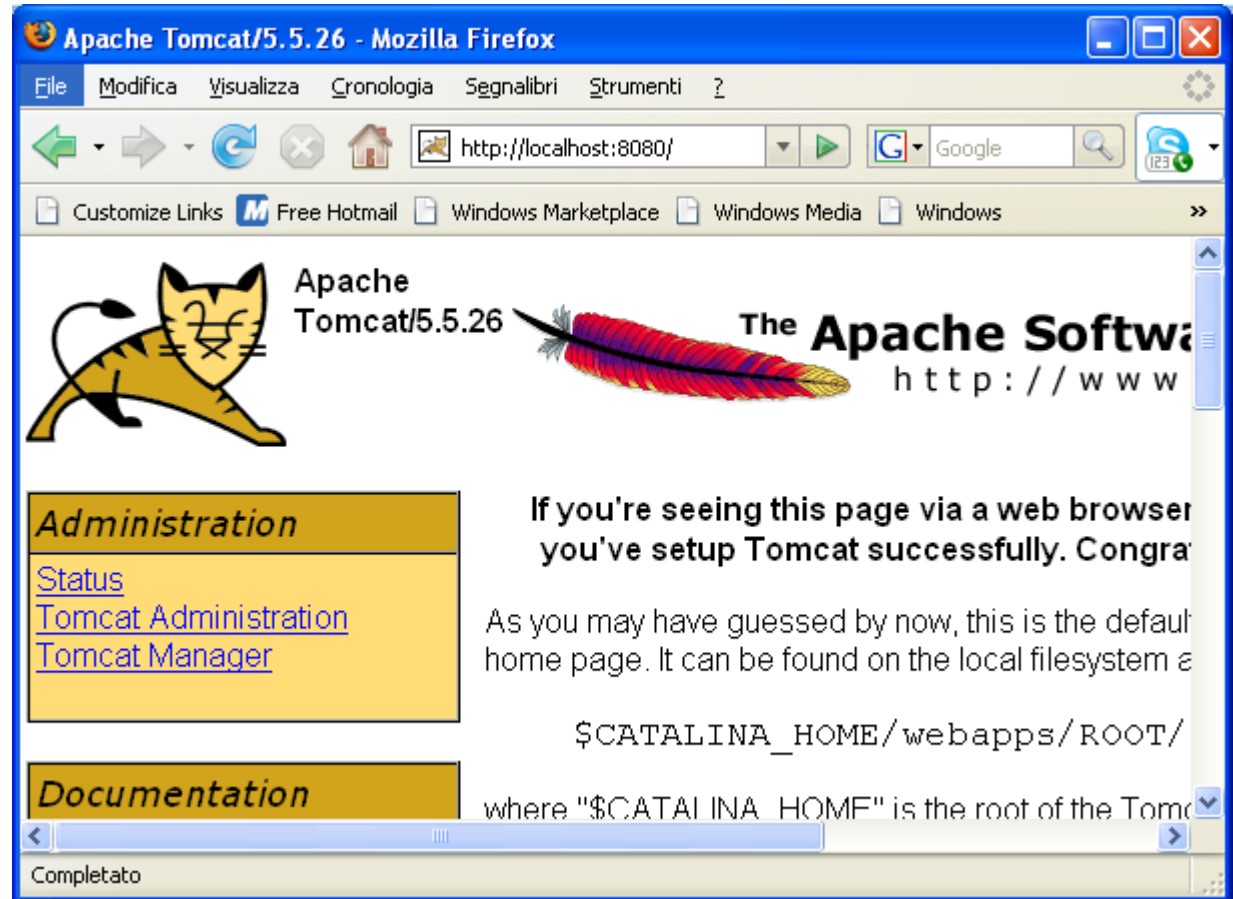
- Go to: <http://jakarta.apache.org/tomcat/>
- For Windows:
 - download the zip version of Tomcat 5.5.
 - Unzip it somewhere.
 - Start the server (run “C:\tomcat_home\bin\startup”)
 - Go to <http://localhost:8080/>
 - Set environment variables (CATALINA_HOME, TOMCAT_LIB, TOMCAT_CP)



