

# Judd Solutions

Development, Mentoring and Training



## Consuming and Producing Web Services with Web Tools

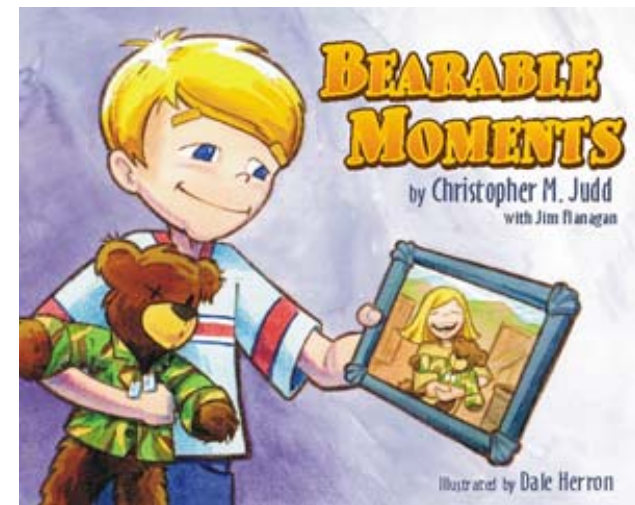
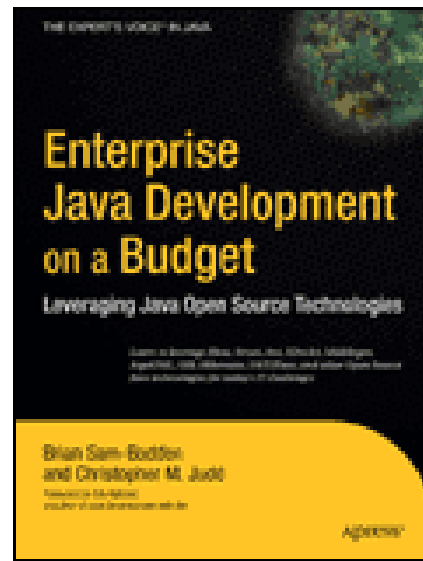
Christopher M. Judd  
President/Consultant  
Judd Solutions, LLC



# Christopher M. Judd



- President/Consultant of Judd Solutions
- Central Ohio Java User Group (COJUG) coordinator





# Other Sessions of Interest



- WTP
  - T-4. Develop Better J2EE Applications with the Web Tools Platform (Wed 8:45)
  - 101. Quick Tour of the Eclipse Web Tools Platform (Thu 8:30)
  - 107. Leveraging JSF Components (Thu 8:30)
  - 201. How to Build Java Web Applications with the Web Tools Platform (Thu 10:30)
  - 301. Facing JavaServer Faces Development with JSF Tools (Thu 1:15)
  - 401. Consuming and Producing Web Services with Web Tools (Thu 3:15)
  - 501. Developing Java Web Services with the Web Tools Platform (Fri 8:45)
  - 504. Developing Rich Applications with JSF and AJAX (Fri 8:45)
  - 701. Building Applications with the Java Persistence API and Dali (Fri 1:45)
  - 706. Step by Step: Making Enterprise JavaBeans with J2EE Standard Tools (Fri 1:45)
  - 801. How to Use and Extend Eclipse's XML and Schema Tools (Fri 3:45)
- Other
  - T-6. Implementing SOA in Eclipse (Wed 8:45)
  - T-7. Callisto Boot Camp: Ten Projects. One Day (Wed 8:45)
  - 106. How to Improve Database Connectivity with the Data Tools Platform (Thu 8:30)
  - 107. Leveraging JSF Components (Thu 8:30)
  - 202. Web 2.0 the Eclipse Way with the Rich AJAX Platform (Thu 10:30)
  - 204. Interacting with Relational Databases (Thu 10:30)
  - 406. Developing and Deploying Services using the SOA Tools Platform (Thu 406)
  - 607. Advanced User Interface Programming Using the Eclipse Rich Client Platform (Fri 10:45)



# Agenda



- WTP Overview
- Web Services Overview
- Consuming
- Producing Web Services (bottom up)
- Testing
- Deployment
- Conclusion



# Web Tools Platform Project (WTP)



- Top Level Eclipse Project
- [www.eclipse.com/webtools/](http://www.eclipse.com/webtools/)
- Included in Callisto release
- Development tools for web and JEE development
  - No runtime dependencies
  - Vendor extensible
- Subprojects
  - Web Standard Tools (WST)
  - J2EE Standard Tools (JST)
  - JavaServer Faces Tools (JSF)
- Current version 1.5
- Dependencies
  - Eclipse 3. 2
  - Eclipse Modeling Framework (EMF) 2.2.0
  - Graphic Editor Framework (GEF) 3.2
  - Java EMF Model (JEM) 1.2



