



# Building Web Services with Apache Axis2

## Part I: Java-First (Bottom-Up) Services

Customized Java EE Training: <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



For live Java training, please see training courses at <http://courses.coreservlets.com/>. Servlets, JSP, Struts, JSF 1.x and JSF 2.0, Ajax, GWT, Java 5, Java 6, Spring, Hibernate, JPA, and customized combinations of topics.



Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization. Contact [hall@coreservlets.com](mailto:hall@coreservlets.com) for details.

# Agenda

- Overview of Web Services
- Installing Apache Axis2
- Making Java-First (Bottom-Up) Web Services
- Deploying and testing your services

5

© 2009 Marty Hall



## Overview of Web Services

Customized Java EE Training: <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

## What are Web Services? (Short Answer)

- **A Web Service is a resource that**
  - Is accessed via HTTP (or HTTPS)
  - Returns XML (SOAP)

7

## What are Web Services? (Long Answer from W3C)

- **A Web service is a software system**
  - identified by a URI
  - whose public interfaces and bindings are defined and described using XML.
  - Its definition can be discovered by other software systems.
    - These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.
- *Definition from*  
*<http://www.w3.org/TR/wsa-reqs/#id2604831>*

8

# Web Applications vs. Web Services

- **Web Apps**
  - Return HTML
  - Take GET or POST data as input
  - Result intended for a human (via a browser)
  - Informal (at best) description of data that resource accepts and result that resource returns
- **Web Services**
  - Return XML (SOAP)
  - Take XML (SOAP) as input
  - Result intended for a program
  - Formal definition of data that resource accepts and result that resource returns

9

# Web Service Advantages

- **Language neutrality**
  - Web Services define the message format, not the programming language used
  - A client does not know the language used by the service; a service does not know the language used by the client
- **Interoperability**
  - Definition of request and response data lets any Web service interact with any other.
- **Low barrier to entry**
  - Very simple tools for Java, Ruby, Microsoft languages, etc.
  - Can write services without knowing WSDL or SOAP!
- **Industry support**
  - Very widely adopted. No danger of investing in short-lived fad ala Ada and CORBA.

10

# Web Service Components

- **SOAP**
  - XML-based structure used to send and receive messages
  - Originally stood for Simple Object Access Protocol
    - Misleading acronym dropped in SOAP version 1.2
- **WSDL**
  - XML-based description of a Web service
    - Where it resides
    - What it can do
    - How to invoke it
  - Stands for Web Service Description Language
- **UDDI**
  - XML-based registry to list and find Web Services
  - Stands for Universal Description, Discovery and Integration

11

# Apache Axis2

- **Set of tools to simplify Web Services**
  - Services
    - Create a Web service from any Java class
    - Create Web service stubs from WSDL files
    - Build WAR file for deployment on any Java-based server
  - 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”
    - These tutorials will use Eclipse plugins
  - Command-line tools
    - Windows and Unix

12

# Uses of Axis2

- **Java-First (Bottom-Up) Services**
  - Start with a normal Java class (POJO)
  - Expose methods as Web Services
  - Covered in this lecture
- **Clients from Java-First Services**
  - Build client for service built with Java-first (bottom-up) approach
  - Covered in second lecture
- **WSDL-First (Top-Down) Services**
  - Start with a WSDL file
  - Create Java stubs automatically
  - Add business logic
  - Covered in third lecture
- **Clients from WSDL-First Services**
  - Build client for service built with WSDL-first (top-down) approach
  - Covered in fourth lecture

13

© 2009 Marty Hall



## Axis2 Setup

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

# Download and Install Axis2

## • Download

- Start at <http://ws.apache.org/axis2/download.cgi>
- Choose latest version
  - This tutorial uses 1.4
- Choose "zip" under Standard Binary Distribution

## • Install

- Unzip into directory of your choice
  - This tutorial uses C:\, resulting in C:\axis2-1.4