Tomcat Installation

2

You should see something like this



Tomcat Installation

3

- For Linux: read the Installation Guide provided in the course website

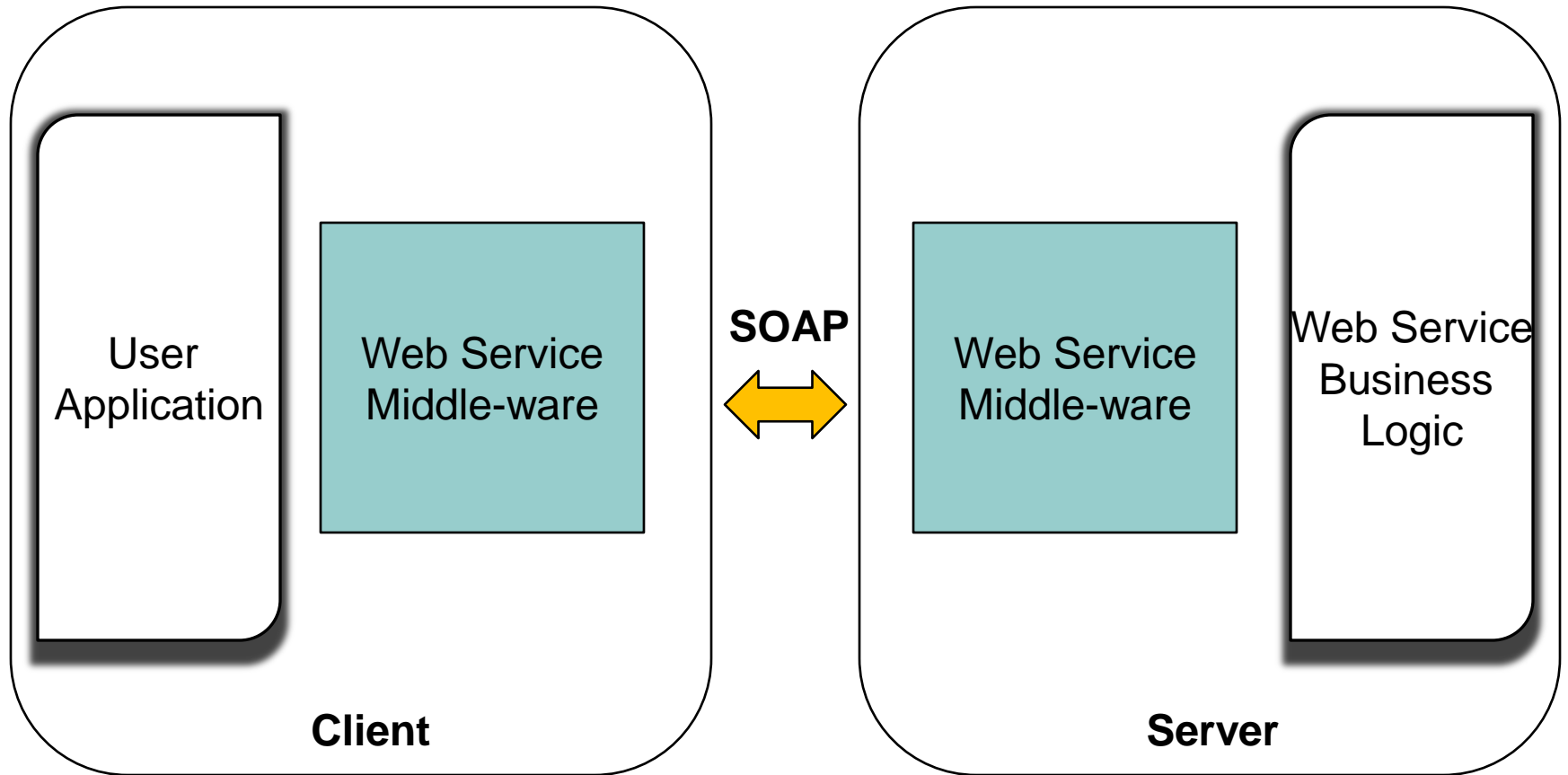


Web Service Middle-ware

An overview on Axis
Functionalities

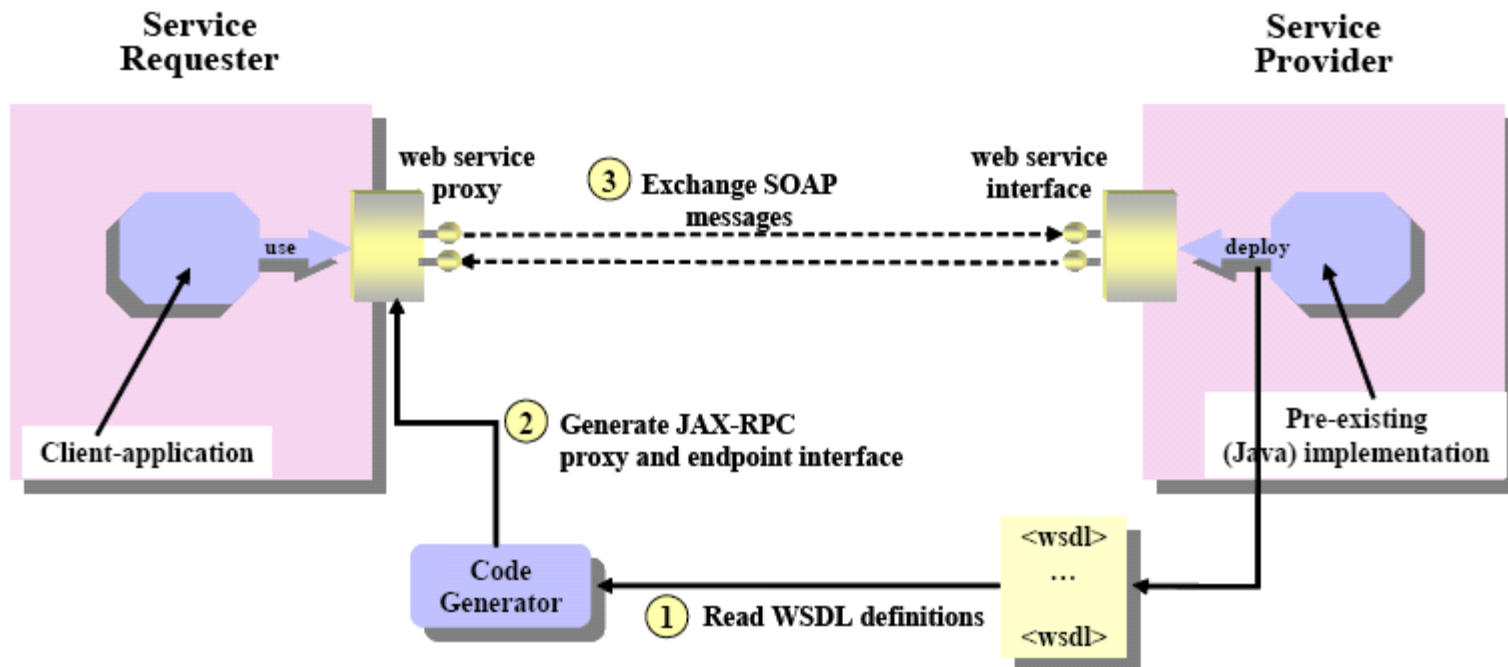
Web Service Middle-ware

