



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Εισαγωγή στα Δίκτυα Υπηρεσιών

**Assisting Lecture 9 - Top Down SOAP Web
Services**

Μύρων Παπαδάκης
Τμήμα Επιστήμης Υπολογιστών

Introduction to Service Networks

CS-592 – Spring 2015

Assisting Lecture: Top- Down WS with Apache
Tomcat, Axis2, Eclipse

Myron Papadakis (myrpap@gmail.com)

Outline

- Axis2, Tomcat, Eclipse
- Top-Down Web Services
 - Area rectangle calculator
 - Area rectangle calculator client
- Bottom-up Web Services
- SoapUI



Axis2

- Axis2 is a **SOAP processing engine** with the main function of delivering incoming SOAP messages into correct applications (every application is assumed to be a Web Service)
- Generate code from both service clients (=stubs) and service providers (=skeletons)
 - **WSDL2Java** tool: code generation from WSDL
 - **Java2WSDL**: generate WSDL from code
- *Data binding* is the term used for techniques that handle the conversion between XML and application data structures
- Axis2 provides a variety of data binding frameworks
 - ADB (Axis2 Data Binding): Limitations only for Axis2 Web services
 - XMLBeans
 - JiBX
 - None (AXIOM)
 - AXIOM is Axis2 document model used for XML Message handling
- More information at:
<http://www.ibm.com/developerworks/webservices/library/ws-java3/>

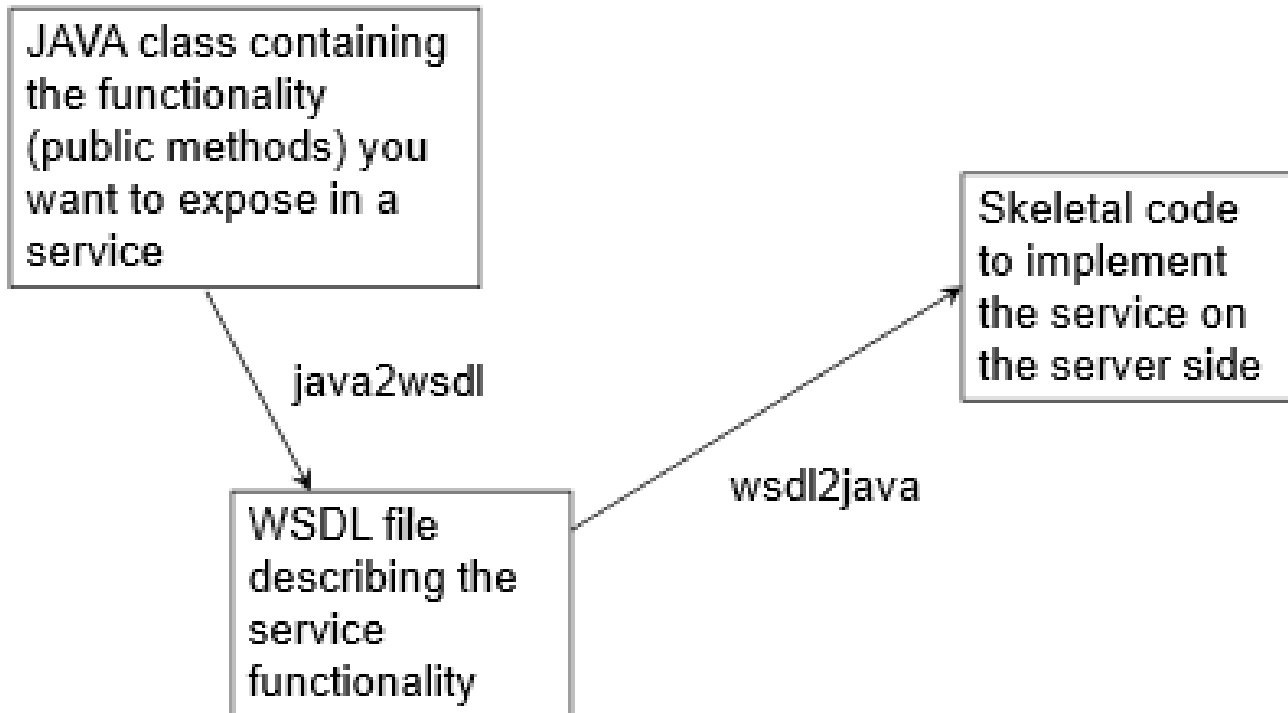
Axis2 Data Binding

- The primary objective of any data binding framework is to map the XML schema constructs to programming language objects,
 - provide the marshaling (i.e., produce an equivalent XML document from a programming language object structure) and
 - unmarshaling (i.e., produce an equivalent programming language object structure from an XML document) capabilities.
- In fact, ADB is designed to support any programming language with some customized templates and also those who write the programming language.
- Currently it has a full implementation only for the Java programming language, and a relatively less implementation for the C programming language.

Apache Axis2

- Set of tools to simplify Web Services
 - Services
 - Create a Web Service from any Java class
 - Create Web Service skeletons from WSDL files
 - Build war file for deployment on any Java-based browser
 - Clients
 - Create client stubs from WSDL files
- Usage
 - Eclipse plugins
 - Integrated into Java EE Version of Eclipse
 - Download free from <http://www.eclipse.org/downloads/>
 - Choose “Eclipse IDE for Java EE Developers”
 - Command-line tools
 - Windows and Unix

Axis2 > Creating Services



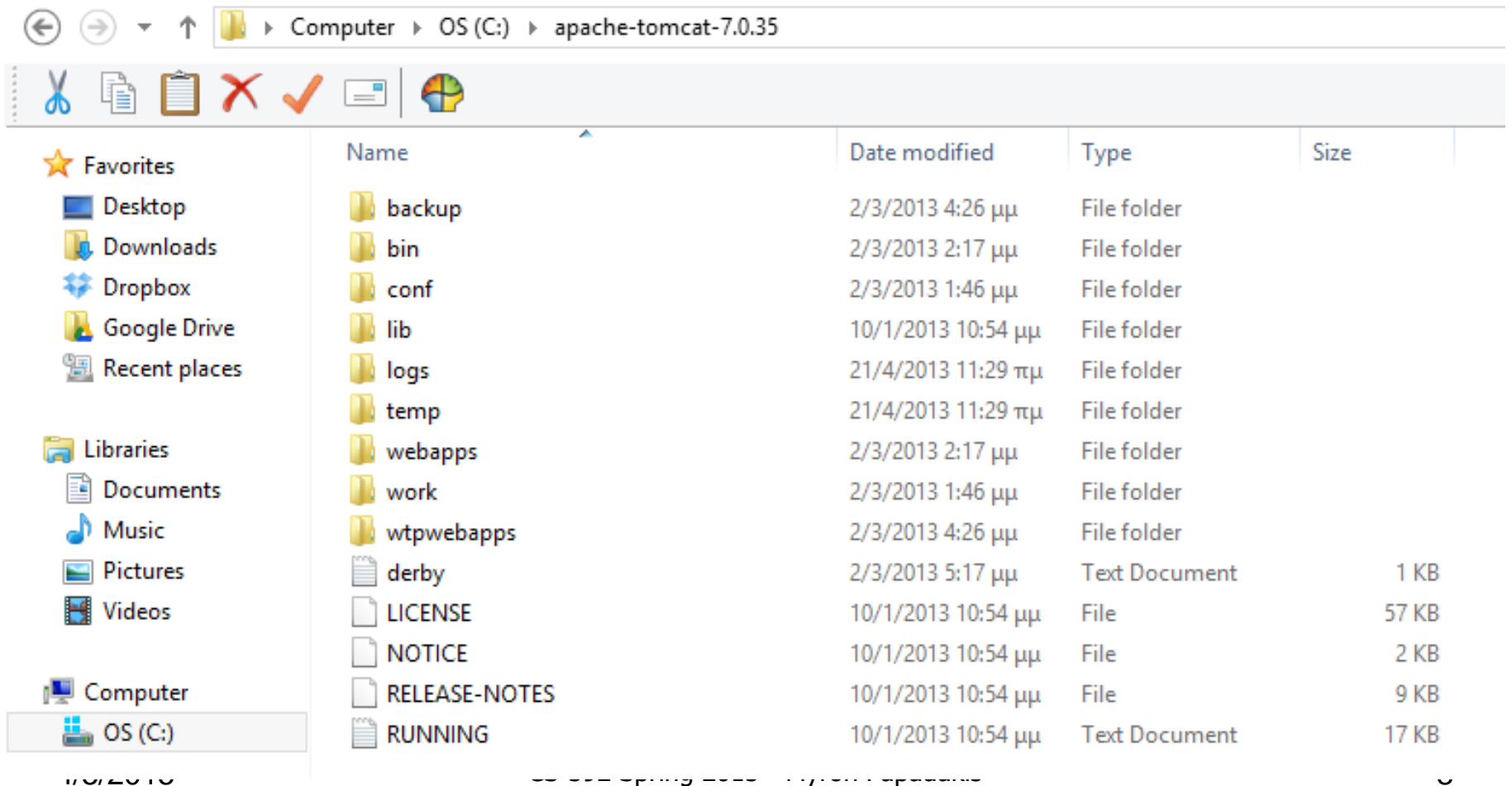
Apache Tomcat

- Download Apache Tomcat 7
- <http://tomcat.apache.org/download-70.cgi>
- Building Apache Tomcat requires a JDK to be installed.
- You can download one from <http://www.oracle.com/technetwork/java/javase/downloads/index.html> or from another JDK vendor.
- Set an environment variable `JAVA_HOME` to the pathname of the directory into which you installed the JDK release



Apache Tomcat

- Extract the zip file for example to local disk C:



The screenshot shows a Windows Explorer window with the address bar set to "Computer > OS (C:) > apache-tomcat-7.0.35". The left sidebar shows the navigation pane with "OS (C:)" selected. The main pane displays a list of files and folders with columns for Name, Date modified, Type, and Size.

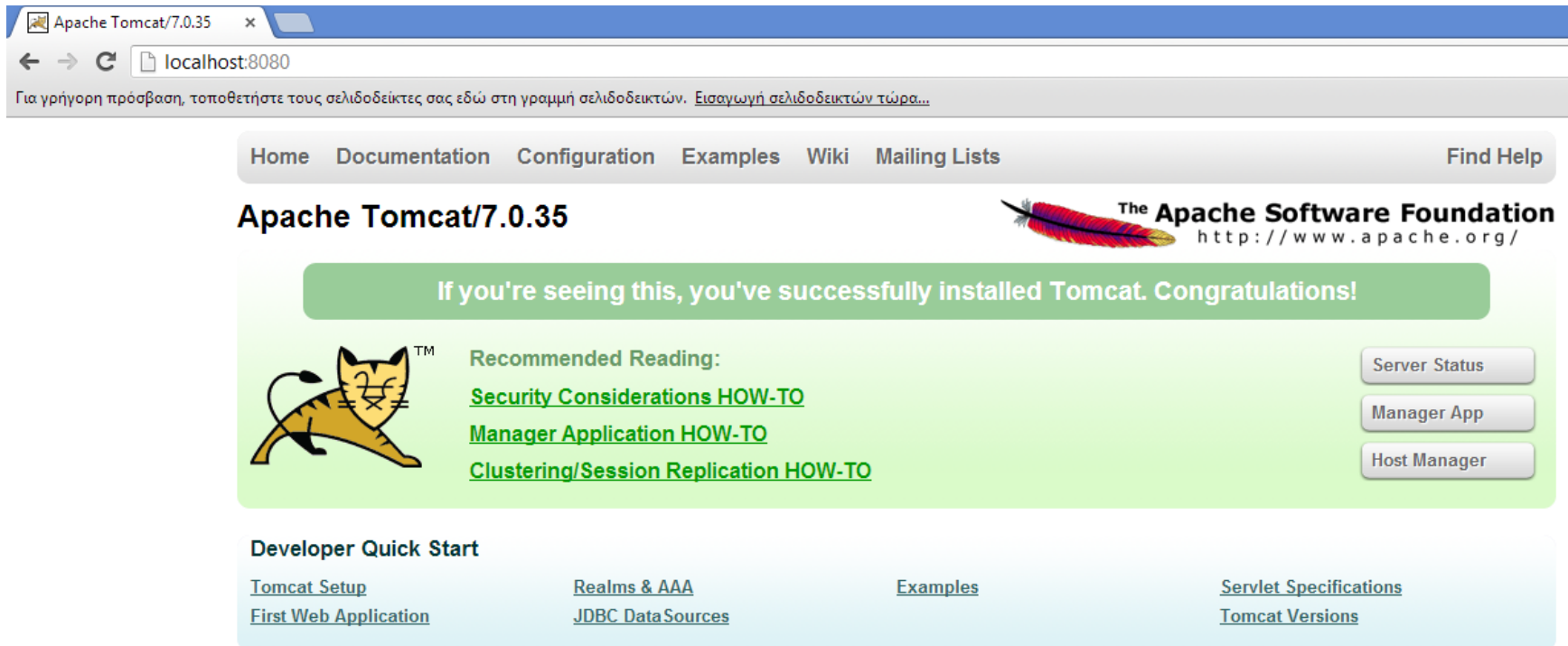
Name	Date modified	Type	Size
backup	2/3/2013 4:26 μμ	File folder	
bin	2/3/2013 2:17 μμ	File folder	
conf	2/3/2013 1:46 μμ	File folder	
lib	10/1/2013 10:54 μμ	File folder	
logs	21/4/2013 11:29 πμ	File folder	
temp	21/4/2013 11:29 πμ	File folder	
webapps	2/3/2013 2:17 μμ	File folder	
work	2/3/2013 1:46 μμ	File folder	
wtpwebapps	2/3/2013 4:26 μμ	File folder	
derby	2/3/2013 5:17 μμ	Text Document	1 KB
LICENSE	10/1/2013 10:54 μμ	File	57 KB
NOTICE	10/1/2013 10:54 μμ	File	2 KB
RELEASE-NOTES	10/1/2013 10:54 μμ	File	9 KB
RUNNING	10/1/2013 10:54 μμ	Text Document	17 KB

Apache Tomcat

- Environment Variable: **CATALINA_HOME** to the path of apache tomcat
- Scripts: **startup**, shutdown inside %CATALINA_HOME%/bin
- Test Tomcat: <http://localhost:8080/>
- **Tomcat instances**
 - if Tomcat has been configured for multiple instances, each instance will have its own CATALINA_BASE configured.