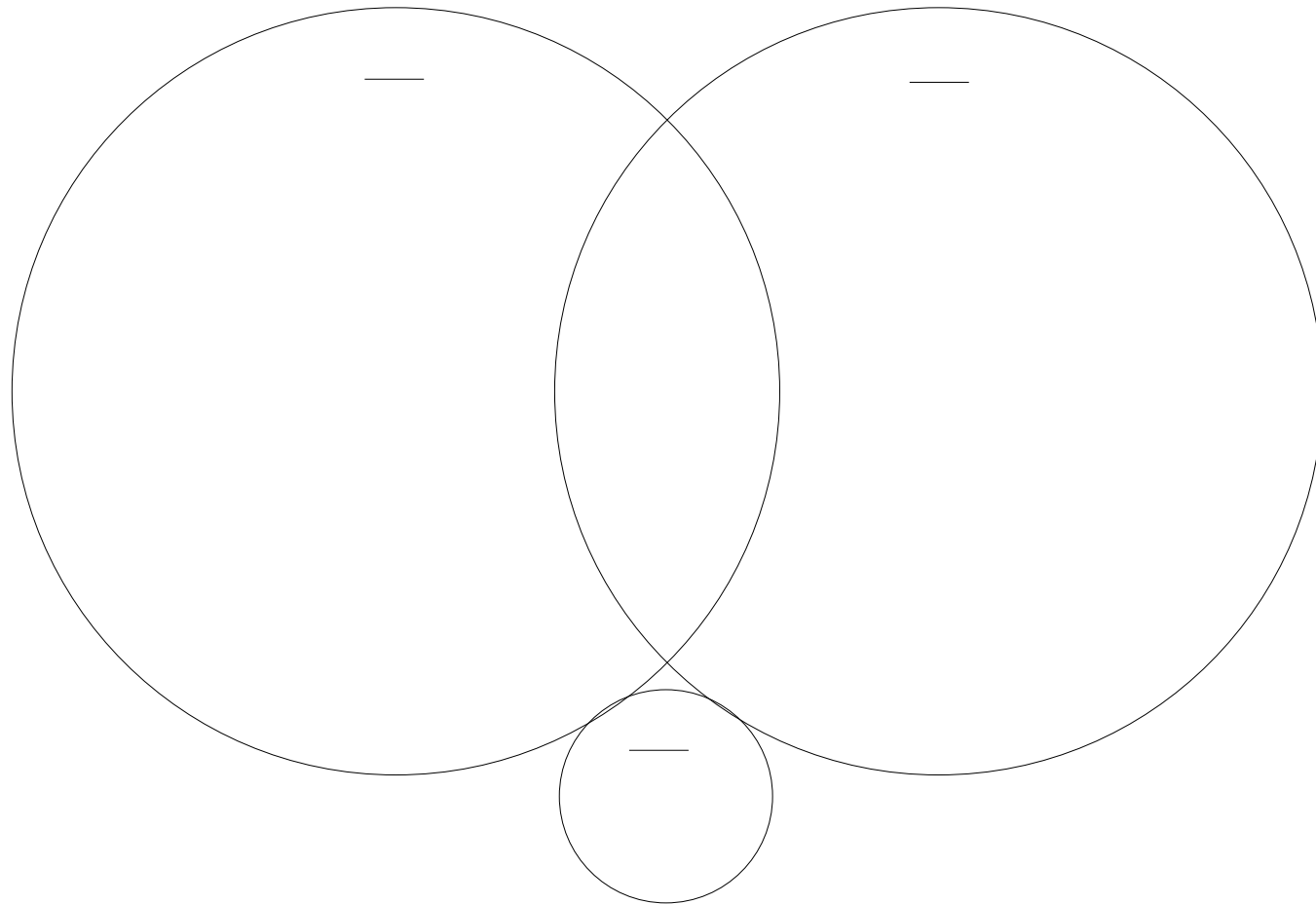
# WTP Subprojects



- Web Standard Tools (WST)
  - Web artifacts
  - Defined by open standards bodies
- J2EE Standard Tools (JST)
  - J2EE components
  - Java Community Process (JCP)
  - Depends on WST
- JavaServer Faces Tools (JSF)



# WTP Scope



Struts

Judd Solutions



# WTP Web Services Scope



- World Wide Web Consortium (W3C)
  - <http://www.w3.org>
  - HTML, XHTML, CSS, XML, XSLT, XML Schema, XML Query
- Organizations for Advancement of Structured Information Standards (OASIS)
  - <http://www.oasis-open.org>
  - e-Business standards for web services
- Web Services Interoperability Organizations (WS-I)
  - <http://www.ws-i.org>
  - Interoperable message exchange between web services
- Java Community Process (JCP)
  - <http://www.jcp.org>
  - Java Web Services APIs





# WTP Installation Options



- Callisto
- All-in-one
  - Eclipse 3.2
  - Eclipse Modeling Framework (EMF) 2.2.0
  - Graphic Editor Framework (GEF) 3.2
  - Java EMF Model (JEM) 1.2
- Update Manager
  - Eclipse.org update site
- Piecemeal

\* JSF is a separate install



# Agenda



- WTP Overview
- Web Services Overview
- Consuming
- Producing Web Services (bottom up)
- Testing
- Deployment
- Conclusion



# Simple Object Access Protocol



From the draft W3C specification:

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.

<http://www.w3.org/TR/soap/>



# WSDL



- Web Service Definition Language
- Describes
  - What the service can do
  - Where it resides
  - How to invoke it
- Elements
  - Types – data type definition
  - Message – definition of data being communicated
  - Port Type – abstract set of operations
  - Binding – concrete protocol and data format
  - Service – collection of related endpoints
  - Port – binding and a network address



# Apache Axis



- Apache Web Services Project
- Open Source
- SOAP implementation
- Version 1.3.0
- <http://ws.apache.org/axis/>



# Agenda



- WTP Overview
- Web Services Overview
- Consuming
- Producing Web Services (bottom up)
- Testing
- Deployment
- Conclusion



# Weather Forecast Web Service



- Forecasting weather by zip code
  - 7 day forecast
  - Place name
  - State code
  - Latitude and Longitude
- Found at X Methods ([www.xmethods.net](http://www.xmethods.net))
- Service description
  - [www.webservicex.net/WeatherForecast.aspx](http://www.webservicex.net/WeatherForecast.aspx)

Zip code:

Place Names: WORTHINGTON  
State Code: OH  
Area Code: 614  
Time Zone: EST  
Latitude: 40.105156  
Longitude: 83.01007  
Allocation Factor: 0.00251  
FIPS Code: 39

7 Day Weather Forecast:

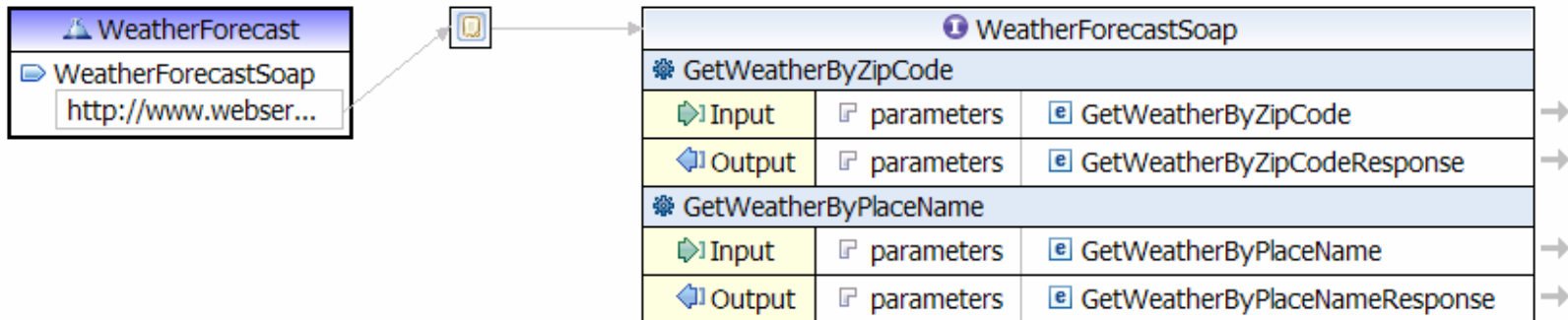
Saturday, July 22, 2006		80°F/57°F 27°C/14°C
Sunday, July 23, 2006		86°F/61°F 30°C/16°C
Monday, July 24, 2006		87°F/65°F 31°C/18°C
Tuesday, July 25, 2006		87°F/68°F 31°C/20°C
Wednesday, July 26, 2006		85°F/68°F 29°C/20°C
Thursday, July 27, 2006		85°F/68°F 29°C/20°C 30%
Friday, July 28, 2006		85°F/68°F 29°C/20°C 30%