1

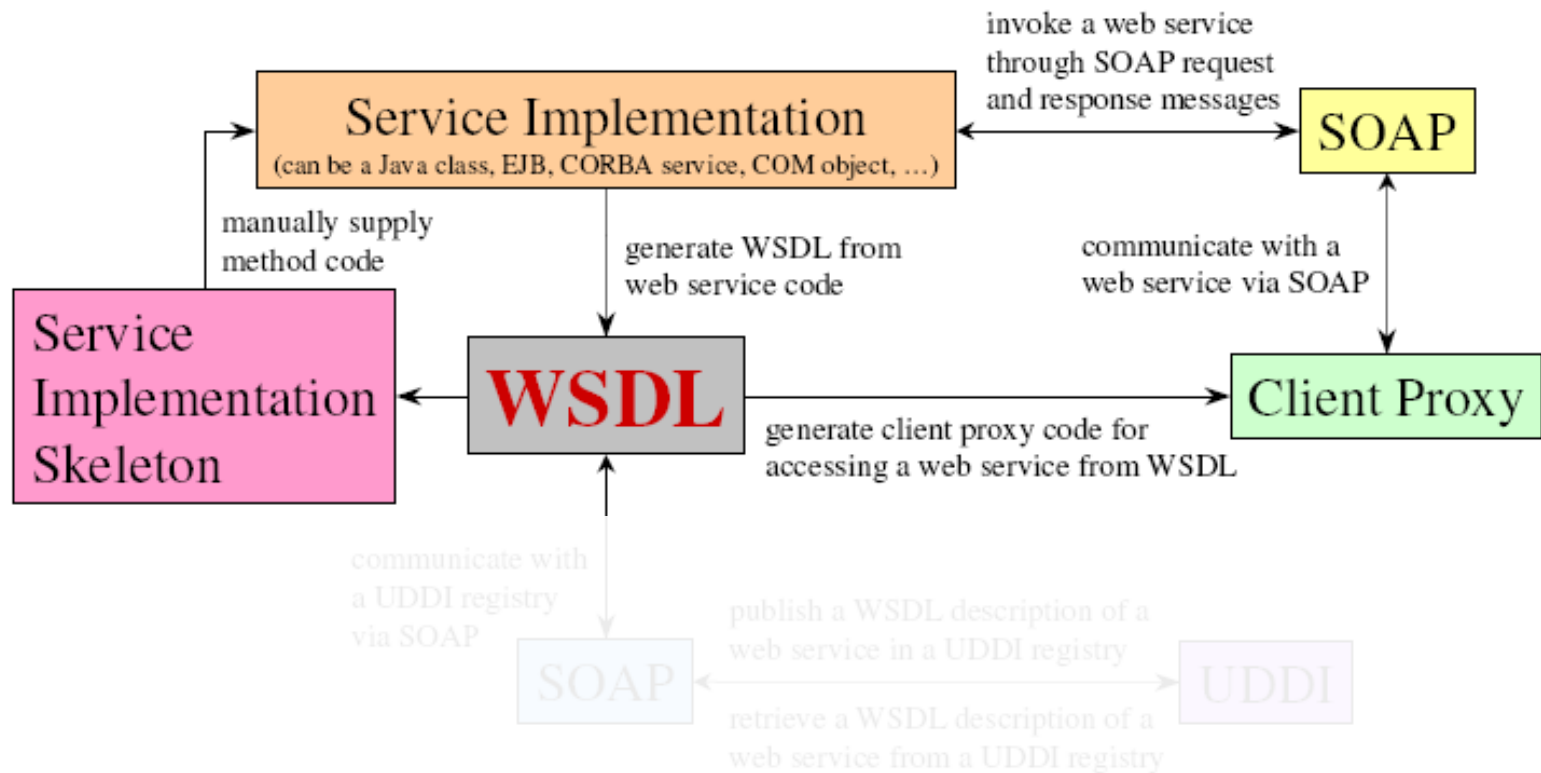


Web Service Middle-ware

2



General Web Service Toolkit Functionality



Apache AXIS

- A SOAP Processing Engine
 - Flexible and extensible architecture
 - Tools, Examples, Documentation, ...
 - A great place to learn about Web Services !!