Test Apache Tomcat

Tomcat Running on port 8080




The screenshot shows a web browser window with the title "Apache Tomcat/7.0.35" and the address bar containing "localhost:8080". Below the browser window, the Apache Tomcat 7.0.35 homepage is displayed. The page features a navigation menu with links for Home, Documentation, Configuration, Examples, Wiki, and Mailing Lists, along with a "Find Help" button. The main heading is "Apache Tomcat/7.0.35" next to the Apache Software Foundation logo and URL. A green banner reads "If you're seeing this, you've successfully installed Tomcat. Congratulations!". Below this, there is a "Recommended Reading" section with links to "Security Considerations HOW-TO", "Manager Application HOW-TO", and "Clustering/Session Replication HOW-TO". To the right of these links are three buttons: "Server Status", "Manager App", and "Host Manager". At the bottom, a "Developer Quick Start" section contains links for "Tomcat Setup", "First Web Application", "Realms & AAA", "JDBC Data Sources", "Examples", "Servlet Specifications", and "Tomcat Versions".

Apache Tomcat/7.0.35

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

The Apache Software Foundation
<http://www.apache.org/>

If you're seeing this, you've successfully installed Tomcat. Congratulations!

 Recommended Reading:

- [Security Considerations HOW-TO](#)
- [Manager Application HOW-TO](#)
- [Clustering/Session Replication HOW-TO](#)

Server Status
Manager App
Host Manager

Developer Quick Start

- [Tomcat Setup](#)
- [First Web Application](#)
- [Realms & AAA](#)
- [JDBC Data Sources](#)
- [Examples](#)
- [Servlet Specifications](#)
- [Tomcat Versions](#)

Multiple Tomcat Instances

- Copy files from the initial tomcat folder to another folder (i.e to C:\tomcat_instance2)
 - Change the following ports of the new instance (conf/server.xml)
 - Shutdown port
 - Connector port
 - Ajp port
 - Redirect port

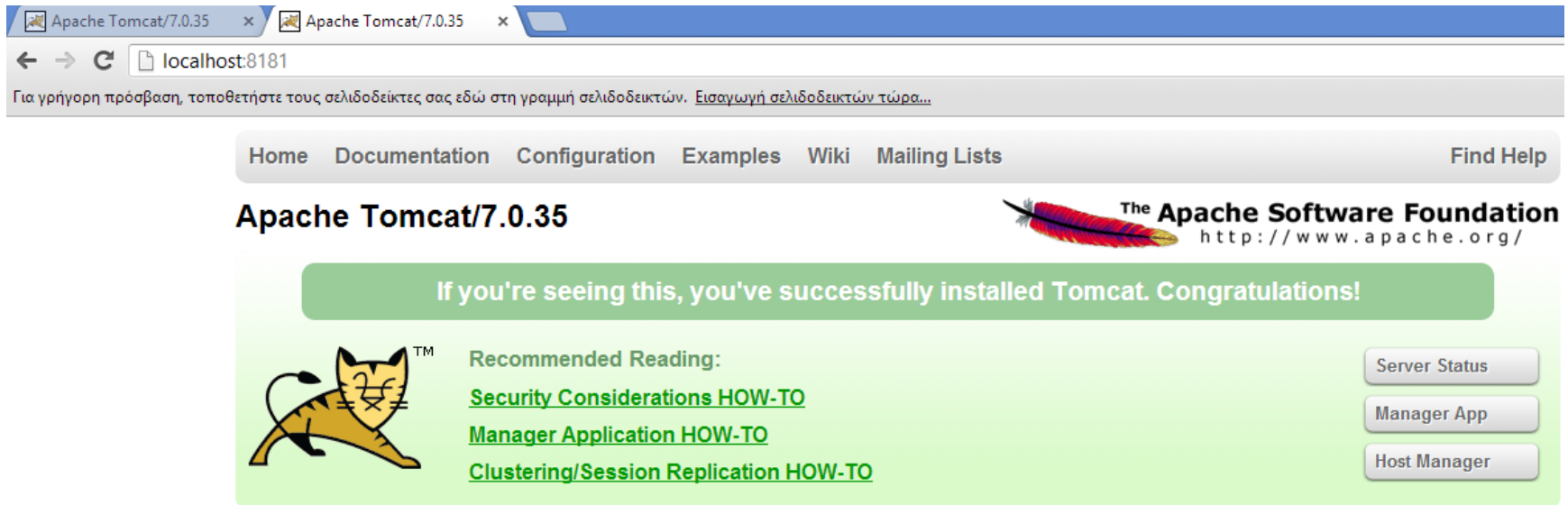
```
21 -->
22 <Server port="8105" shutdown="SHUTDOWN">
```

```
69 --/
70 <Connector port="8181" protocol="HTTP/1.1"
71           connectionTimeout="20000"
72           redirectPort="81443" />
```

```
89
90 <!-- Define an AJP 1.3 Connector on port 8009 -->
91 <Connector port="8109" protocol="AJP/1.3" redirectPort="81443" />
92
```

Testing Additional Tomcat Instances

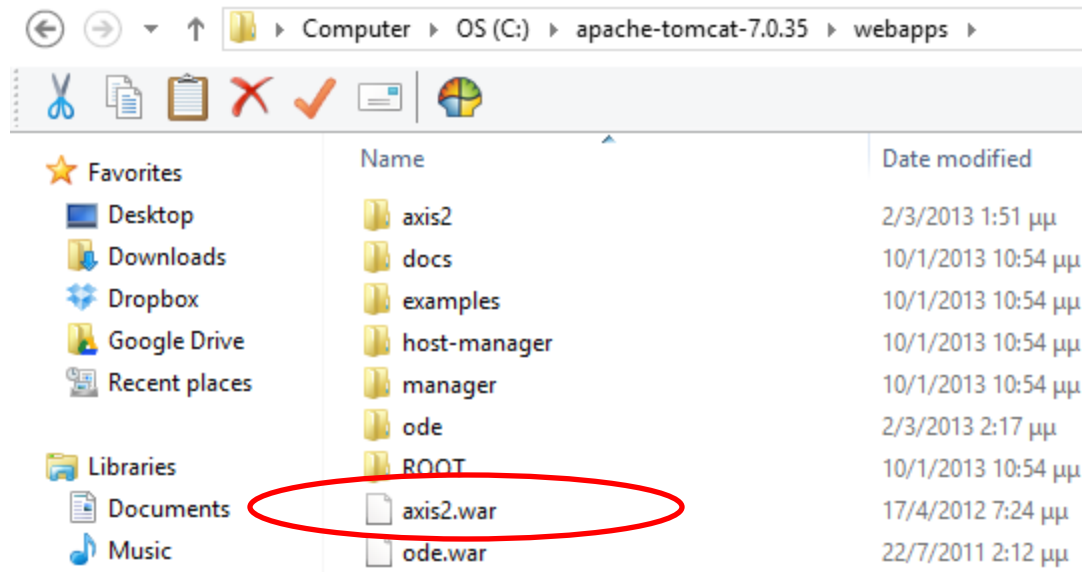
2nd Tomcat Instance running on port 8181 (the 1st runs already on port 8080)



The screenshot shows a web browser window with two tabs labeled "Apache Tomcat/7.0.35". The address bar displays "localhost:8181". Below the address bar is a navigation menu with links for "Home", "Documentation", "Configuration", "Examples", "Wiki", "Mailing Lists", and "Find Help". The main content area features the "Apache Tomcat/7.0.35" logo and the "The Apache Software Foundation" logo with the URL "http://www.apache.org/". A green banner reads "If you're seeing this, you've successfully installed Tomcat. Congratulations!". To the left is the Tomcat logo (a stylized orange cat). To the right, under "Recommended Reading:", are three links: "Security Considerations HOW-TO", "Manager Application HOW-TO", and "Clustering/Session Replication HOW-TO". On the far right, there are three buttons: "Server Status", "Manager App", and "Host Manager".

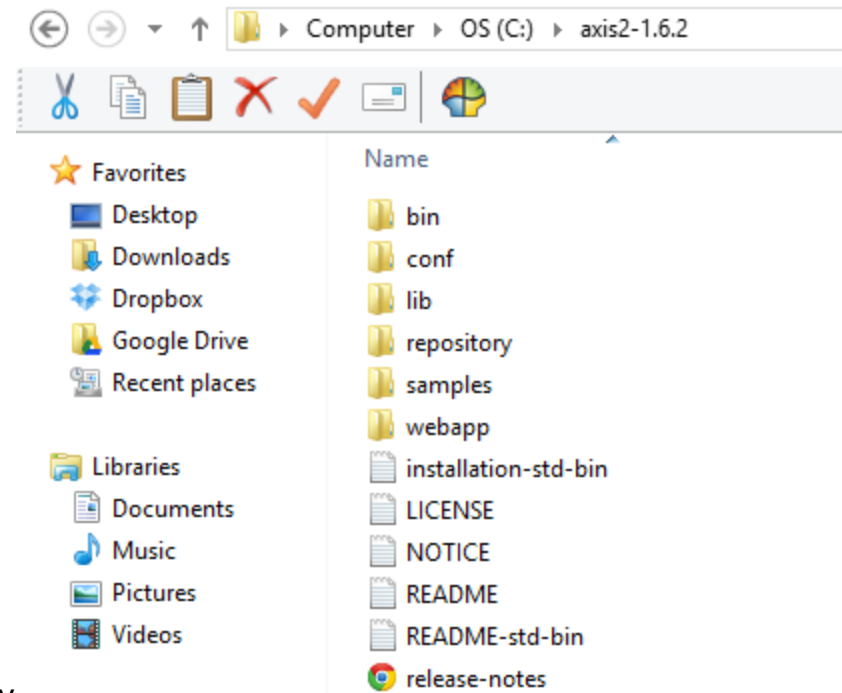
Axis2 Installation

- <http://axis.apache.org/axis2/java/core/download.cgi>
- Get the **WAR** distribution for deployment on Tomcat
- Place it into the “*webapps*” folder of tomcat



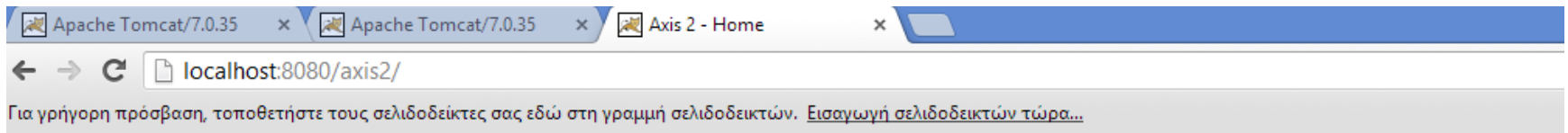
Axis2 Installation

- Get also the **standard distribution** for service and client programming
- Choose “zip” under *Binary Distribution*
- Unzip to folder of your choice, ie “C:\”
- Test Axis2 deployment
- <http://localhost:8080/axis2/>



Testing Axis2 Deployment

<http://localhost:8080/axis2/>



Welcome!