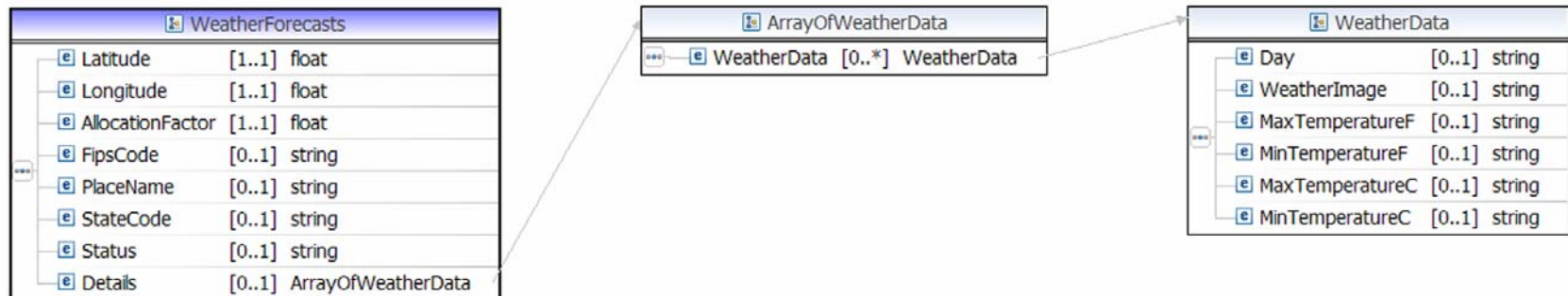
# Weather Forecast Web Service



## Service description



## Data transfer objects







# Consuming Web Services Strategy



A common anti-pattern is to generate the web service client stub code directly into an application. Instead generate the code in its own project, export it as a jar and manage it as a jar.

- Less files to maintain
- Improves reuse of client code
- Generated code does not follow code quality standards



# Consuming Web Services Steps



1. Create Java Project
2. Download WSDL
3. Generate Web Service Client from WSDL
4. Export client Jar



# Create Project



- Create a standard Java Project
- File > New > Project > Java Project

The screenshot shows the 'New Java Project' dialog box. The title bar reads 'New Java Project'. Below the title bar, the main heading is 'Create a Java project' with a subtext 'Create a Java project in the workspace or in an external location.' and a folder icon. The 'Project name' field contains 'weatherforecast-ws-client'. The 'Contents' section has two radio buttons: 'Create new project in workspace' (selected) and 'Create project from existing source'. Below this is a 'Directory' field with the path 'E:\workspaces\eclipseworld\weatherforecast-1' and a 'Browse...' button. The 'JRE' section has two radio buttons: 'Use default JRE (Currently 'jre1.5.0\_06')' (selected) and 'Use a project specific JRE: jre1.5.0\_06' (with a dropdown arrow). A 'Configure JREs...' link is to the right. The 'Project layout' section has two radio buttons: 'Use project folder as root for sources and class files' (selected) and 'Create separate source and output folders'. A 'Configure default...' link is to the right. At the bottom, there are buttons for '< Back', 'Next >', 'Finish', and 'Cancel', along with a help icon (?) on the left.



## Download WSDL



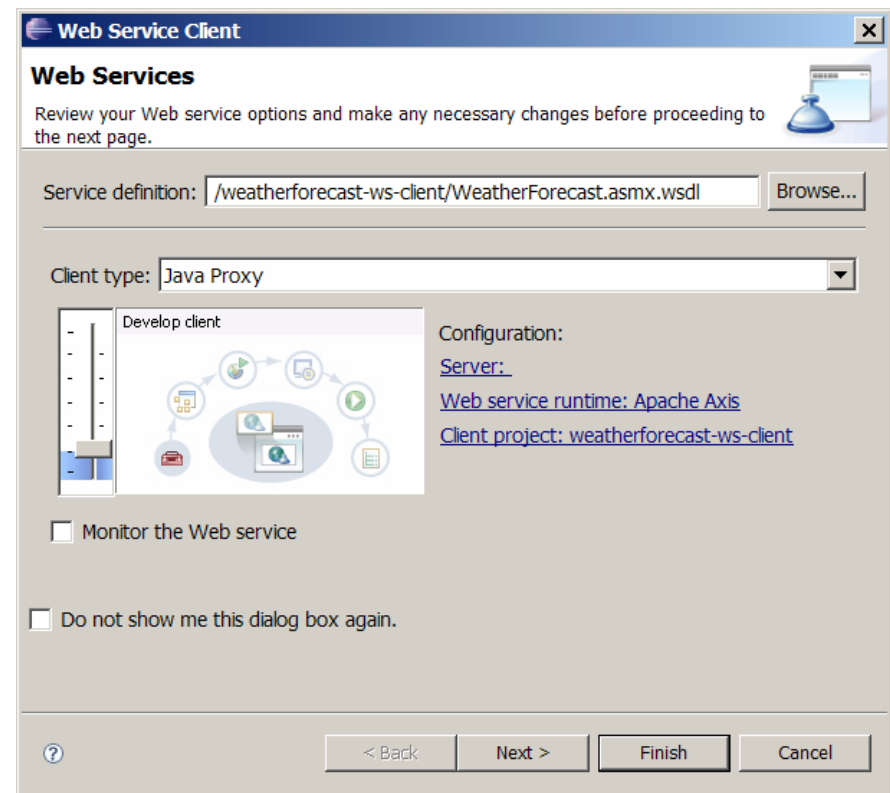
- Download WSDL
- Add WSDL to project
- Include it in the exported Jar file
- Provides traceability incase there is a need to know which WSDL version was used to generate stubs and Jar



# Generate Web Service Client



- Right click on WSDL and choose Web Services > Generate Client  
or  
File > New > Web Services > Web Service Client
- Specify Java Proxy
- Monitor traffic using TCP/IP Monitor