- Open-source, hosted by Apache Software Foundation
- Packages for Java and C++



Axis Installation

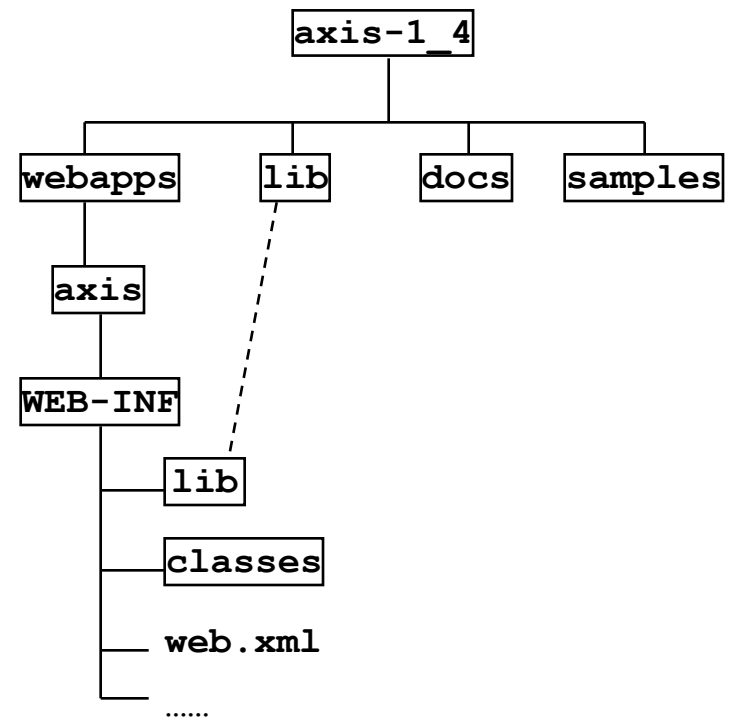
- Download version 1.4 from <http://ws.apache.org/axis/>.
- For Linux: read the Installation Guide provided in the course website
- For Windows: download the binary zip version. Unzip it somewhere.