Welcome to the new generation of Axis. If you can see this page you have successfully deployed the Axis2 Web Application. However, to ensure that Axis2 is properly working

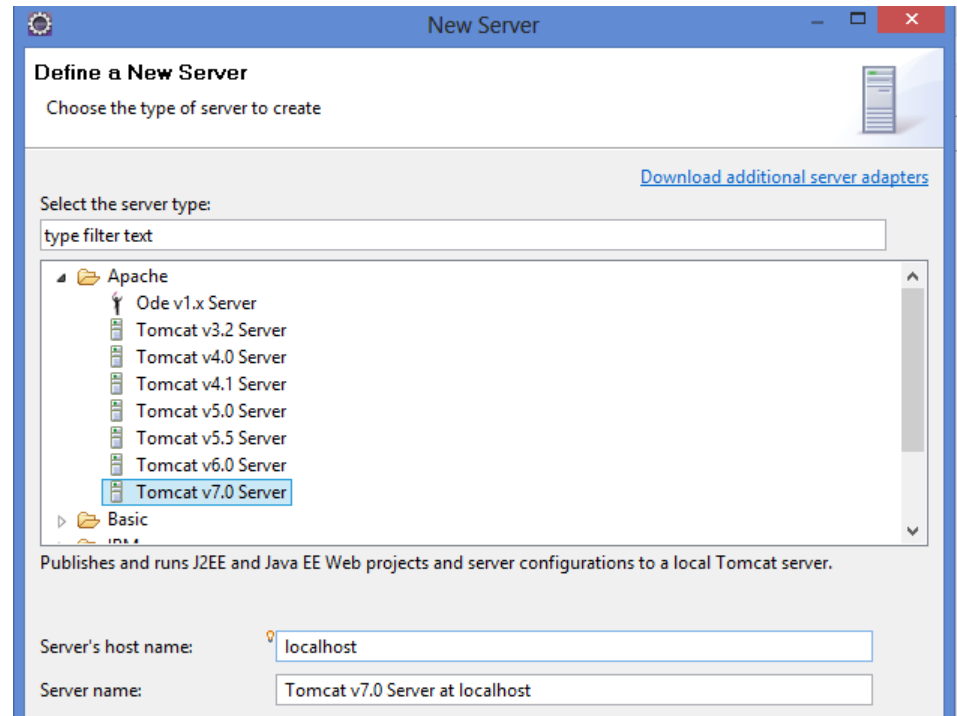
- [Services](#)
View the list of all the available services deployed in this server.
- [Validate](#)
Check the system to see whether all the required libraries are in place and view the system information.
- [Administration](#)
Console for administering this Axis2 installation.

Eclipse Axis2 Lab Example



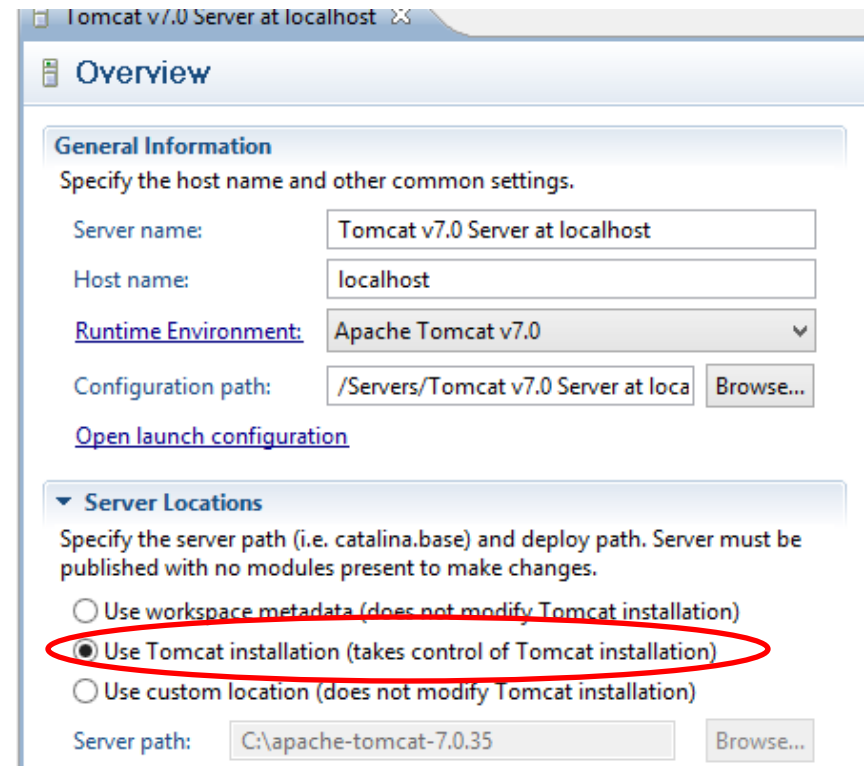
Eclipse > Add Server

- Download Eclipse for Java EE developers (i.e Helios, Indigo, Juno)
 - Helios is the preferred one
- Window → Show view → Servers → Add server
- Select “**Server Type**” and then under “**Apache**”, “**Tomcat VX.0 Server**” .
- Then, define “Tomcat installation Directory”, JRE



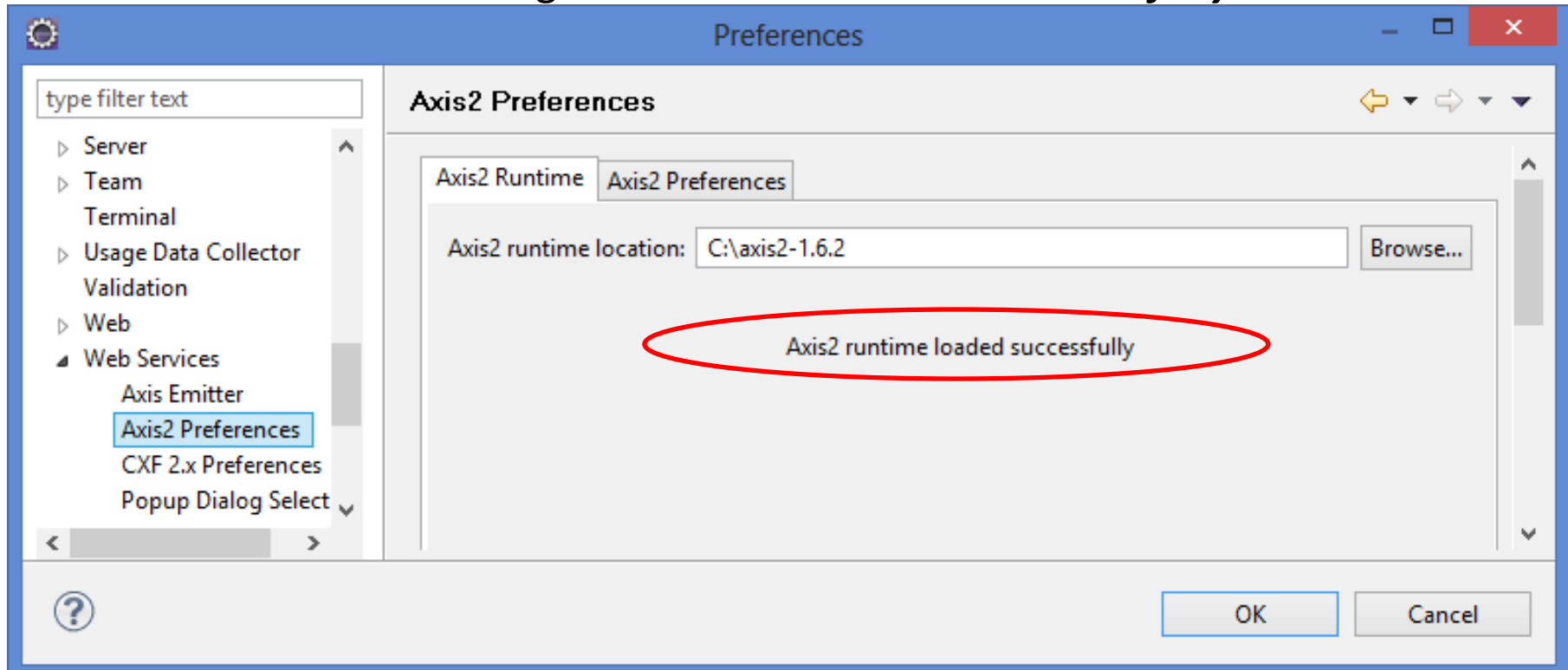
Eclipse > Add Server

- By default, Eclipse does not modify referenced Tomcat but rather creates an instance of the specified server
- If you want to use the genuine, then select from “**Servers**”, right click and select “**Open**”.
- Choose “**Use Tomcat Installation**” under “**Server Locations**” and set the server path



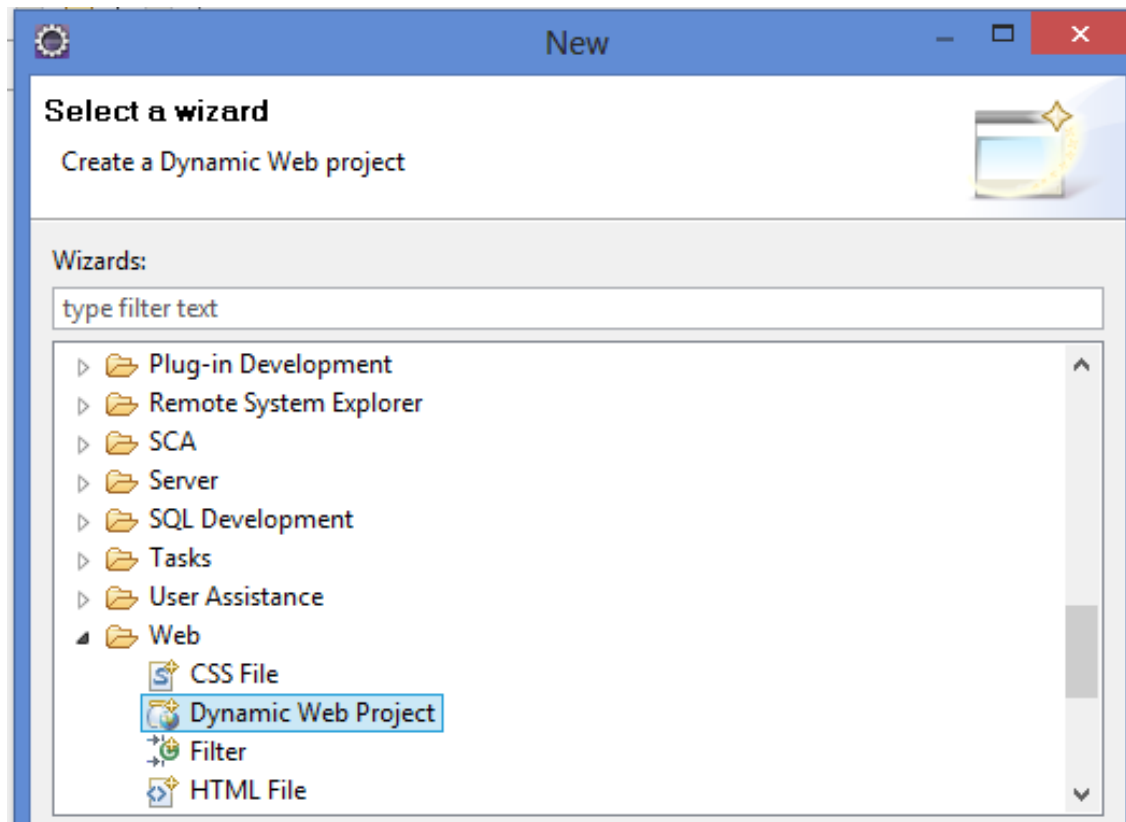
Tell Eclipse About Axis2

- Window → Preferences → Web Services → Axis2 Preferences
- Set install path of Axis2 Set **“Axis2 runtime location”** to the path you have unzipped Axis2 (i.e C:\axis2-1.6.2)
- You will see the message **“Axis2 runtime loaded successfully”**.



Eclipse and Axis2 Project

- File → New → Other → Web → Dynamic Web Project



Eclipse and Axis2 Project

- Set the following:
 - **Target Runtime:** set to the server you want
 - **Dynamic web module version:** Don't use "3", but one of "2.2 ,2.3, 2.4, 2.5"
 - **Under Configuration:** Project Facets add: **"Axis2 Web Services"**

The image shows two overlapping dialog boxes in the Eclipse IDE. The left dialog is titled 'New Dynamic Web Project' and contains the following fields:

- Project name:** TravelReservationProject
- Project location:** Use default location. Location: C:\workspace\TravelReservationProject
- Target runtime:** Apache Tomcat v7.0
- Dynamic web module version:** 2.5 (circled in red)
- Configuration:** <custom> (with a 'Modify...' button circled in red)

The right dialog is titled 'Project Facets' and shows a list of facets with their versions. The 'Axis2 Web Services' facet is circled in red.