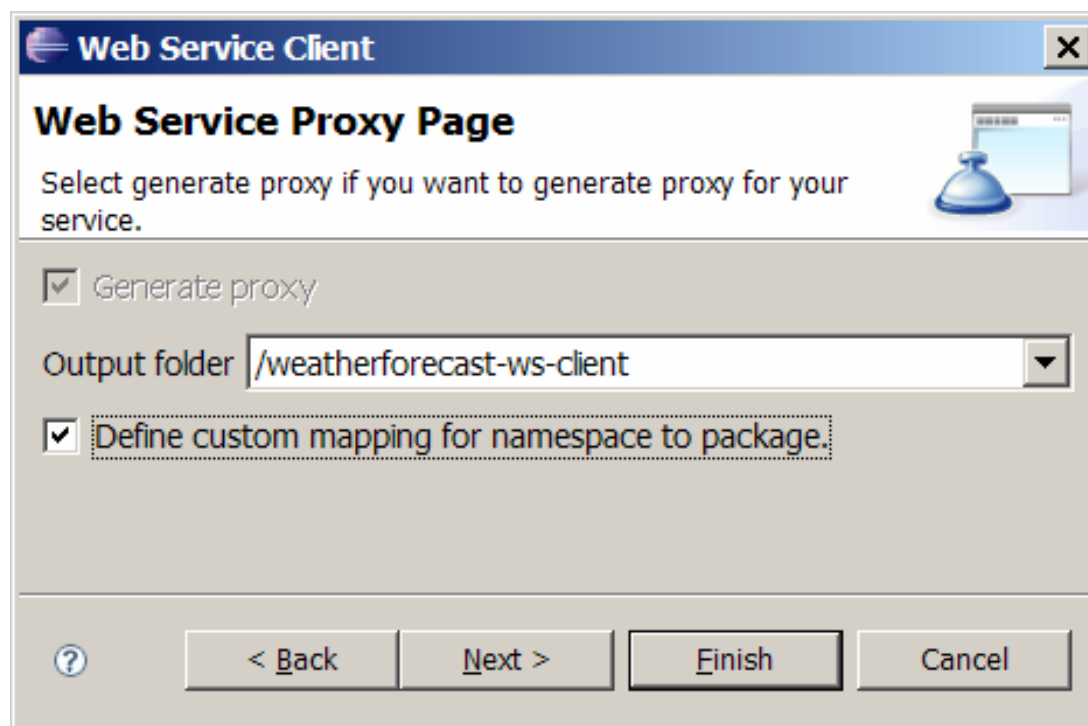




# Generate Web Service Client



- Specify project to generate the code in



The image shows a Windows-style dialog box titled "Web Service Client". The main heading is "Web Service Proxy Page". Below the heading, there is a text instruction: "Select generate proxy if you want to generate proxy for your service." To the right of this text is a small icon of a computer monitor with a blue bell in front of it. There are three checked checkboxes: "Generate proxy", "Define custom mapping for namespace to package.", and "Output folder" (which is a dropdown menu). The dropdown menu currently shows the path "/weatherforecast-ws-client". At the bottom of the dialog, there are four buttons: a help button (question mark in a circle), "< Back", "Next >", and "Finish". A "Cancel" button is also present at the bottom right.



# Generate Web Service Client



- Optionally, can specify namespace to package mapping
- Convert targetNamespace to a more conventional Java package naming convention.

The dialog box is titled "Web Service Client" and "Web Service Client Namespace to Package Mapping". It contains a table with two columns: "namespace" and "package". The first row has the values "http://www.webservices.net" and "net.webservices.weather". Below the table are buttons for "Import...", "Add", and "Remove". At the bottom are buttons for "< Back", "Next >", "Finish", and "Cancel".

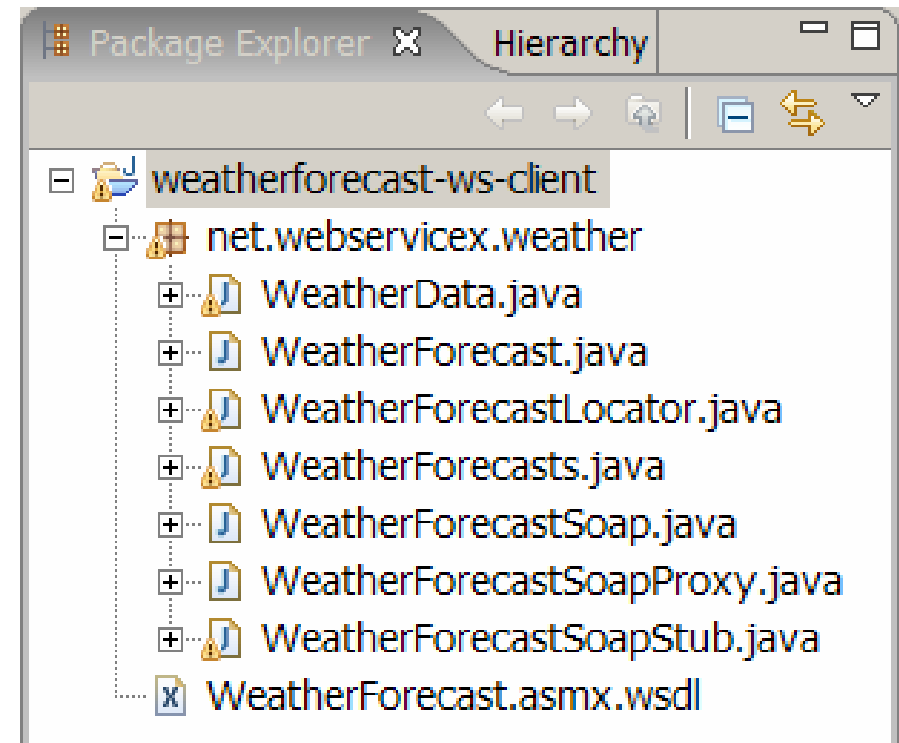
namespace	package
http://www.webservices.net	net.webservices.weather



# Output



- Classes
  - Service interface
  - Dynamic proxy service impl
  - Stub service impl
  - Service locator interface
  - Service locator impl
  - Data transfer objects
- Axis Jars
  - axis.jar
  - jaxrpc.jar
  - saaj.jar
  - wsdl4j-1.5.1.jar
  - commons-discovery-0.2.jar
  - commons-logging-1.0.4.jar



\* When using the client Jar, these Axis jars will be required as well.





# Using Generate Classes



```
WeatherForecastLocator forecastLocator = new WeatherForecastLocator();  
WeatherForecastSoap wfSoap = forecastLocator.getWeatherForecastSoap();  
WeatherForecasts forecasts = wfSoap.getWeatherByZipCode(zipCode);  
WeatherData[] details = forecasts.getDetails();
```

```
System.out.println(forecasts.getLatitude());  
System.out.println(forecasts.getLongitude());
```

```
for (int i = 0; i < details.length; i++) {  
    WeatherData detail = details[i];  
    System.out.println(detail.getDay());  
    System.out.println(detail.getWeatherImage());  
    System.out.println(detail.getMaxTemperatureF());  
}
```



# Export client Jar



**JAR Export**

**JAR File Specification**

Define which resources should be exported into the JAR.