Deploy Axis

- Copy `webapps\axis` tree to `webapps` directory of Tomcat.
- Run, or restart, Tomcat.

Directory Structure:

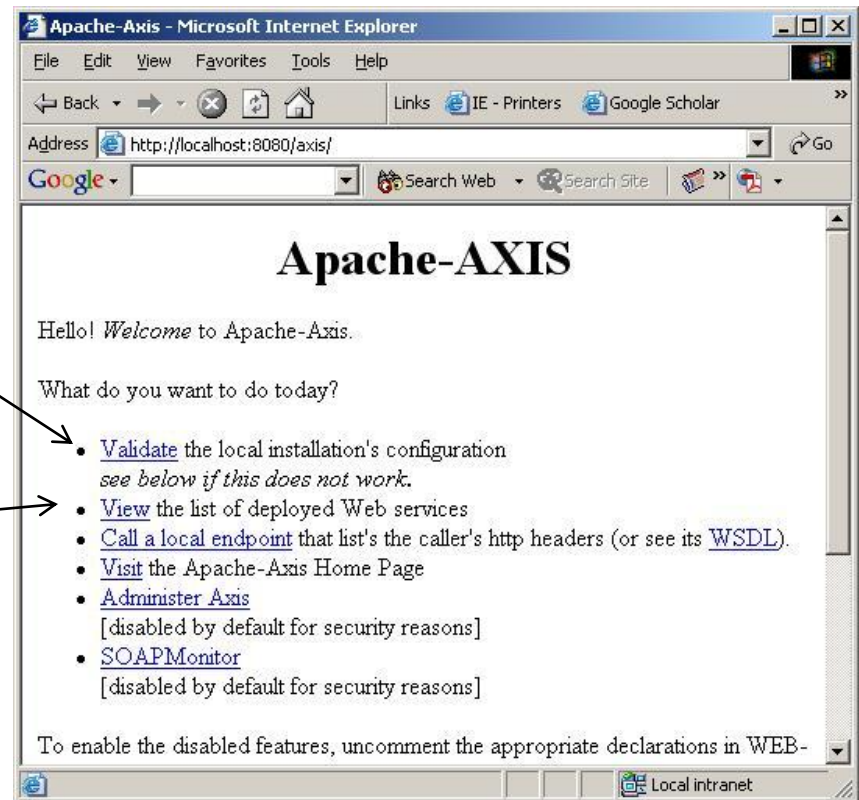


Test the Deployment



- Point your browser to <http://localhost:8080/axis>, you should see something like this:

Click on “Validate” to check the installation

Click on “view” to see all the current deployed web services



How to create a Web Service

- There are two basic methods:
 1. Implement the service (write a Java class) and produce the WSDL:

 2. Create the WSDL and produce the Java class




How to produce a WSDL

- There are three basic methods:

1. Instant Deployment (JWS)

- Generates a WSDL from an URL
- For simple services



2. WSDD Deployment

- Generates a WSDL from an URL
- For wsdd deployed services

3. Use Java2WSDL tool

- Generates WSDL from Java service implementation classes

Deploying Web Services: JWS

1

- Steps

- If not there already, copy “axis” directory in “webapps” directory of the Axis distribution to the deployment directory of a server that supports Java servlets
 - For Tomcat, this is the “webapps” directory
- Copy any Java source file that implements a web service into this “axis” directory
 - No special code is required
 - All public, non-static methods are exposed