Project Facet	Version
<input checked="" type="checkbox"/> Axis2 Web Services	
<input type="checkbox"/> CXF 2.x Web Services	1.0
<input checked="" type="checkbox"/> Dynamic Web Module	2.5
<input checked="" type="checkbox"/> Java	1.7
<input type="checkbox"/> JavaScript	1.0
<input type="checkbox"/> JavaServer Faces	2.0
<input type="checkbox"/> JAX-RS (REST Web Services)	1.1
<input type="checkbox"/> JPA	2.0
<input type="checkbox"/> WebDoclet (XDoclet)	1.2.3

Eclipse Top-Down WS Example

Eclipse Lab Example (Top-Down Approach)

- Create a new Web Service that calculates the area of a rectangle. We need a single operation (i.e. calculateRectangleArea)
 - Input: width and height (float).
 - Use a complexType (i.e Dimension) for the input
 - Output: float
- Use the top-down for building this service (first we must define the WSDL file for this service)
- Test the WS using a client (Web Services Explorer, manually written client or SOAPUI)

Eclipse Lab Example (Area Calculation)

- Create a project with the support of Axis2 features. Open **File -> New -> Other... -> Web -> Dynamic Web Project**

The screenshot shows the Eclipse IDE interface during the creation of a new Dynamic Web Project. The 'New Dynamic Web Project' dialog is open, showing the following configuration:

- Project name: Axis2WS
- Project location: C:\workspace\Axis2WS (Use default location checked)
- Target runtime: Apache Tomcat v7.0
- Dynamic web module version: 2.5
- Configuration: <custom> (Modify... button highlighted)

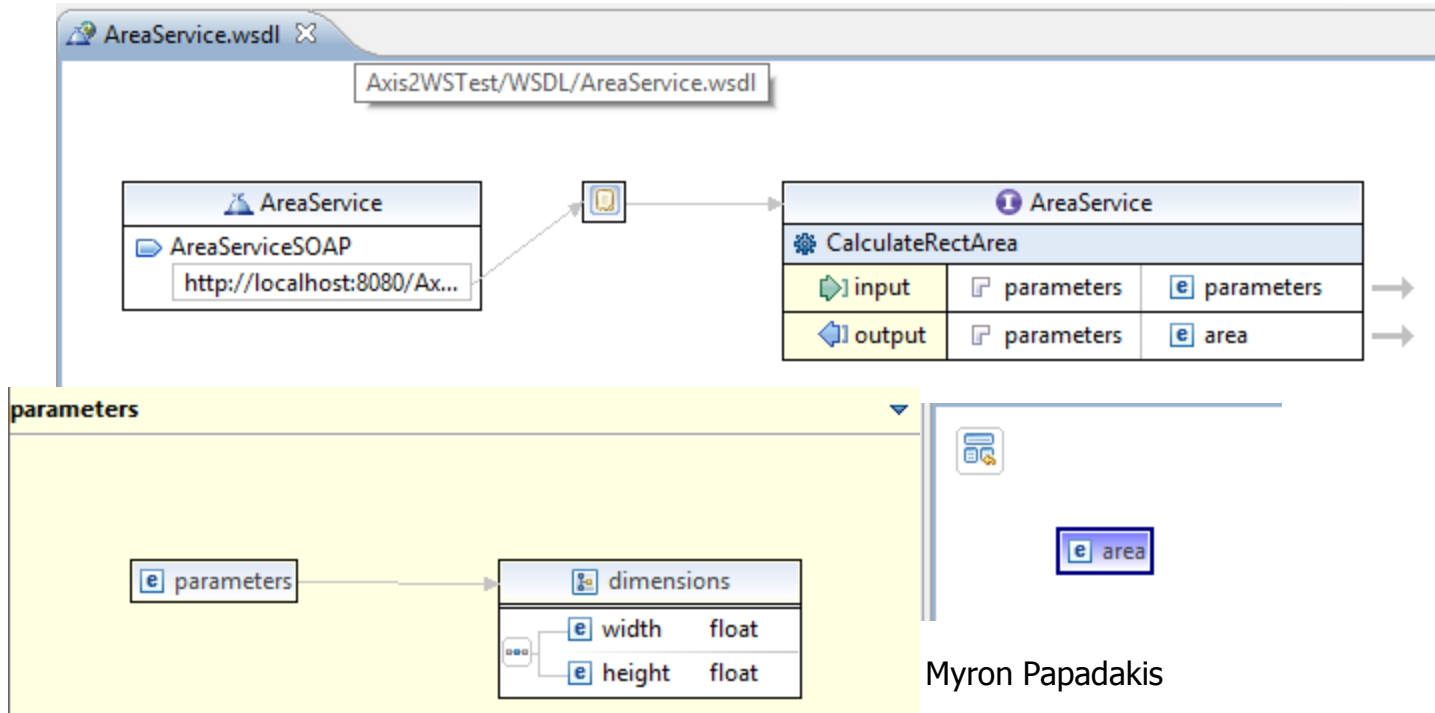
The 'Project Facets' dialog is also open, showing the following configuration:

- Configuration: <custom>
- Project Facet: Axis2 Web Services (checked)
- Axis2 Web Services Core: 1.1 (checked)
- Axis2 Web Services Extensions: 1.1 (checked)
- Dynamic Web Module: 2.5 (checked)
- Java: 1.7 (checked)
- JavaScript: 1.0 (unchecked)
- JavaServer Faces: 2.0 (unchecked)
- JAX-RS (REST Web Services): 1.1 (unchecked)
- JPA: 2.0 (unchecked)
- WebDoclet (XDoclet): 1.2.3 (unchecked)

Eclipse Lab Example > Create WSDL

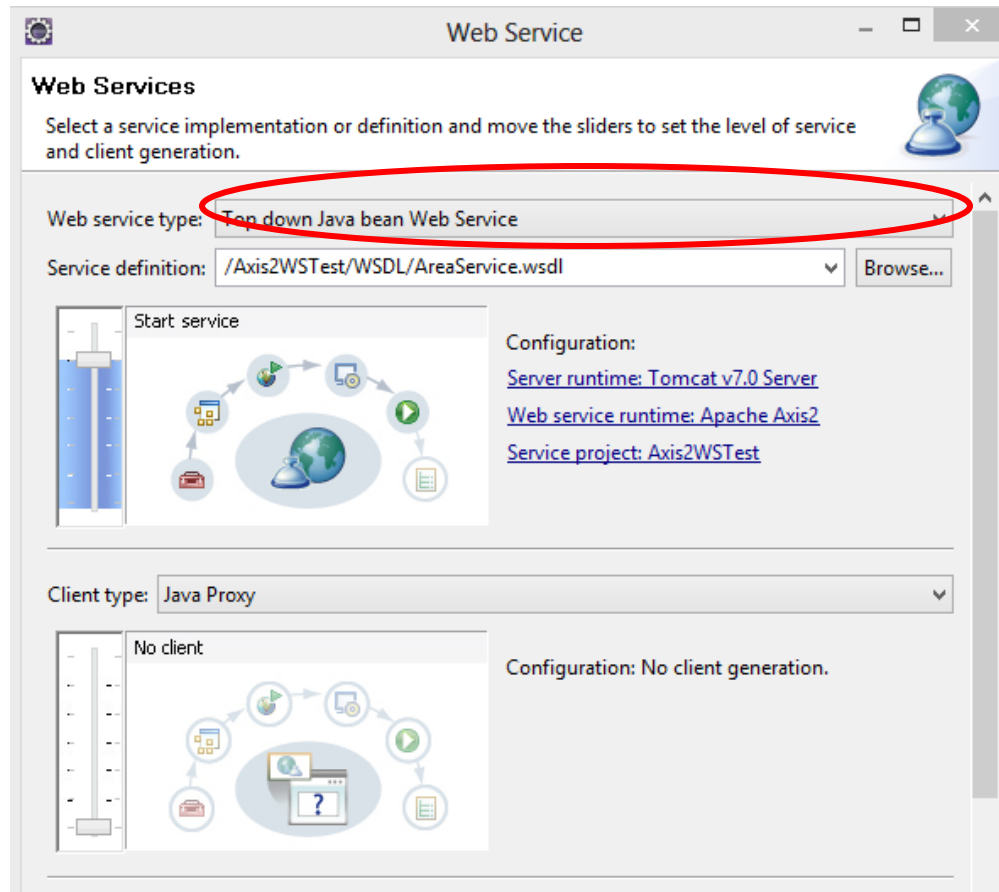
- Create a folder called WSDL on the Axis2WSTest project.
- Right click on this folder and then New → Other → Web Services → WSDL and name this file AreaService.wsdl

<http://localhost:8080/Axis2WSTest/services/AreaService>



Eclipse Lab Example

- By selecting AreaService.wsdl source the Open File -> New -> Other... -> Web Services -> Web Service



Skeleton JAVA bean configuration page

Web Service

Axis2 Web Service Skeleton Java Bean Configuration

Select the appropriate code generation settings

Service Name: AreaService

Port Name: AreaServiceSOAP

Databinding: ADB

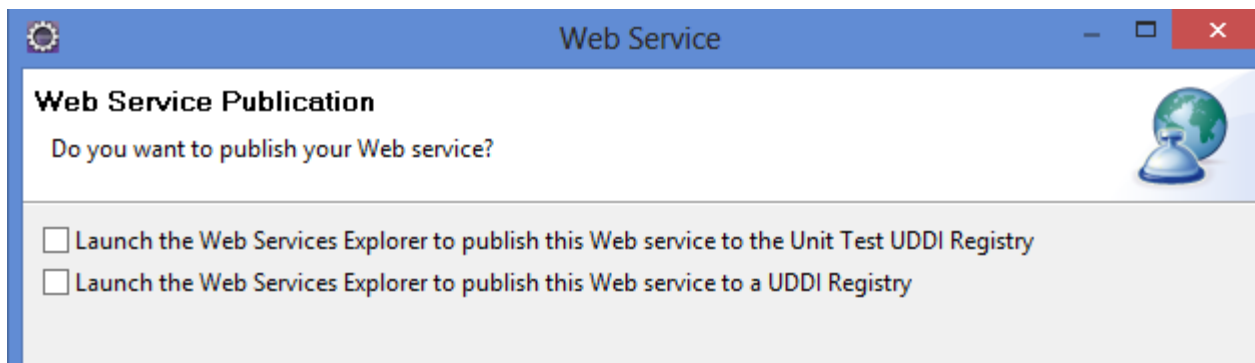
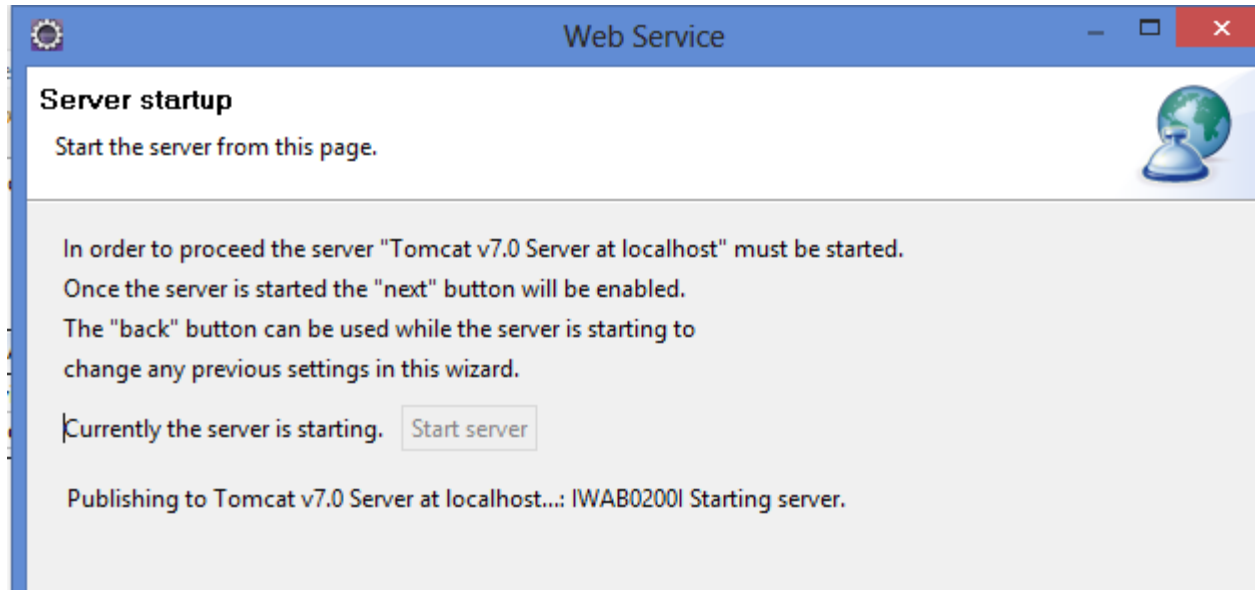
Custom package name: org.tempuri.areaservice

Generate an interface for the skeleton

Generate all types for all elements referred by schemas

Namespace	Package
http://schemas.xmlsoap.org/wsdl/	org.xmlsoap.schemas.wsdl
http://tempuri.org/AreaService/	org.tempuri.areaservice
http://www.w3.org/2001/XMLSchema	org.w3.www._2001.xmlschema
http://schemas.xmlsoap.org/wsdl/soap/	org.xmlsoap.schemas.wsdl.soap