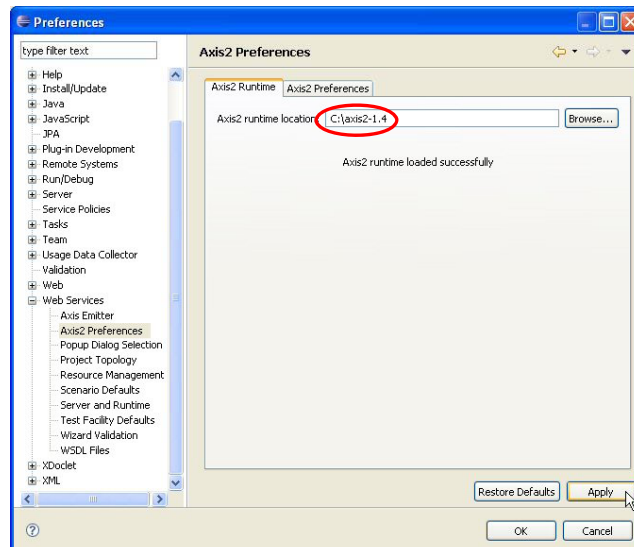


15

# Tell Eclipse about Axis2

## • Window → Preferences → Web Services → Axis2 Preferences

- For “Axis2 runtime location”, enter install path from previous slide
- Press Apply
- Press OK



16



# Building a Java-First (Bottom-Up) Service

Customized Java EE Training: <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

## Idea

- **Take normal Java class (POJO)**

```
public class Blah {  
    public int doFoo(String arg) { ... }  
    public String doBar(double arg) { ... }  
}
```

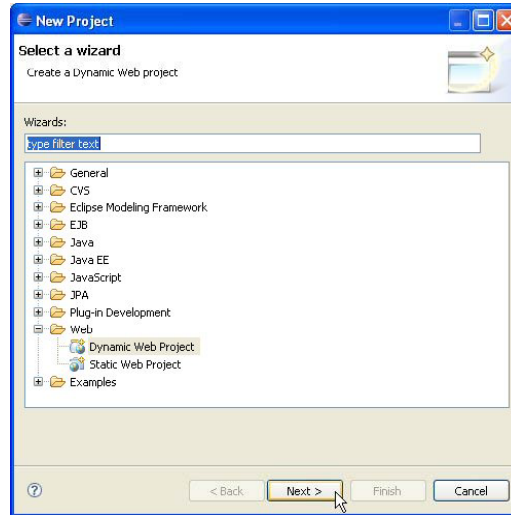
- No special interfaces, parent classes, packages, or method names

- **Automatically expose methods as Web Services**

- Blah becomes service name
- doFoo and doBar accessible via HTTP and SOAP

# Make Dynamic Web Project

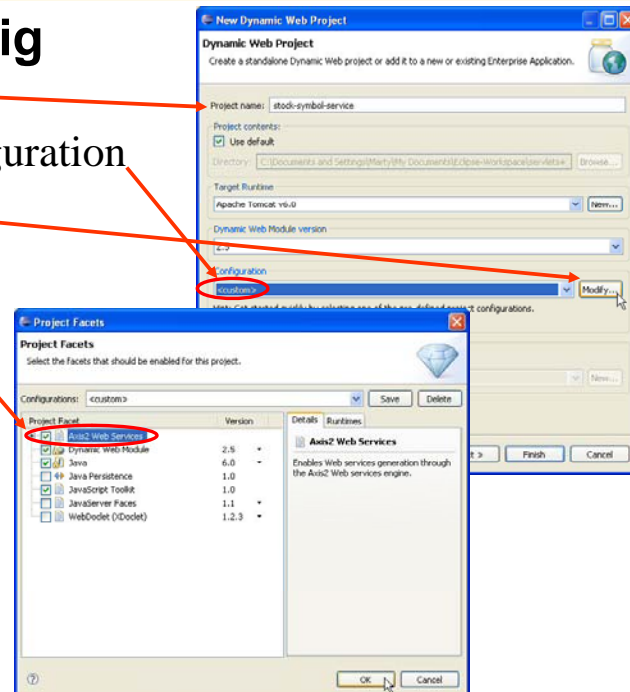
- **Make base project**
  - File → New → Project → Web → Dynamic Web Project
    - Press Next
- **Shortcut**
  - If you have previously made a Dynamic Web Project in that workspace, you can just do New → Dynamic Web Project