Select the resources to export:

<input checked="" type="checkbox"/> weatherforecast-ws-client	<input type="checkbox"/> .classpath
	<input type="checkbox"/> .project
	<input checked="" type="checkbox"/> readme.txt
	<input checked="" type="checkbox"/> WeatherForecast.asmx.wsdl

Export generated class files and resources  
 Export all output folders for checked projects  
 Export java source files and resources  
 Export refactorings for checked projects. [Select refactorings...](#)

Select the export destination:

JAR file:

Options:

Compress the contents of the JAR file  
 Add directory entries  
 Overwrite existing files without warning



# Agenda



- WTP Overview
- Web Services Overview
- Consuming
- Producing Web Services (bottom up)
- Testing
- Deployment
- Conclusion



# Producing Web Services



- Bottom up
  - Java code -> Generate WSDL
- Top down
  - WSDL -> Generate Java code



## Bottom Up Steps



1. Create Dynamic Web Project
2. Create Service and DTOs
3. Generate Web Service



# Create Dynamic Web Project



- File > New > Other > Web > Dynamic Web Project
- Project Name
- Target server

The screenshot shows the 'New Dynamic Web Project' dialog box in the Eclipse IDE. The dialog is titled 'New Dynamic Web Project' and contains the following fields and options:

- Project name:** A text field containing 'order-ws-server'.
- Project contents:** A section with a checked checkbox for 'Use default' and a 'Directory:' field containing 'E:\workspaces\eclipseworld\order-ws-server' with a 'Browse...' button.
- Target Runtime:** A dropdown menu showing 'Apache Tomcat v5.5' and a 'New...' button.
- Configurations:** A dropdown menu showing '<custom>' and a hint: 'Hint: Get started quickly by selecting one of the pre-defined project configurations.'
- EAR Membership:** A section with an unchecked checkbox for 'Add project to an EAR' and an 'EAR Project Name:' field containing 'order-ws-serverEAR' with a 'New...' button.

At the bottom of the dialog, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.



# Create Dynamic Web Project



- Facets and versions

The screenshot shows the 'New Dynamic Web Project' dialog box. The 'Project Facets' section is active, showing a list of facets and their versions. The 'Configurations' dropdown is set to '<custom>'. The 'Project Facet' list includes 'Dynamic Web Module' (checked, version 2.4 ...), 'Java' (checked, version 5.0 ...), 'JavaServer Faces' (unchecked, version 1.1), and 'WebDoclet (XDoclet)' (unchecked, version 1.2.3 ...). A '<< Show Runtimes' button is located below the list. At the bottom of the dialog are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

Project Facet	Version
<input checked="" type="checkbox"/> Dynamic Web Module	2.4 ...
<input checked="" type="checkbox"/> Java	5.0 ...
<input type="checkbox"/> JavaServer Faces	1.1
<input type="checkbox"/> WebDoclet (XDoclet)	1.2.3 ...



# Create Dynamic Web Project



- Context Root
- Directories

The image shows a 'New Dynamic Web Project' dialog box with the following fields and controls:

- Web Module**: Configure web module settings. (Icon: folder with globe)
- Context Root:**
- Content Directory:**
- Java Source Directory:**
- Buttons:
- Help icon:  (bottom left)





# Create Service



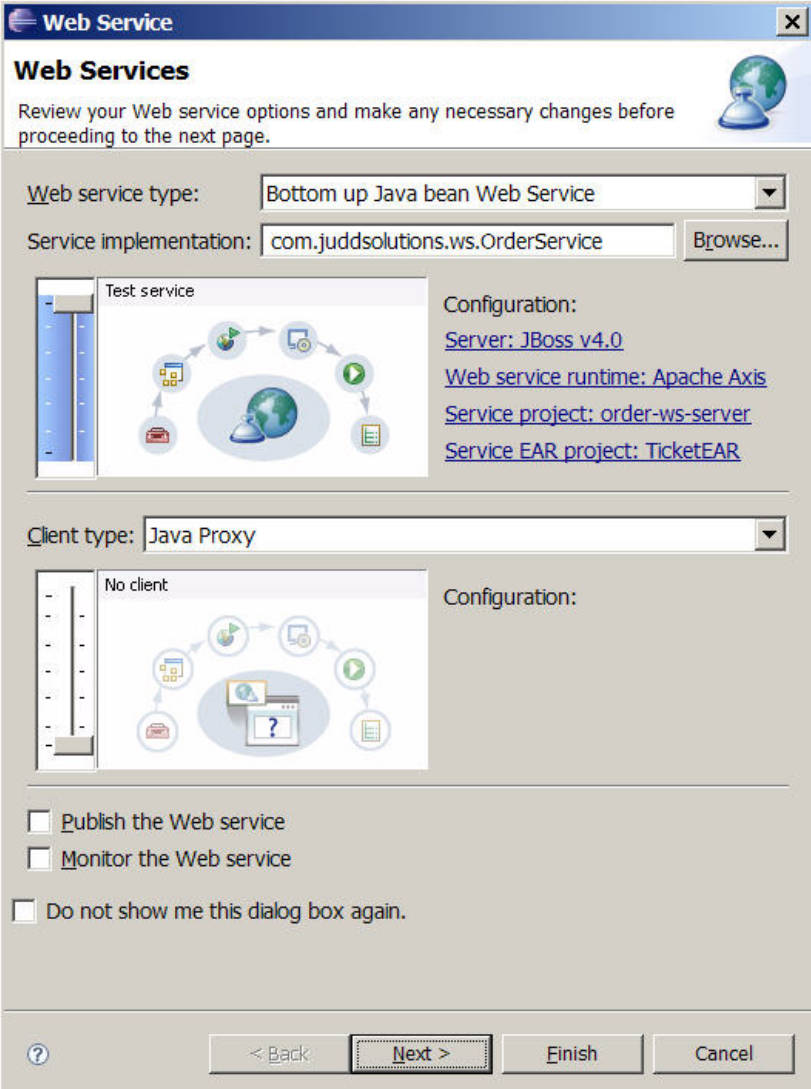
```
public class OrderService {  
  
    private static final float DISCOUNT = 0.9f;  
  
    public float quote(float price, int quantity) {  
        if (quantity > 100) {  
            price = price * DISCOUNT;  
        }  
        return price * quantity;  
    }  
}
```



# Generate Web Service



- New > File > Other > Web Services > Web Service or  
Right click on Class: Web Services > Create Web Services
- Select the amount to be generated
- Monitor using TCP/IP Monitor



The image shows a 'Web Service' dialog box with the following fields and options:

- Web Services** (Title bar)
- Review your Web service options and make any necessary changes before proceeding to the next page.
- Web service type:** Bottom up Java bean Web Service
- Service implementation:** com.juddsolutions.ws.OrderService (with a Browse... button)
- Test service:** A diagram showing a service being tested by a client.
- Configuration:**
  - Server: [JBoss v4.0](#)
  - Web service runtime: [Apache Axis](#)
  - Service project: [order-ws-server](#)
  - Service EAR project: [TicketEAR](#)
- Client type:** Java Proxy
- No client:** A diagram showing a service without a client.
- Configuration:** (Empty)
- Publish the Web service
- Monitor the Web service
- Do not show me this dialog box again.
- Buttons: << Back, Next >, Finish, Cancel