Eclipse Lab Example



Area Service Skeleton

```
AreaService.wsdl | Inline Schema of Are | AreaServiceSkeleton.j | AreaServiceSkeleton.j x
package org.example.www.areaservice;
/**
 * AreaServiceSkeleton java skeleton for the axisService
 */
public class AreaServiceSkeleton{

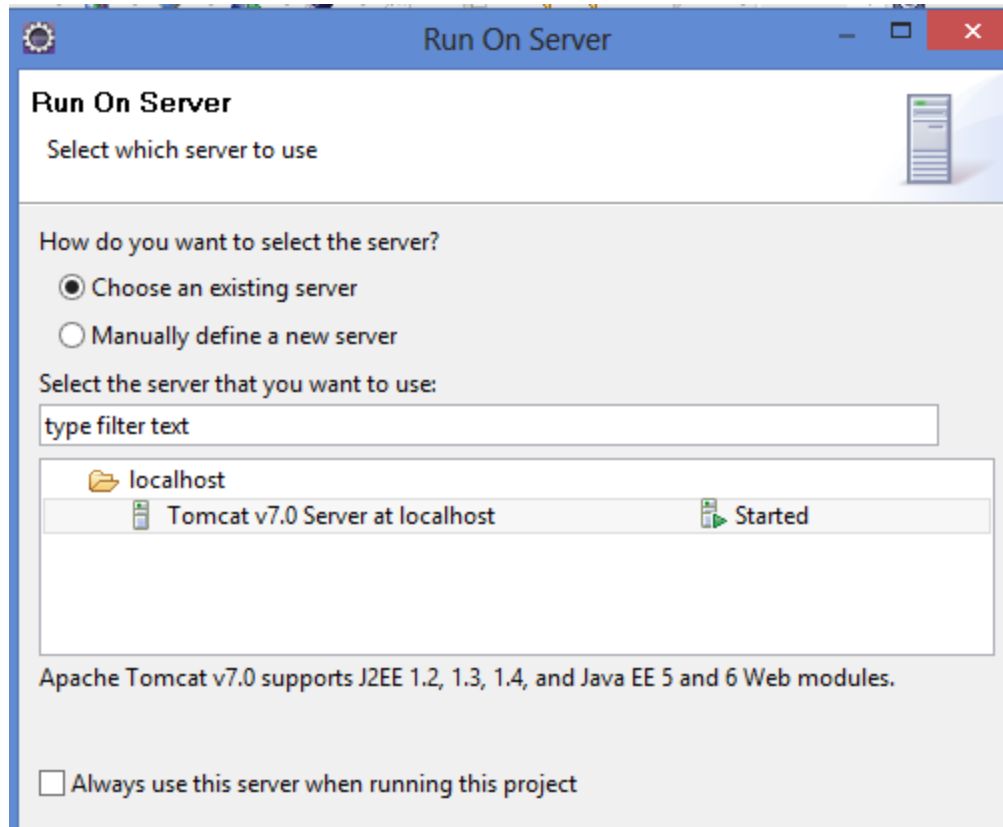
    /**
     * Auto generated method signature
     *
     * @param parameters
     * @return area
     */

    public org.example.www.areaservice.Area calculateRectArea
    (
        org.example.www.areaservice.Parameters parameters
    )
    {
        org.example.www.areaservice.Area area = new org.example.w
        area.setArea(parameters.getParameters().getHeight() *
        parameters.getParameters().getWidth());
        return area;

        //TODO : fill this with the necessary business logic
        // throw new java.lang.UnsupportedOperationException("Please i
    }
}
```

Running the project

- Right Click the Project and select Run As → Run on Server
- Restart the server if prompted to do so



Axis2 Services

The image shows two browser windows side-by-side. The left window is titled 'Axis 2 - Home' and shows the main Axis2 interface. The right window is titled 'List Services' and shows a detailed view of available services.

Left Window (Axis 2 - Home):

- URL: <http://localhost:8080/Axis2WSTest/>
- Logo: The Apache Software Foundation logo (a feather).
- Text: "The Apache Software Foundation" and "http://www.apache.org".
- Section: **Welcome!**
- Text: "Welcome to the new generation of Axis. If you can see this page you have successfully installed Axis. If everything is working, we encourage you to click on the validate link."
- Links:
 - Services** (circled in red): View the list of all the available services deployed in this system.
 - [Validate](#): Check the system to see whether all the required libraries are installed.
 - [Administration](#): Manage the Axis2 system.

Right Window (List Services):

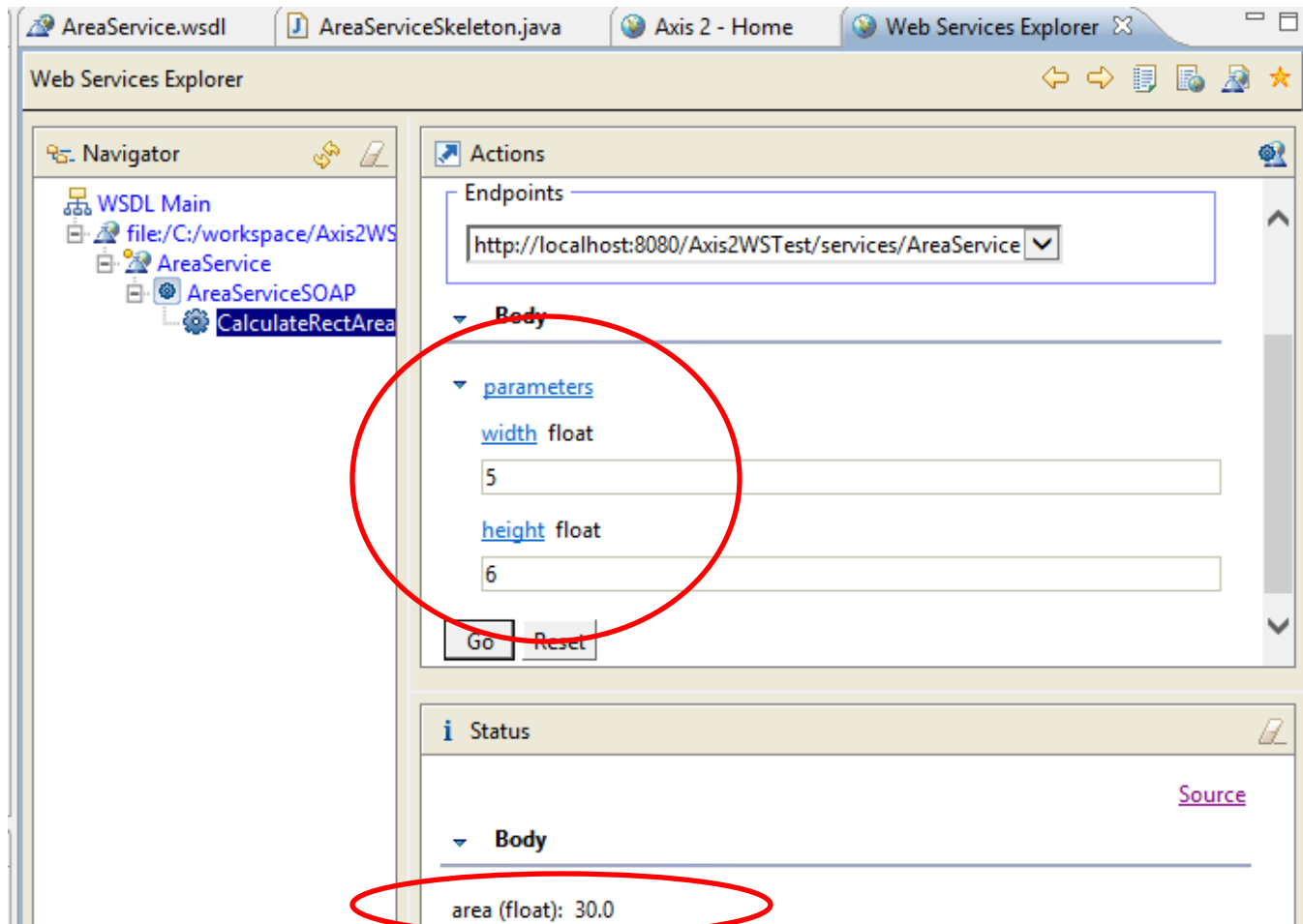
- URL: <http://localhost:8080/Axis2WSTest/services/listServices>
- Section: **Available services**
- Section: **Version**
 - Service Description : Version
 - Service EPR : <http://localhost:8080/Axis2WSTest/services/Version>
 - Service Status : Active
 - Available Operations
 - getVersion
- Section: **AreaService**
 - Service Description : AreaService
 - Service EPR : <http://localhost:8080/Axis2WSTest/services/AreaService>
 - Service Status : Active

Web Service Client

The screenshot shows an IDE interface with a project tree on the left. The 'WSDL' folder is expanded, and 'AreaService' is selected. A context menu is open over 'AreaService', listing various actions. The 'Web Services' option is highlighted, and its sub-menu is visible, showing 'Test with Web Services Explorer' as the selected option. Other options in the sub-menu include 'Publish WSDL File' and 'Generate Java Bean Skeleton'. A small inset window in the top right shows a service named 'CalculateRectAr' with two input/output fields: 'input' and 'output'.

⚙️ CalculateRectAr	
➡️ input	📄
⬅️ output	📄

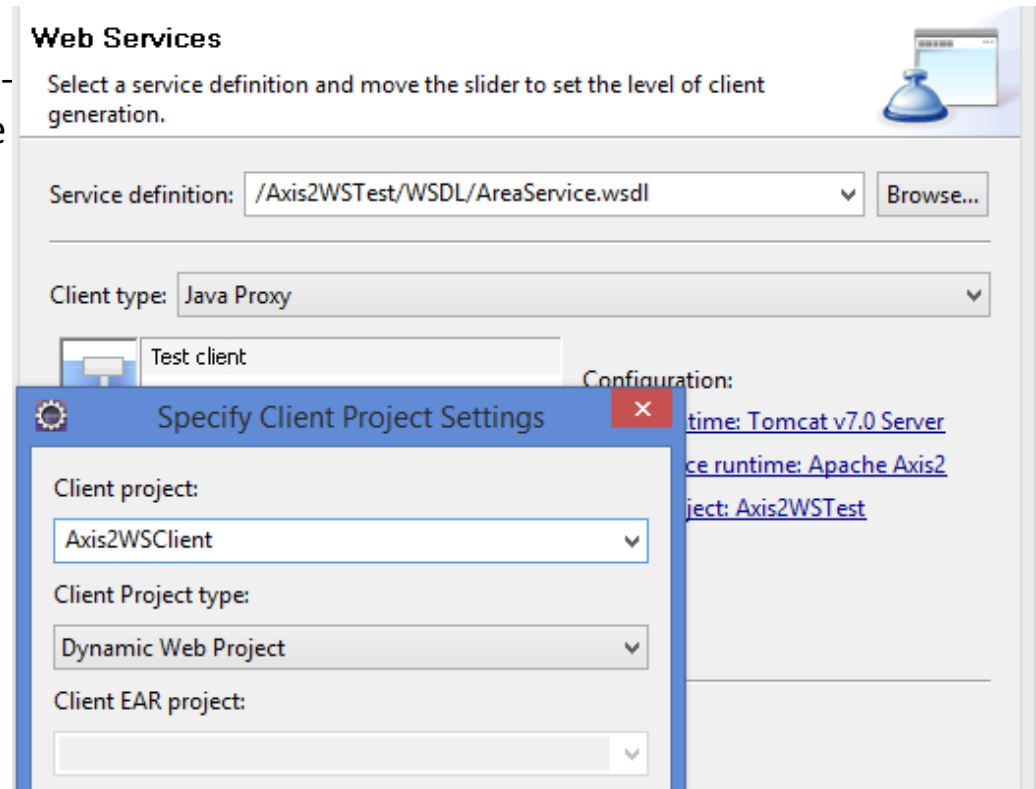
Web Service Client



Consuming Web service by writing a Web Service Client

Consuming Web service using Web Service Client

- Select the AreaService.wsdl file in the Axis2Test/WSDL folder Open File -> New -> Other... -> Web Services -> Web Service Client.
- Click Next.
- Move the client slider to the Test Client position.
- Click the Client project link under Configuration and type in Axis2WClient as the name of the Client project in the dialog that appears:



Consuming Web services using Web

Web Services
Select a service definition and move the slider to set the level of client generation.

Progress Information
Operation in progress...
Installing Axis2 Web Services Core facet...

Axis2 Client Web Service Configuration
Select the appropriate code generation settings