19

# Add Axis2 Support to Dynamic Web Project

- **Make custom config**
  - Enter a Project name
  - Choose custom configuration
  - Press Modify
- **Add Axis2 facet**
  - Select checkbox for Axis2 Web Services
  - Press OK
  - Press Finish on Dynamic Web Project screen



20

# Make Service Bean (Java Class with Regular Methods)

- **Create normal Java class**
  - No special requirements for
    - Package
    - Interfaces
    - Superclass
    - Names of methods (need not be getBlah and setBlah)
  - Public methods will later become part of Web Service
    - Added automatically when you make bottom-up Web Service
- **Lifecycle**
  - Class will be instantiated on each request
    - So, needs zero-argument constructor
    - Fields are not persistent unless they are static

21

## Service Bean: Example

```
public class StockSymbolService {
    public String findCompany(String symbol) {
        if ((isEmpty(symbol))) {
            return("Missing stock symbol");
        }
        String company = symbolMap.get(symbol.toUpperCase());
        if (company != null) {
            return(company);
        } else {
            return(String.format("Unknown symbol: %s.", symbol));
        }
    }

    public void updateCompany(String symbol, String company) {
        symbolMap.put(symbol.toUpperCase(), company);
    }
}
```

22

## Service Bean: Example (Continued)

```
private static String[][] companies =
    { { "AAPL", "Apple Inc." },
      { "IBM", "International Business Machines Corp." },
      { "JAVA", "Sun Microsystems Inc." },
      { "MSFT", "Microsoft Corp." },
      { "ORCL", "Oracle Corp." },
      { "RHT", "Red Hat Inc." }
    };

private static Map<String,String> symbolMap =
    new HashMap<String,String>();

static {
    for(String[] company: companies) {
        symbolMap.put(company[0], company[1]);
    }
}

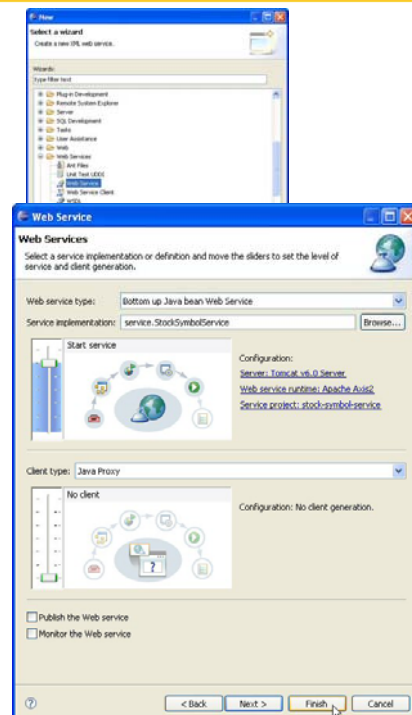
private boolean isEmpty(String val) {
    return((val == null) || val.trim().equals(""));
}
```

Class instantiated (via zero-arg constructor) on every request. So, persistent data should be static. If data does not change, making it static improves performance. But in this case, updateCompany changes the data, so it would not work at all if the Map was a non-static instance variable.

23

## Make Web Service from Service Bean

- **Use bottom-up service**
  - R-click StockSymbolService.java
  - New → Other → Web Services → Web Service → Press Next
    - You can also select StockSymbolService.java and do File → New → Other → Web Services → Web Service
  - Verify settings on Web Services wizard
    - Bottom up service
    - service.StockSymbolService as implementation class
    - Apache **Axis2** as service runtime
  - Press Finish
    - In future, you can deploy app normally (R-click server, Add and Remove Projects, select project, start server)



24

# Verify Service is Deployed

– `http://host/app-name/axis2-web/`

The image shows two overlapping browser windows. The left window displays the Axis 2 'Welcome!' page with a 'Services' link. The right window displays the 'List Services' page for 'StockSymbolService', showing its status as 'Active' and available operations: 'updateCompany' and 'findCompany'. A blue arrow points from the 'Services' link in the left window to the 'List Services' page in the right window.