Deploying Web Services: JWS

2

- Change the file extension from “.java” to “.jws”
- To view the WSDL of a JWS web service, enter the following URL in a web browser

<http://host:port/axis/file-name.jws?wsdl>

JWSHandler,
JWSProcessor
RPCProvider
classes
do the job

- A file “.class” is created under the dir
“...\webapps\axis\WEB-INF\jwsClasses”
 - If the class is in a package, the appropriate subdirectory is created under “...\jwsClasses”



Example JWS Web Service

- Create a Java class using this code:

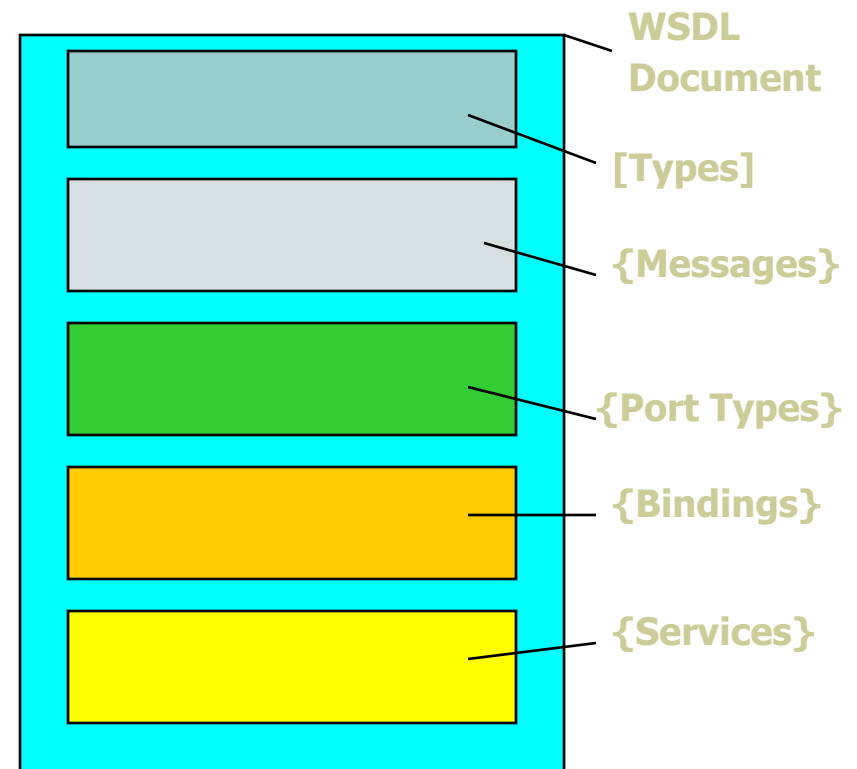
```
public class AddFunction {  
    public int addInt(int a, int b){  
        return (a+b);  
    }  
}
```

- Name the file `AddFunction.jws`. Notice the filename extension – it is `.jws` (for Java Web Service). Make sure the name of the file is identical to the name of the Java class.
- Deploy it by copying the file to `webapps/axis` directory.
- You are done!!



WSDL Refresh

- A WSDL document describes
 - *What* the service can do
 - *Where* it resides
 - *How* to invoke it



The WSDL File

- Examine its WSDL description. Point your browser to <http://localhost:8080/axis/AddFunction.jws?wsdl>

```
- <wsdl:message name="addIntResponse">
  <wsdl:part name="addIntReturn" type="xsd:int" />
</wsdl:message>
<wsdl:message name="addIntRequest">
  <wsdl:part name="a" type="xsd:int" />
  <wsdl:part name="b" type="xsd:int" />
</wsdl:message>
<wsdl:portType name="AddFunction">
  <wsdl:operation name="addInt" parameterOrder="a b">
    <wsdl:input message="impl:addIntRequest" name="addIntRequest" />
    <wsdl:output message="impl:addIntResponse" name="addIntResponse" />
  </wsdl:operation>
</wsdl:portType>
```