Service Name: AreaService

Port Name: AreaServiceSOAP

Databinding: ADB

Custom package name: org.tempuri.areaservice

Client mode

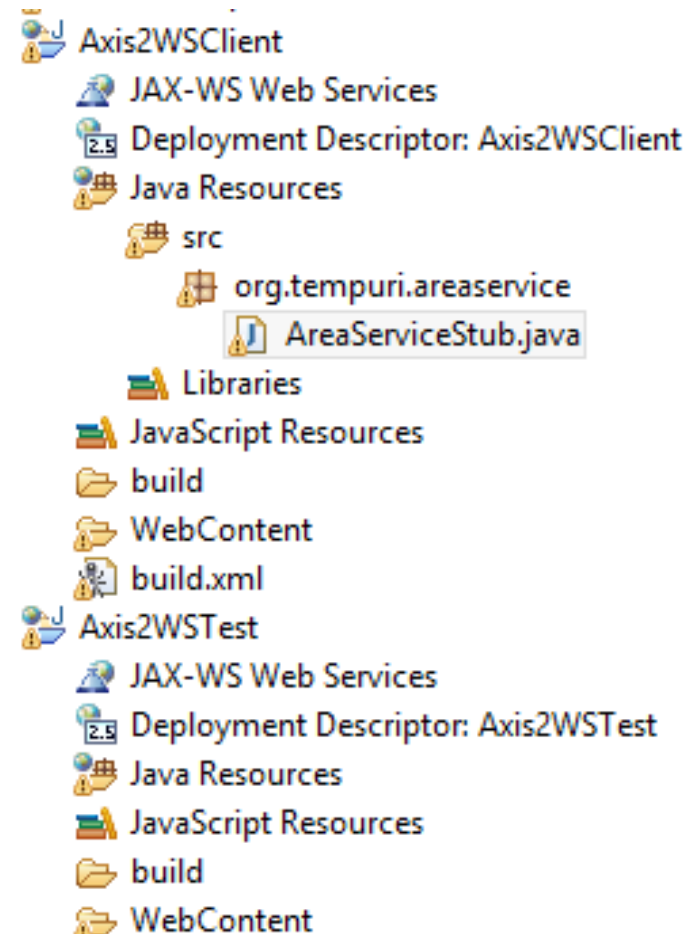
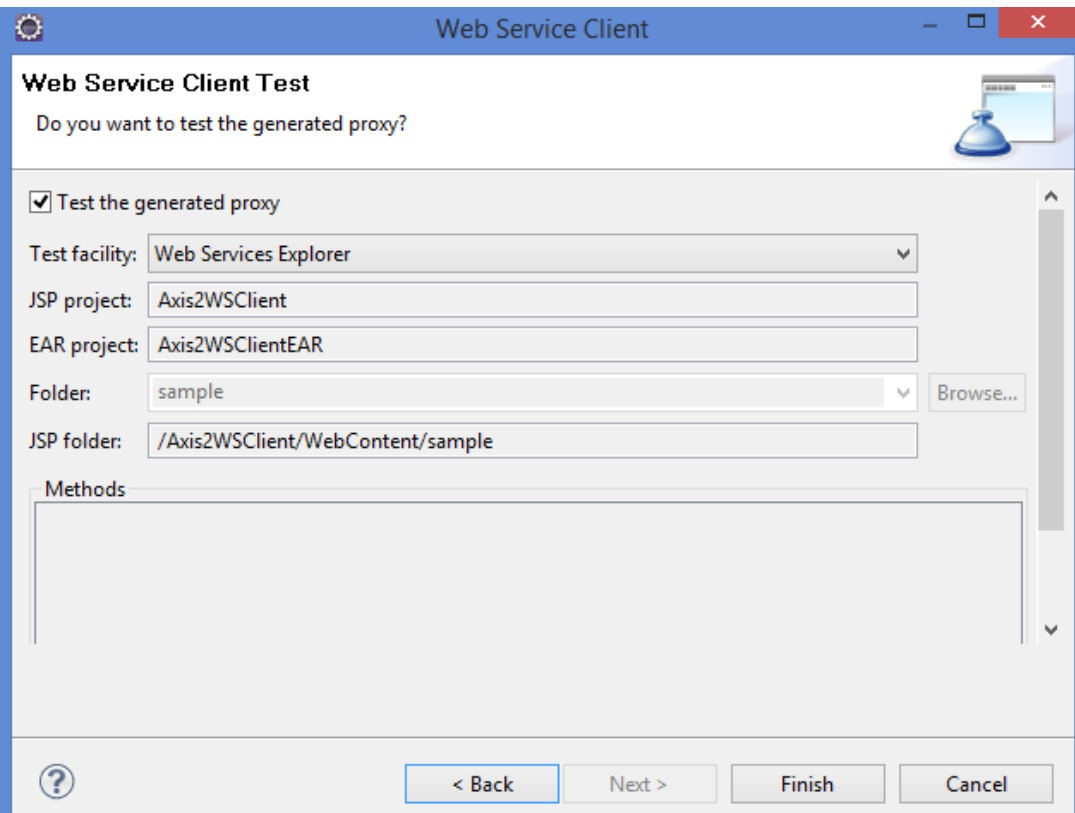
- Generate a client which supports both synchronous and asynchronous invocation
- Generate a synchronous client
- Generate an asynchronous client

Generate a JUnit test case to test the service

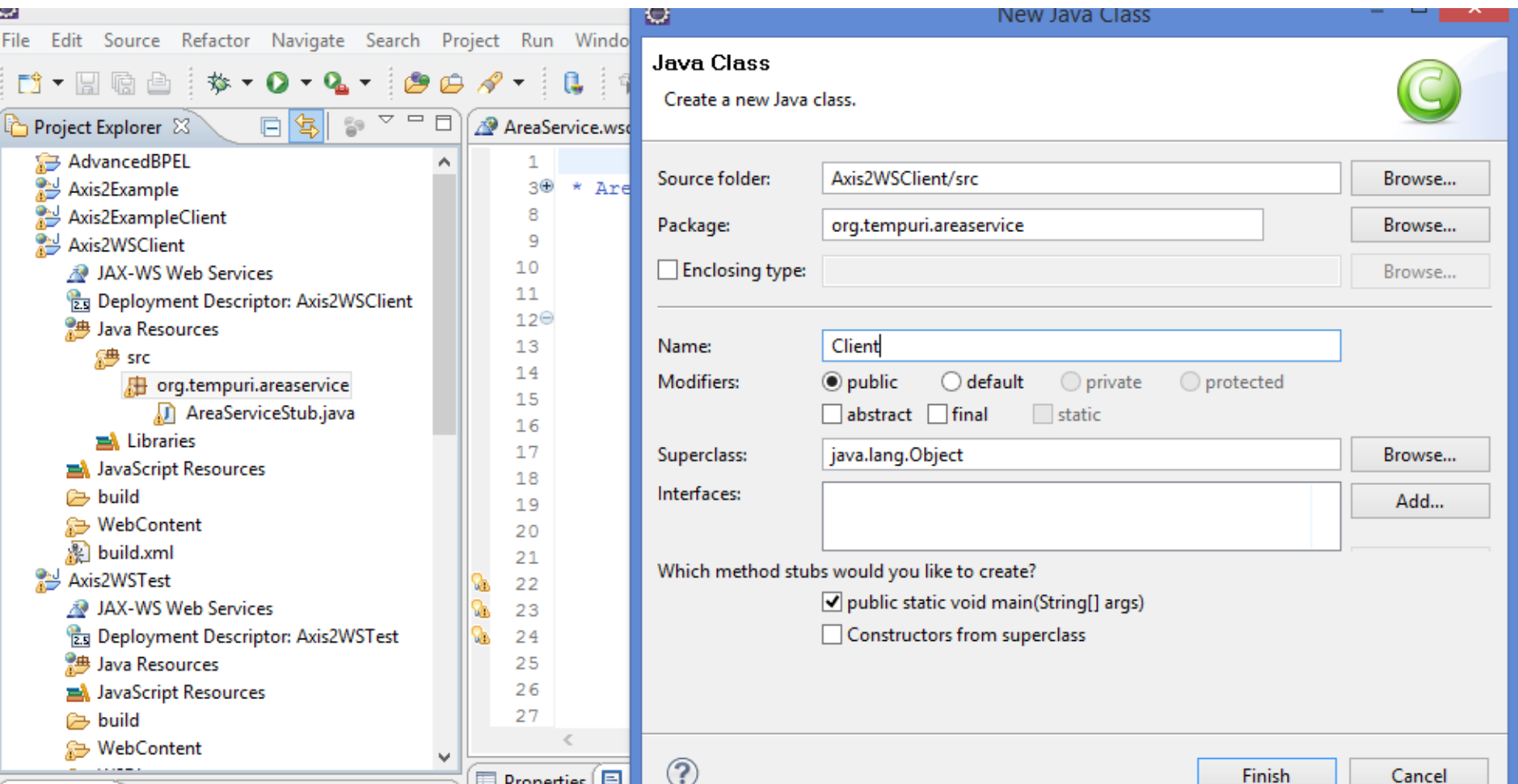
Generate all types for all elements referred to by schemas

Namespace	Package
http://schemas.xmlsoap.org/wsdl/	org.xmlsoap.schemas.wsdl
http://tempuri.org/AreaService/	org.tempuri.areaservice
http://www.w3.org/2001/XMLSchema	org.w3.www._2001.xmlschema
http://schemas.xmlsoap.org/wsdl/soap/	org.xmlsoap.schemas.wsdl.soap

Consuming Web service using Web Service Client



Consuming Web service using Web Service Client



Consuming Web service using Web Service Client

The screenshot shows a Java IDE window titled 'Web Services Explorer' with the URL 'http://localhost:8080/Axis2WSTest/ser'. The code in the editor is as follows:

```
16 public static void main(String[] args) {
17
18     AreaServiceStub stub;
19     try {
20         stub = new AreaServiceStub();
21         Parameters parameters = new Parameters();
22         Dimensions dimensions = new Dimensions();
23         float height = 0, width = 0;
24
25         Scanner scanner = new Scanner(System.in);
26         System.out.println("Enter height:");
27         height = scanner.nextFloat();
28         dimensions.setHeight(height);
29
30         System.out.println("Enter width:");
31         width = scanner.nextFloat();
32         dimensions.setWidth(width);
33
34         parameters.setParameters(dimensions);
35         Area resp = stub.calculateRectArea(parameters);
36         float area = resp.getArea();
37         System.out.printf("The area is %f\n", area);
    }
```

The IDE also shows a 'Console' window with the following output:

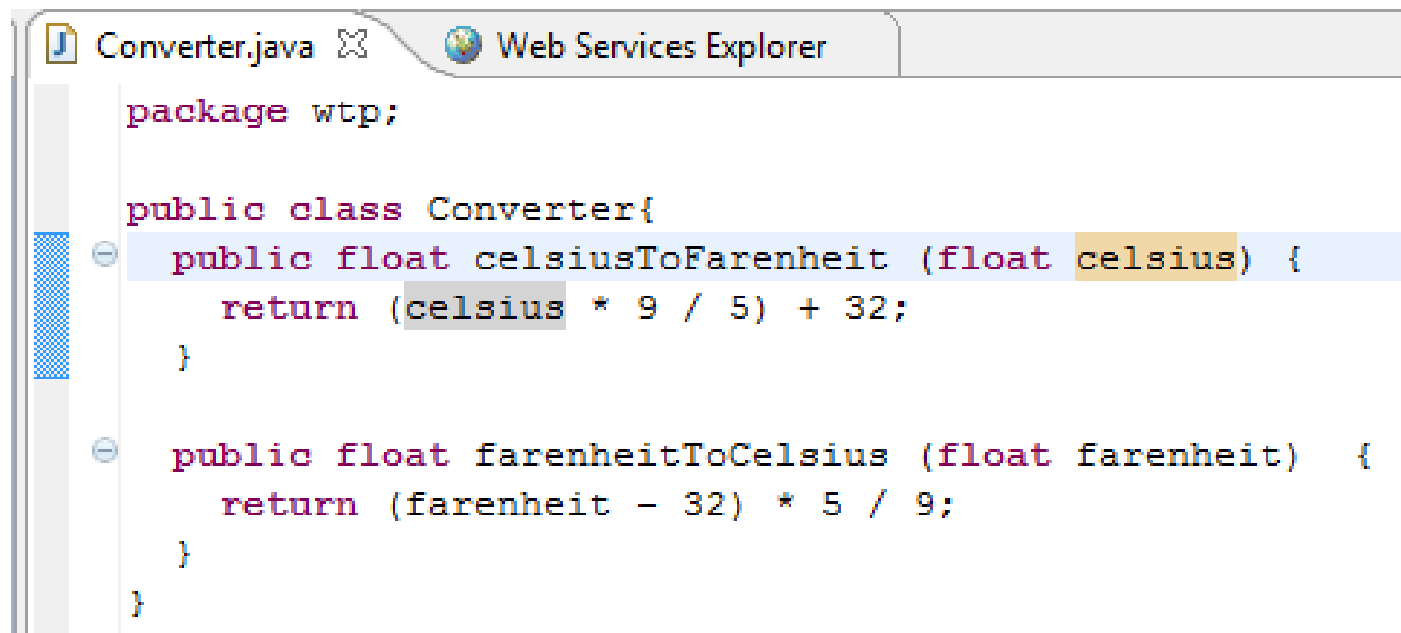
```
<terminated> Client (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (A
Enter height:
5
Enter width:
6
The area is 30.0
```


Bottom-up Example

Temperature Example

Eclipse Lab Example : Creating a Bottom Up Web Service

- Create a new Dynamic Project and a new Java class
- Select the **Converter.java** file.
- Open **File -> New -> Other... -> Web Services -> Web Service**



```
package wtp;

public class Converter{
    public float celsiusToFahrenheit (float celsius) {
        return (celsius * 9 / 5) + 32;
    }

    public float fahrenheitToCelsius (float fahrenheit) {
        return (fahrenheit - 32) * 5 / 9;
    }
}
```

Eclipse > Creating a Bottom Up Web Service


Web Services

Select a service implementation or definition and move the sliders to set the level of service and client generation.