# Generate Web Service



- Select methods to expose
- Select type
  - RPC
    - Performance
    - Simple types
  - Document
    - Interoperability
    - XML

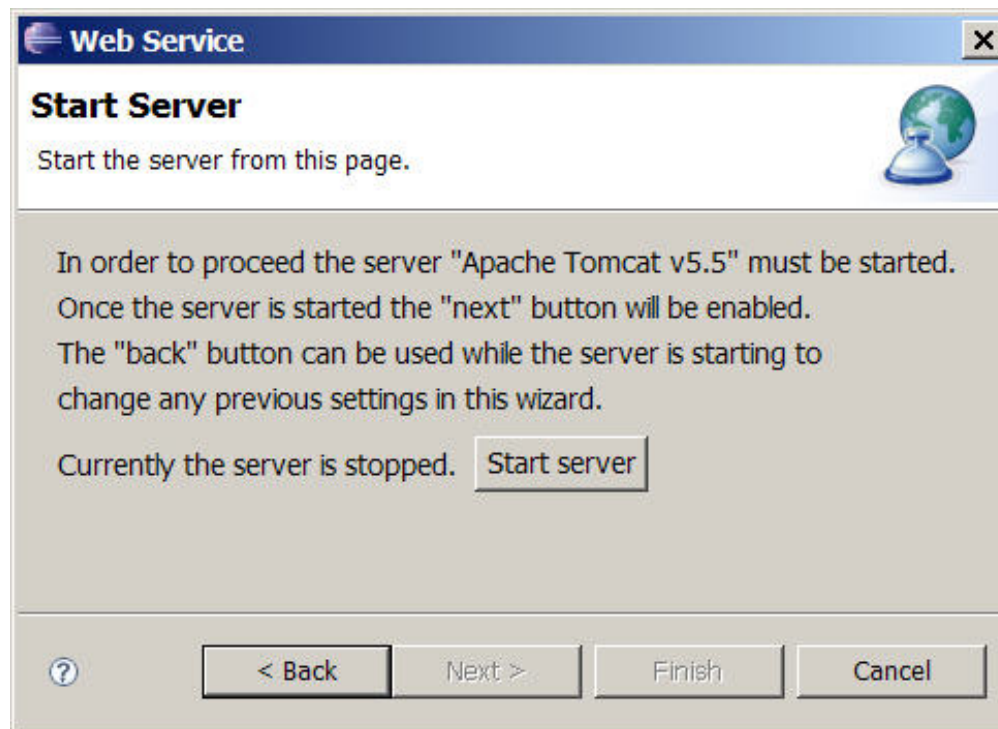
The image shows a dialog box titled "Web Service Java Bean Identity" with the subtitle "Configure the Java bean as a Web service." The dialog has a "WSDL file:" field containing "OrderService.wsdl". Below this is a "Methods" list with a single entry "quote(float,int)" which is checked. There are "Select All" and "Deselect All" buttons. Under "Style and use", there are three radio buttons: "document/literal (wrapped)", "document/literal", and "RPC/encoded" (which is selected). There is also a checkbox for "Define custom mapping for package to namespace." At the bottom, there are buttons for "< Back", "Next >", "Finish", and "Cancel".



# Generate Web Service



- Start server to local application service

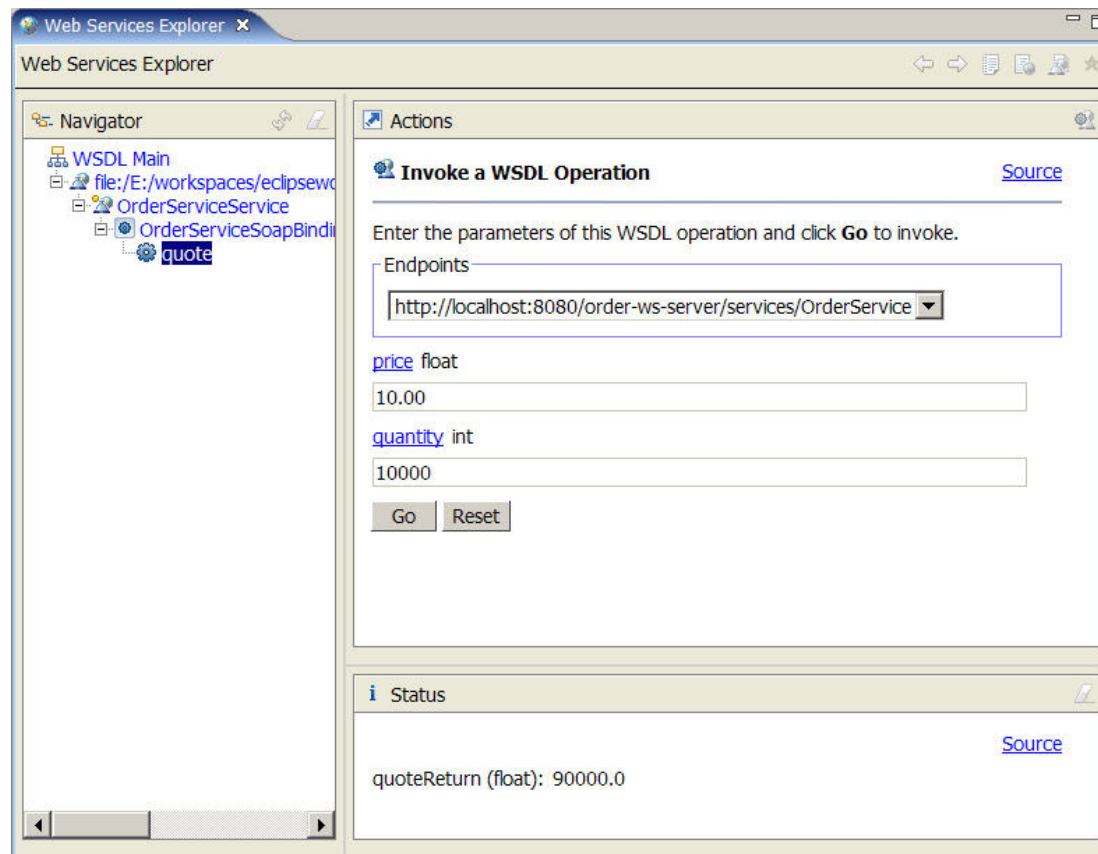
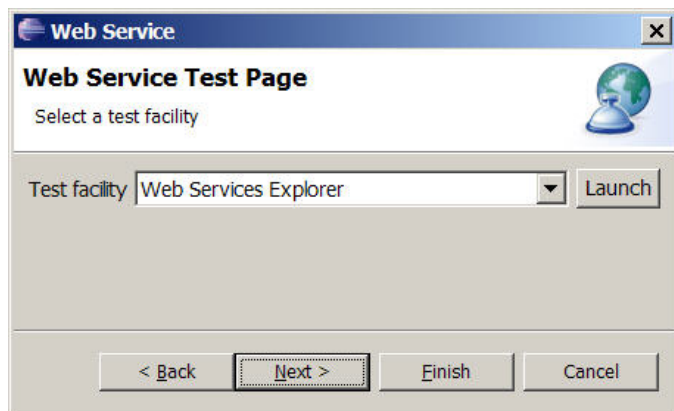