Click Services. For your service, you should see a list of the public methods, now exposed as Web Service operations.

25

# Test Operations

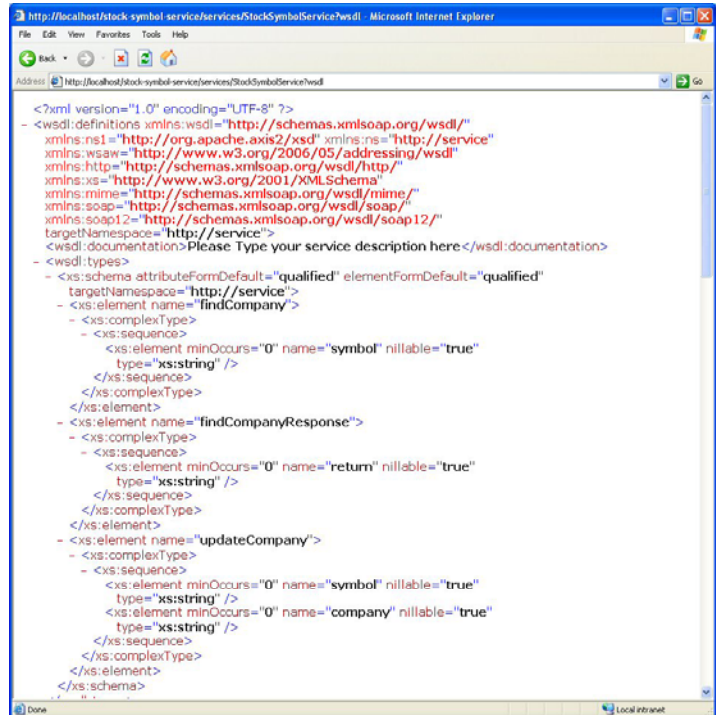
• `http://host/app-name/services/service-name/op-name?param=value`

The image shows three overlapping browser windows demonstrating test operations. The top window shows the 'findCompany?symbol=msft' endpoint returning XML: `<ns:findCompanyResponse xmlns:ns="http://service"><ns:return>Microsoft Corp.</ns:return></ns:findCompanyResponse>`. The middle window shows the 'updateCompany?symbol=msft&company=Yahoo' endpoint. The bottom window shows the 'findCompany?symbol=msft' endpoint returning XML: `<ns:findCompanyResponse xmlns:ns="http://service"><ns:return>Yahoo</ns:return></ns:findCompanyResponse>`.

26

# View WSDL

- `http://host/app-name/services/service-name?wsdl`
  - You can also click on service name in the listServices page shown earlier
- Save this URL
  - You will use it when you make a client (see next section)



```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:ns1="http://org.apache.axis2/xsd" xmlns:ns="http://service"
  xmlns:soap="http://www.w3.org/2006/05/addressing/soap"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  targetNamespace="http://service">
  <wsdl:documentation>Please. Type your service description here</wsdl:documentation>
  <wsdl:types>
    <xs:schema attributeFormDefault="qualified" elementFormDefault="qualified"
      targetNamespace="http://service">
      <xs:element name="findCompany">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" name="symbol" nillable="true"
              type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="findCompanyResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" name="return" nillable="true"
              type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="updateCompany">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" name="symbol" nillable="true"
              type="xs:string" />
            <xs:element minOccurs="0" name="company" nillable="true"
              type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>

```

27

# Summary

- **Make dynamic Web project**
  - File → New → Project → Web → Dynamic Web Project
- **Add Axis2 support**
  - Choose custom configuration, press Modify
  - Select checkbox for Axis2 Web Services
- **Create normal Java class**
  - No special requirements
- **Expose class as Web Service**
  - R-click Java class
  - New → Other → Web Services → Web Service
- **Deploy**
  - R-click server, Add and Remove Projects, select project, start server
- **Test**
  - `http://host/app-name/axis2-web/`

28



# Questions?

**Customized Java EE Training: <http://courses.coreservlets.com/>**

Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.