Web service type: Bottom up Java bean Web Service

Service implementation: wtp.Converter


Start service



Configuration:
[Server runtime: Tomcat v7.0 Server](#)
[Web service runtime: Apache Axis2](#)
[Service project: ConverterProj](#)

Client type: Java Proxy

Test client



Configuration:
[Server runtime: Tomcat v7.0 Server](#)
[Web service runtime: Apache Axis2](#)
[Client project: ConverterProjClient](#)

Publish the Web service
 Monitor the Web service

Web Service

Axis2 Web Service Java Bean Configuration

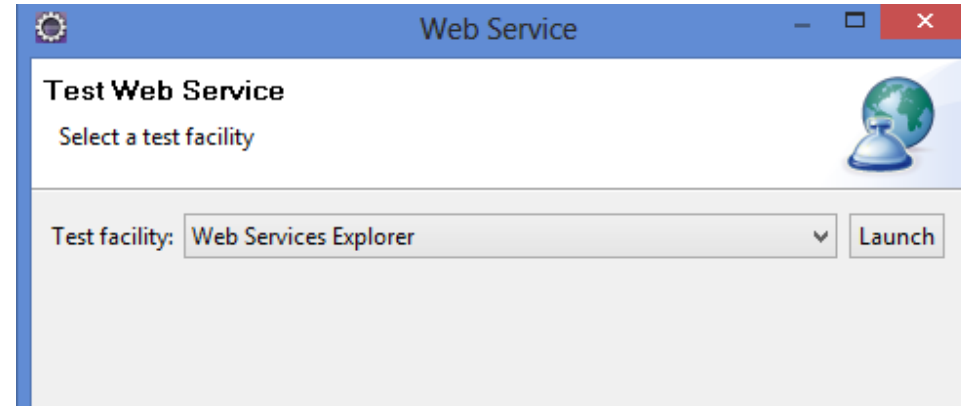
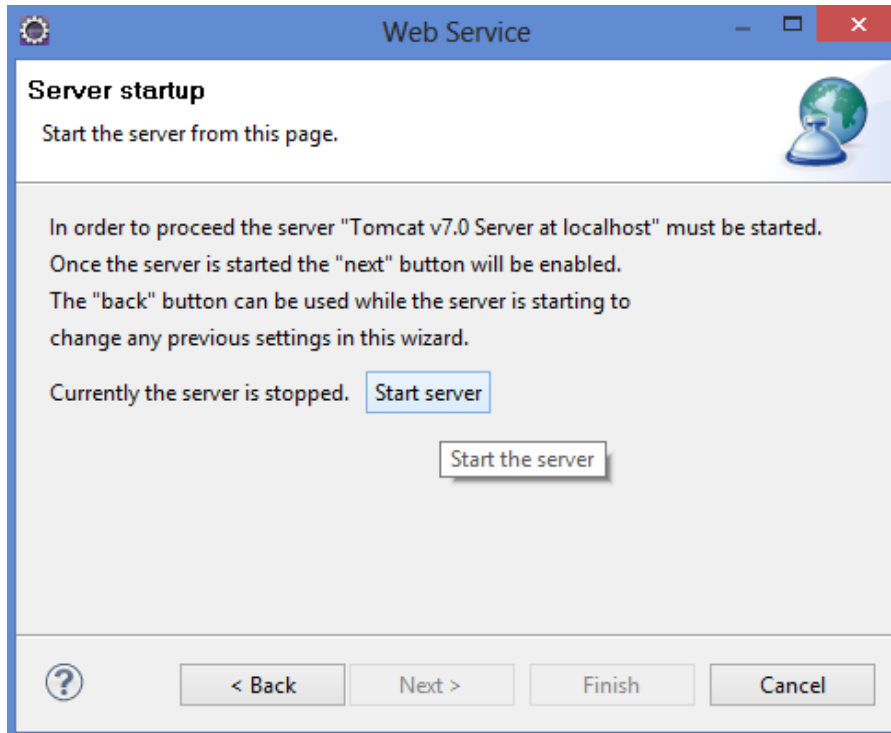
Customize your Web service

Axis2 Web Service Java Bean Configuration

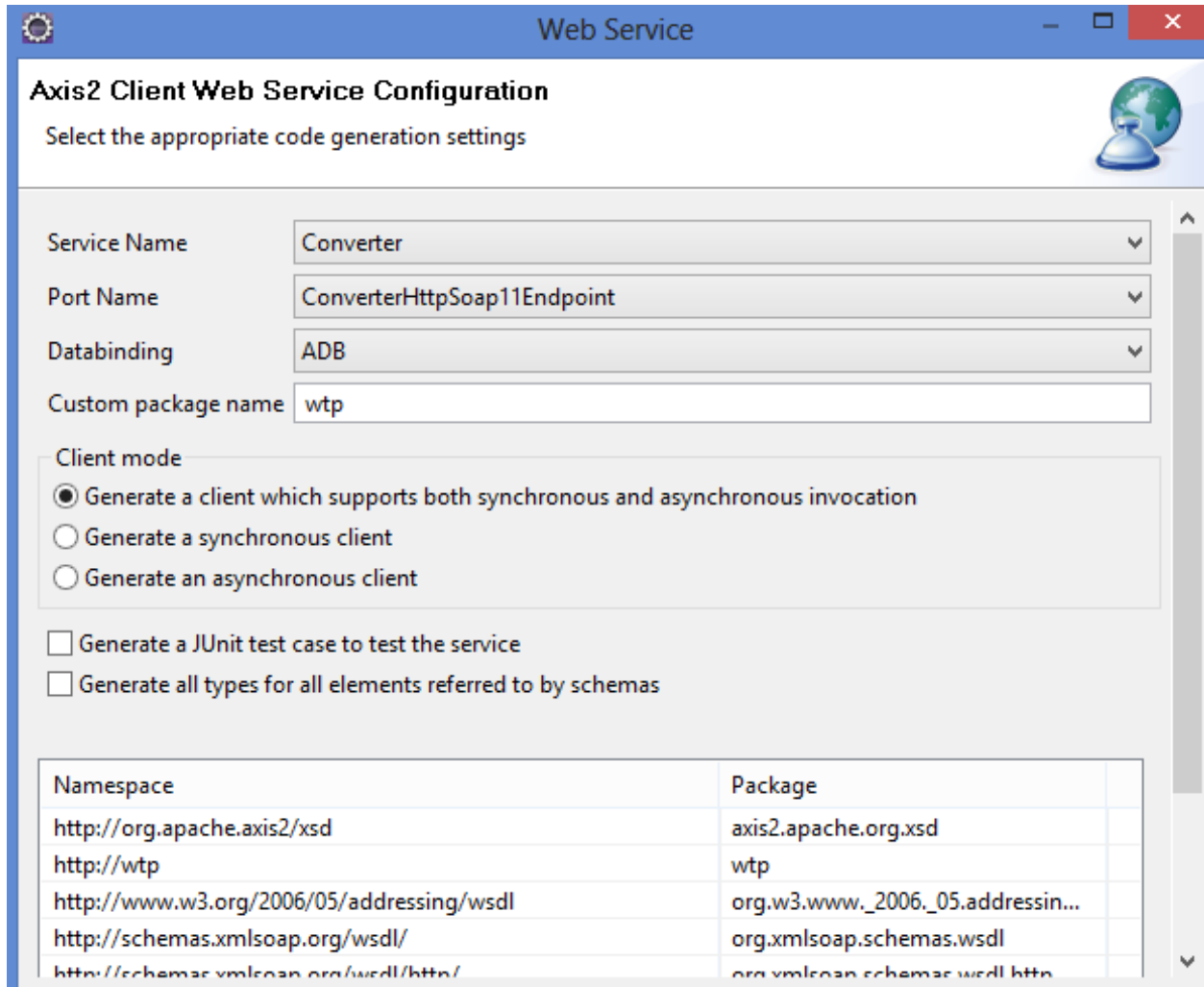
Use an existing services.xml file

Generate a default services.xml file

Eclipse > Creating a Bottom Up Web Service



Eclipse > Creating a Bottom Up Web Service



Axis2 Client Web Service Configuration
Select the appropriate code generation settings

Service Name: Converter

Port Name: ConverterHttpSoap11Endpoint

Databinding: ADB

Custom package name: wtp

Client mode

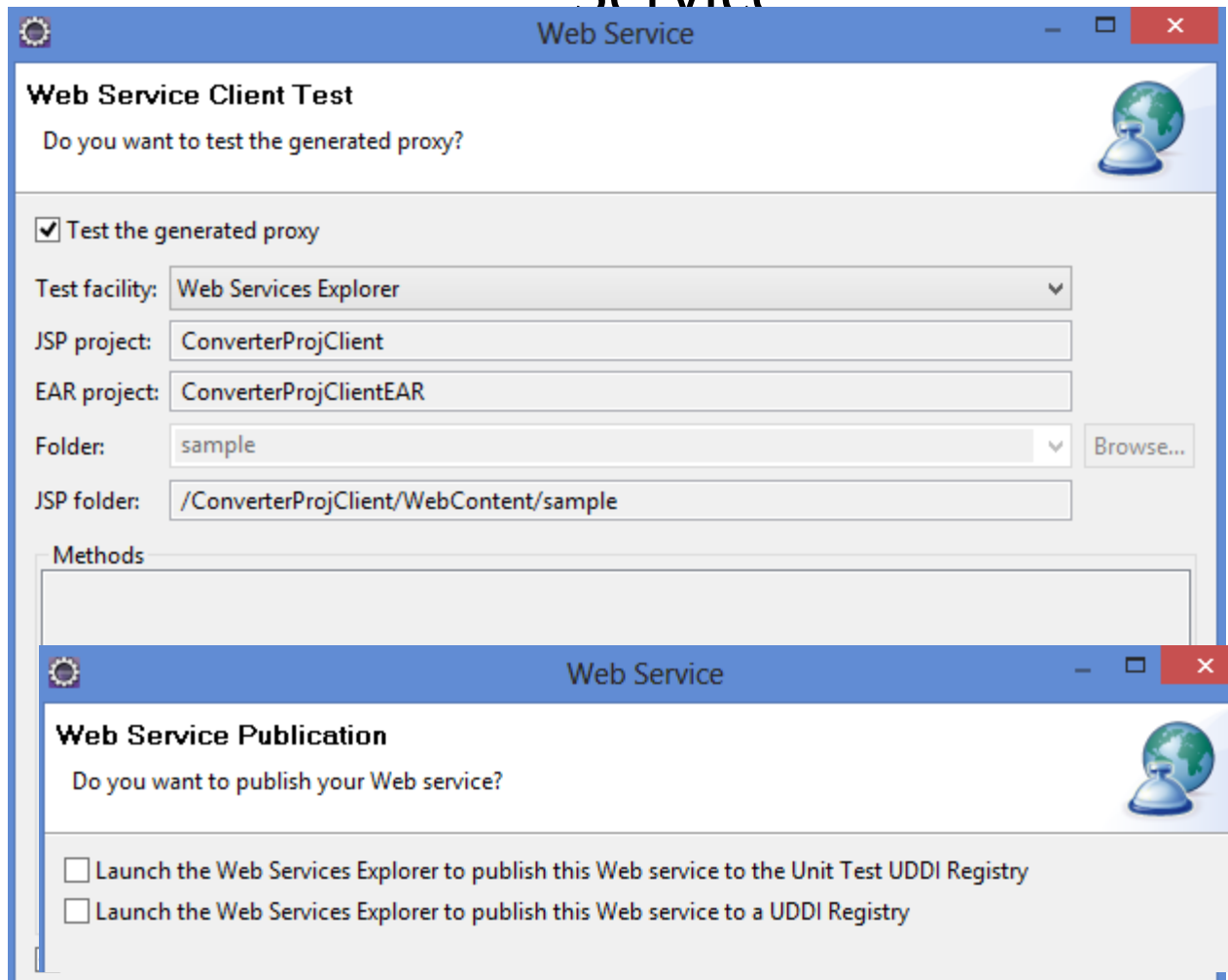
- Generate a client which supports both synchronous and asynchronous invocation
- Generate a synchronous client
- Generate an asynchronous client

Generate a JUnit test case to test the service

Generate all types for all elements referred to by schemas

Namespace	Package
http://org.apache.axis2/xsd	axis2.apache.org.xsd
http://wtp	wtp
http://www.w3.org/2006/05/addressing/wsdl	org.w3.www._2006._05.addressin...
http://schemas.xmlsoap.org/wsdl/	org.xmlsoap.schemas.wsdl
http://schemas.xmlsoap.org/wsdl/http/	org.xmlsoap.schemas.wsdl.http

Eclipse > Creating a Bottom Up Web Service



Eclipse > Creating a Bottom Up Web Service

The screenshot displays the Eclipse IDE interface for testing a web service. The 'Actions' window is active, showing the configuration for the 'celsiusToFahrenheit' operation. The endpoint is set to 'http://localhost:7177/ConverterProj/services/Converter.ConverterHttpSoap11Endpoint/'. The 'Body' section is expanded to show the 'celsius' parameter, which is a float type. A table with two columns, 'Values', is visible, with the value '37' entered in the first row. The 'Go' and 'Reset' buttons are located at the bottom of the 'Actions' window. The 'Status' window below shows the response for the 'celsiusToFahrenheitResponse' operation, indicating a return value of 98.6.

Actions

Enter the parameters for the WSDL operation "celsiusToFahrenheit" and click **Go** to invoke.