# Generate Web Service



- Test Client





# Generate Web Service



- Publish web service to UDDI registry

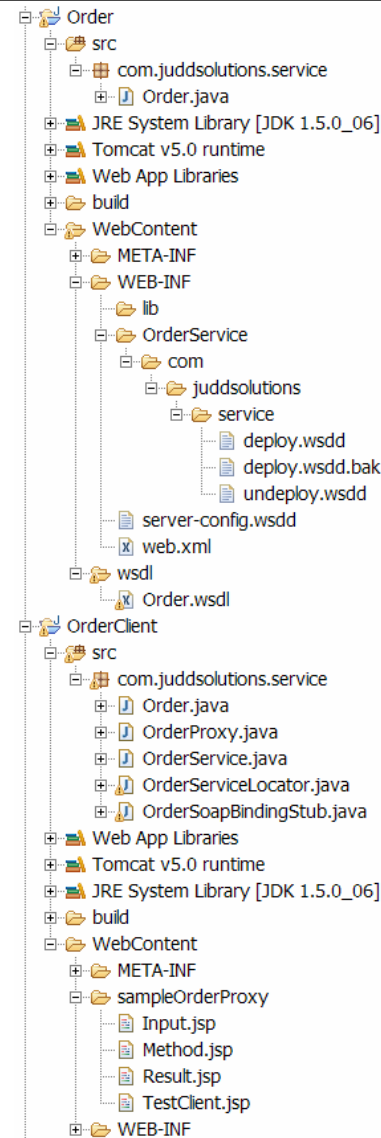
The image shows a Windows-style dialog box titled "Web Service". The main heading is "Web Service Publication" with a globe icon. Below the heading is the question "Do you want to publish your Web service?". There are two checkboxes: the first is "Launch the Web Services Explorer to publish this Web service to the Unit Test UDDI Registry" and the second is "Launch the Web Services Explorer to publish this Web service to a UDDI Registry". Below these is a label "Public UDDI Registry" followed by a dropdown menu currently showing "SAP UDDI Test Registry". At the bottom, there are four buttons: a help button (question mark), "< Back", "Next >", "Finish", and "Cancel".



# Output



- Web Service
  - Axis deployment descriptors
  - WSDL
- Test Client
  - Dynamic web app
  - Proxy
  - JSPs





# Data Types



- Simple Java data types
- JavaBeans
  - No parameter constructor
- Arrays





# Agenda



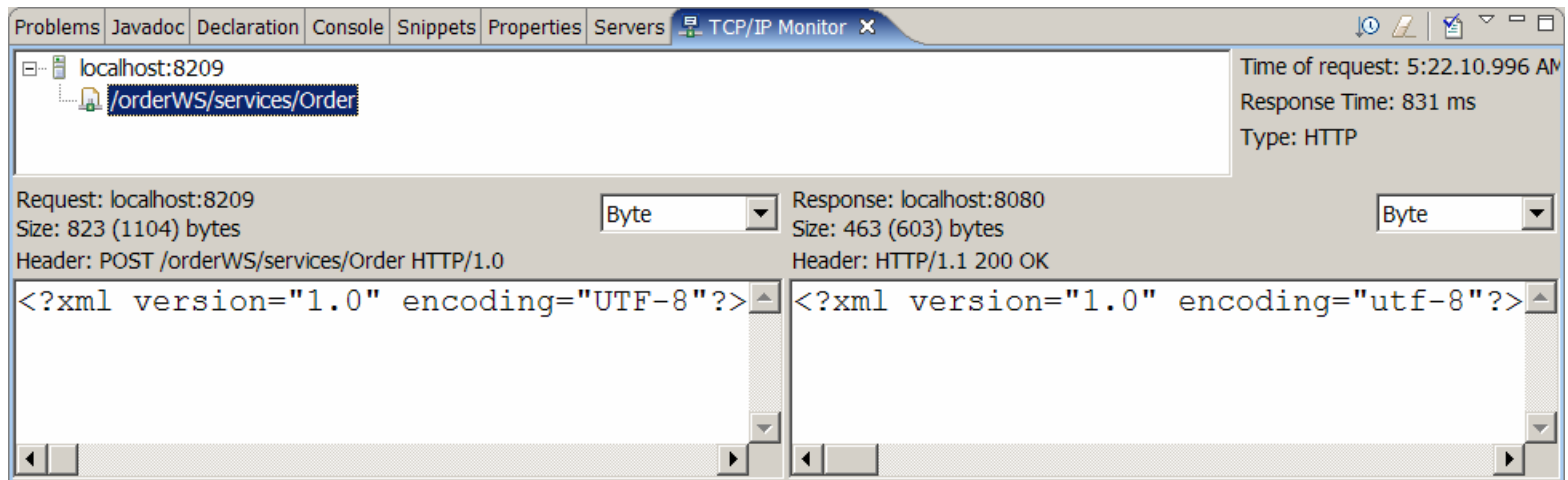
- WTP Overview
- Web Services Overview
- Consuming
- Producing Web Services (bottom up)
- Testing
- Deployment
- Conclusion