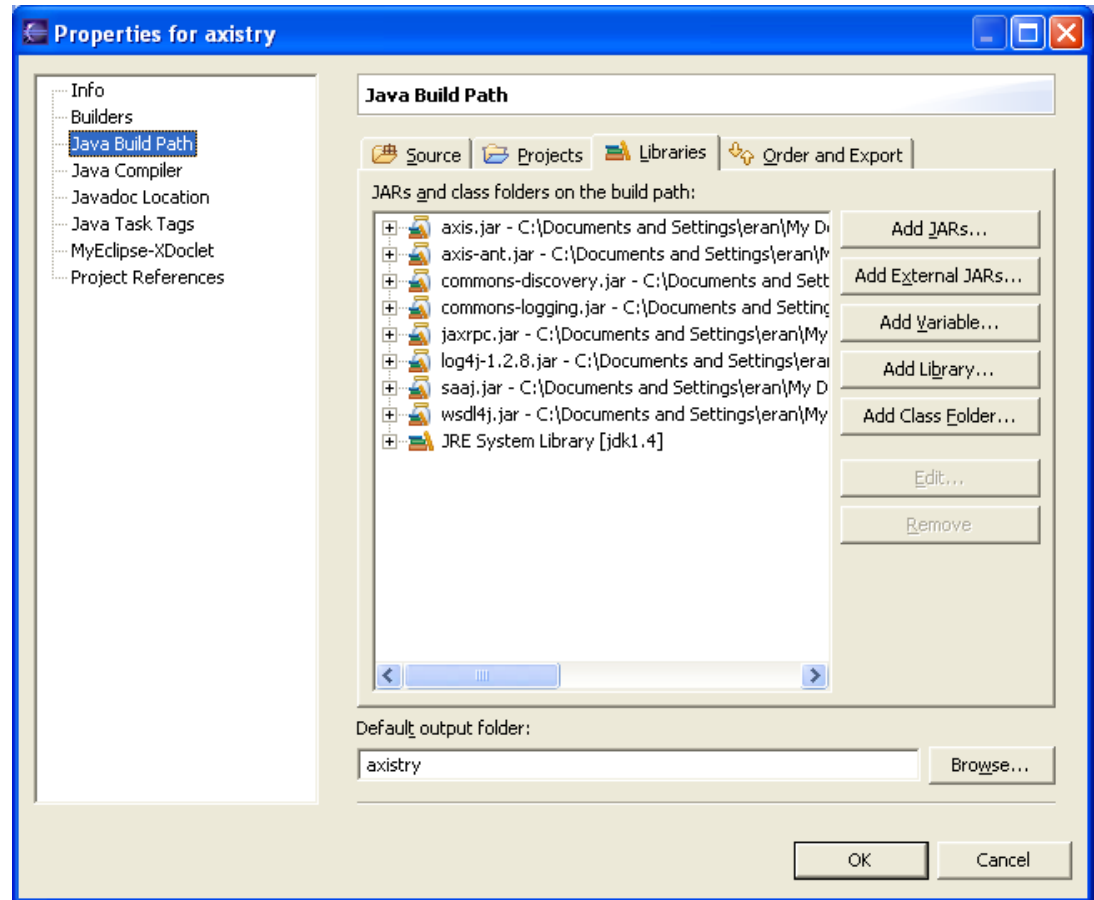
Invoking JWS Services

- Two Options:
 - Using Dynamic Invocation Interface (DII)
 - Using Stubs (WS Proxy) generated from Service WSDL description (**next lecture...**)



Simple Client Test – using DII with Eclipse

- Create a new Eclipse Java Project
- Add all the jars under “...\\webapps\\axis\\WEB-INF\\lib” to the java build path libraries (using external jars)
- Create a new class, called “TestAddFunction”



Client Code – DII approach

```
import org.apache.axis.client.Service;
import org.apache.axis.client.Call;
import javax.xml.namespace.QName;

public class TestAddFunction {
    public static void main(String[] args) {
        try {
            String endpoint = "http://localhost:8080/axis/AddFunction.jws";
            Service service = new Service();
            Call call = (Call) service.createCall();
            call.setOperationName(new QName(endpoint, "addInt"));
            call.setTargetEndpointAddress(new java.net.URL(endpoint));
            Integer ret = (Integer) call.invoke(new Object[] { new
Integer(5), new Integer(6) });
            System.out.println("addInt(5, 6) = " + ret);
        } catch (Exception e) {
            System.err.println("Execution failed. Exception: " + e);
        }
    }
}
```