Endpoints

http://localhost:7177/ConverterProj/services/Converter.ConverterHttpSoap11Endpoint/

Body

celsiusToFahrenheit

celsius float [Add](#) [Remove](#)

	Values
<input type="checkbox"/>	
<input type="checkbox"/>	37

Go **Reset**

Status

celsiusToFahrenheitResponse

return (float): 98.6

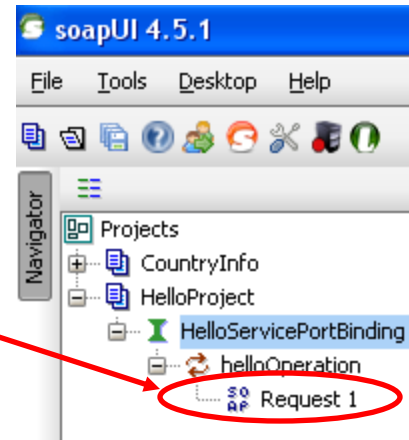
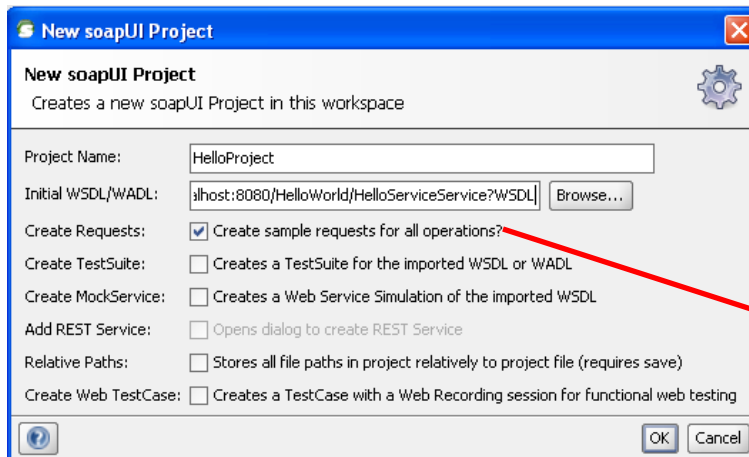
SoapUI

Soap UI tool

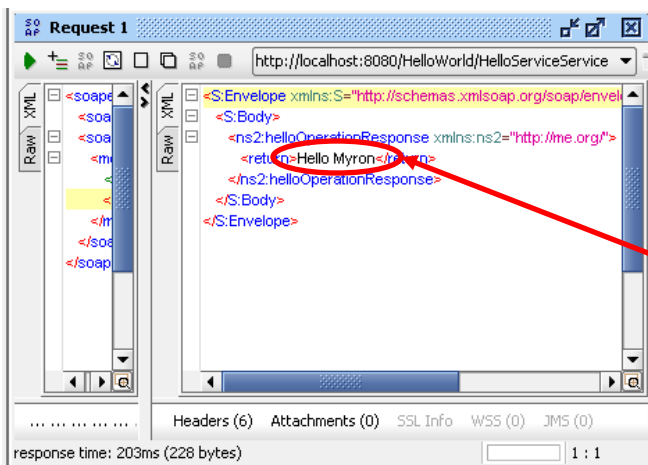
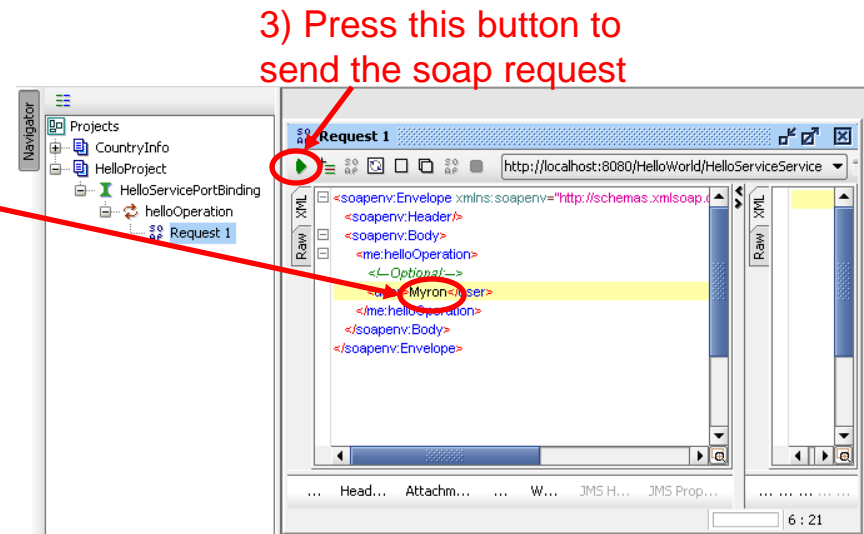
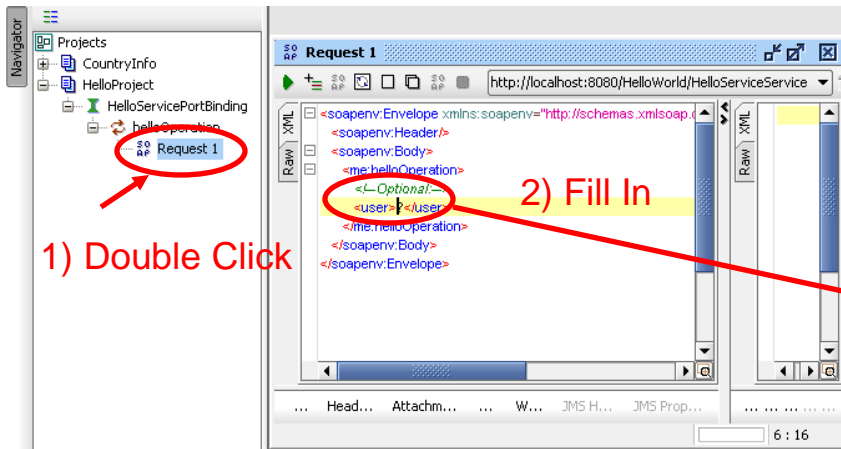
- <http://www.soapui.org/Working-with-soapUI/getting-started.html>
- Plugins for netbeans and eclipse also exist.
- Some slides from the 1st Assisting Lecture follow next..

SOAPUI Example 1 (1/2)

- Assume a web service “HelloService” that has a single operation which takes as input a single string from a user (e.g “Myron”) and returns a greeting message to the user (e.g “Hello Myron”)
- Assume that the WSDL of this web service is located in the following url: <http://localhost:8080/HelloWorld/HelloServiceService?WSDL>
- Invoke this web service using the SOAPUI tool
 - Create a new project (*File > New soapUI Project*), then set the project’s name and the url of the WSDL and press ok

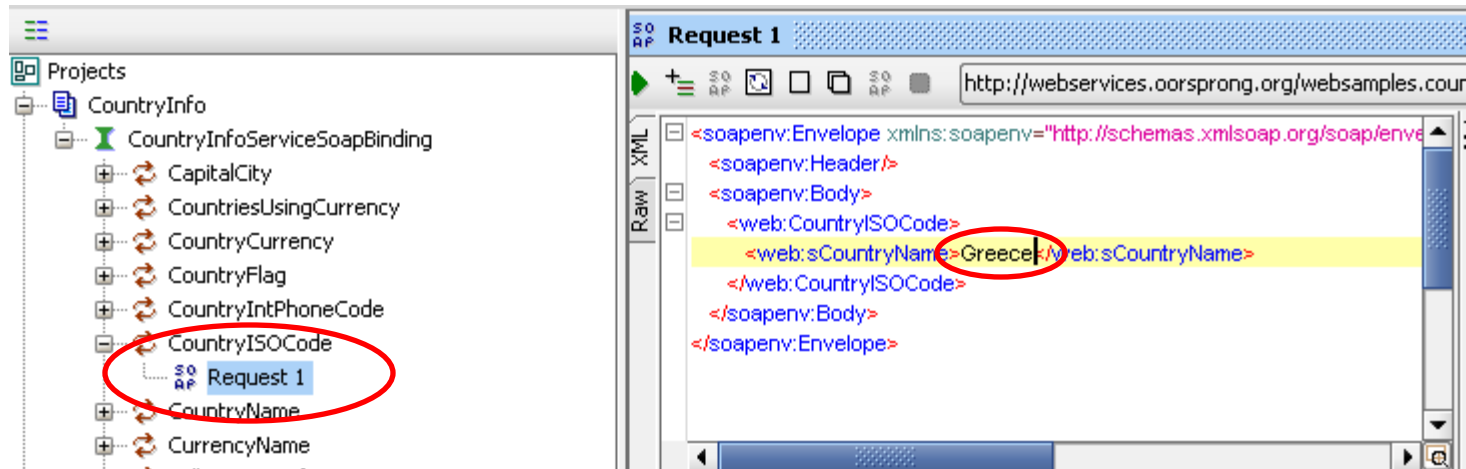


SOAPUI Example 1 (2/2)

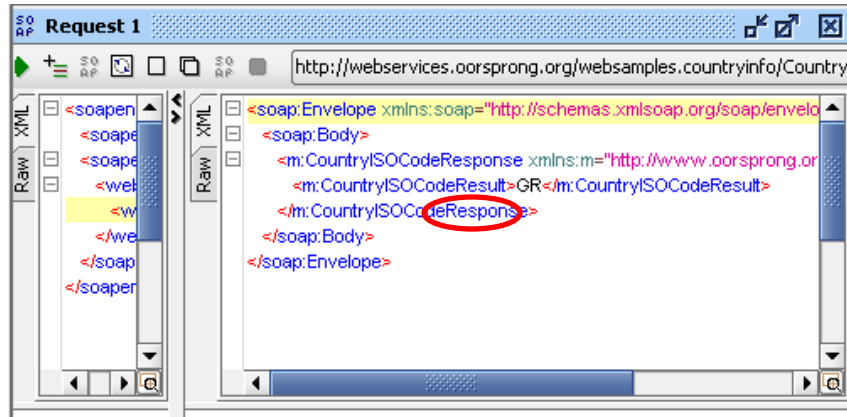


SOAPUI Example 2 (1/2)

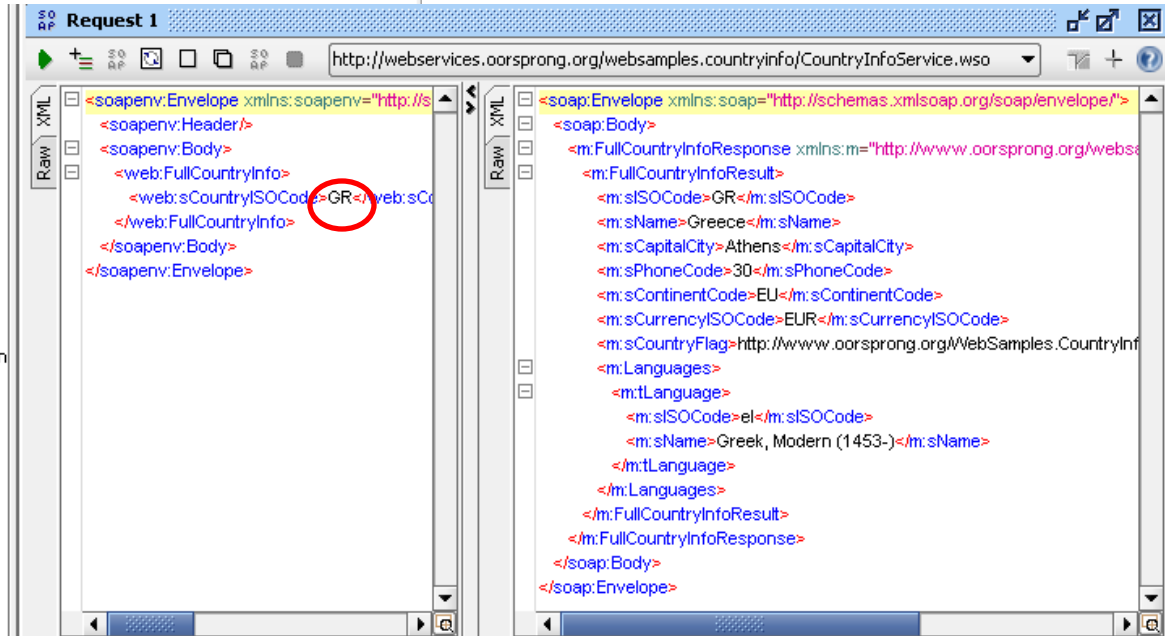
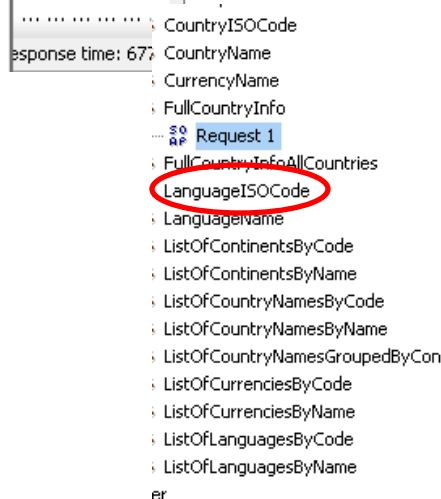
- Create a new project “CountryInfo” and set the WSDL to the following url:
 - <http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL>
 - This service provides several operations for retrieving information about a particular country
 - Find the iso code of “Greece” by sending a request (soap message)



SOAPUI Example 2 (2/2)



- The response shows that the iso code of Greece is “GR”
- Use this code to make a request to the operation “FullCountryInfo”



References

- **Bottom-up Approach:**

http://www.eclipse.org/webtools/community/tutorials/BottomUpAxis2WebService/bu_tutorial.html

- **Top-down Approach:**

http://www.eclipse.org/webtools/community/tutorials/TopDownAxis2WebService/td_tutorial.html

- <http://www.soapui.org/Working-with-soapUI/getting-started.html>

Τέλος Ενότητας



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο



ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «**Εκπαίδευση και Δια Βίου Μάθηση**» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



Σημειώματα

Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά Δημιουργού - Μη Εμπορική Χρήση - Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».



[1] <http://creativecommons.org/licenses/by-nc-nd/4.0/>

•Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

•Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Κρήτης, Μύρων Παπαδάκης. «**Εισαγωγή στα Δίκτυα Υπηρεσιών. Assisting Lecture 9 - Top Down SOAP Web Services**».

Έκδοση: 1.0. Ηράκλειο/Ρέθυμνο 2015. Διαθέσιμο από τη δικτυακή διεύθυνση:
<https://elearn.uoc.gr/course/view.php?id=416/>