# TCP/IP Monitoring



- View SOAP request and response

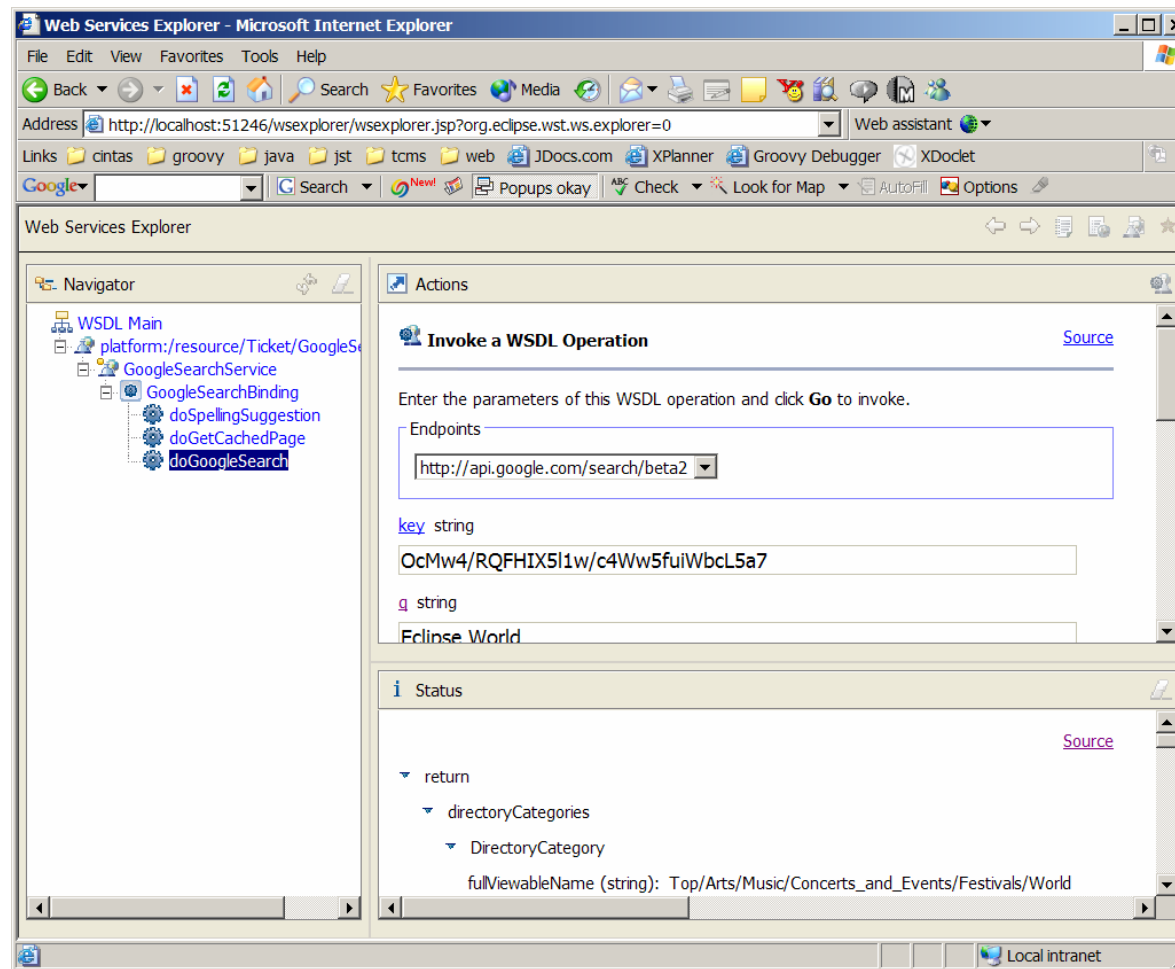




# Web Service Explorer Testing



- Test any Web Services

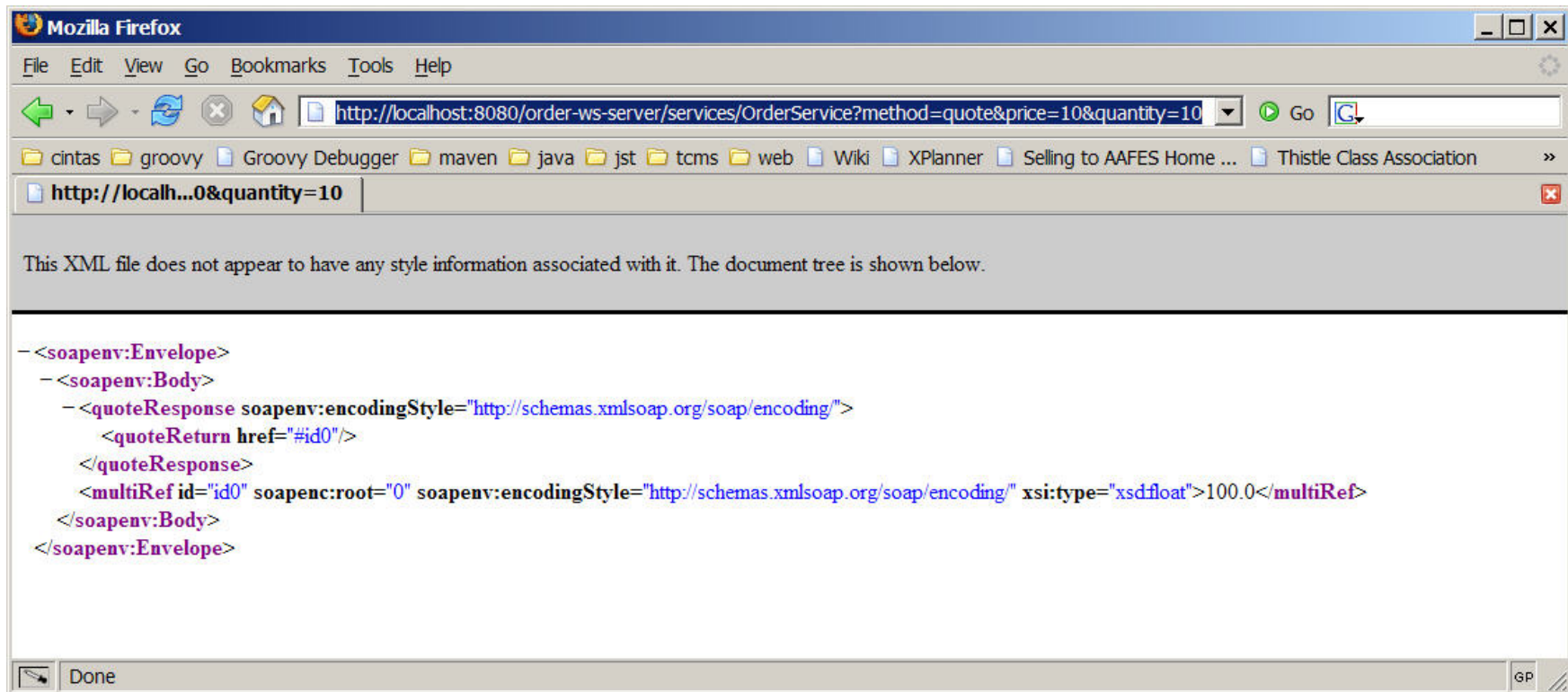




# Browser Testing



- Invoke Axis service via URL





# WSDL Editor



- Graphically edit WSDL

OrderService		
quote		
➔ Input	price	float
	quantity	int
⬅ Output	quoteReturn	float



# Agenda



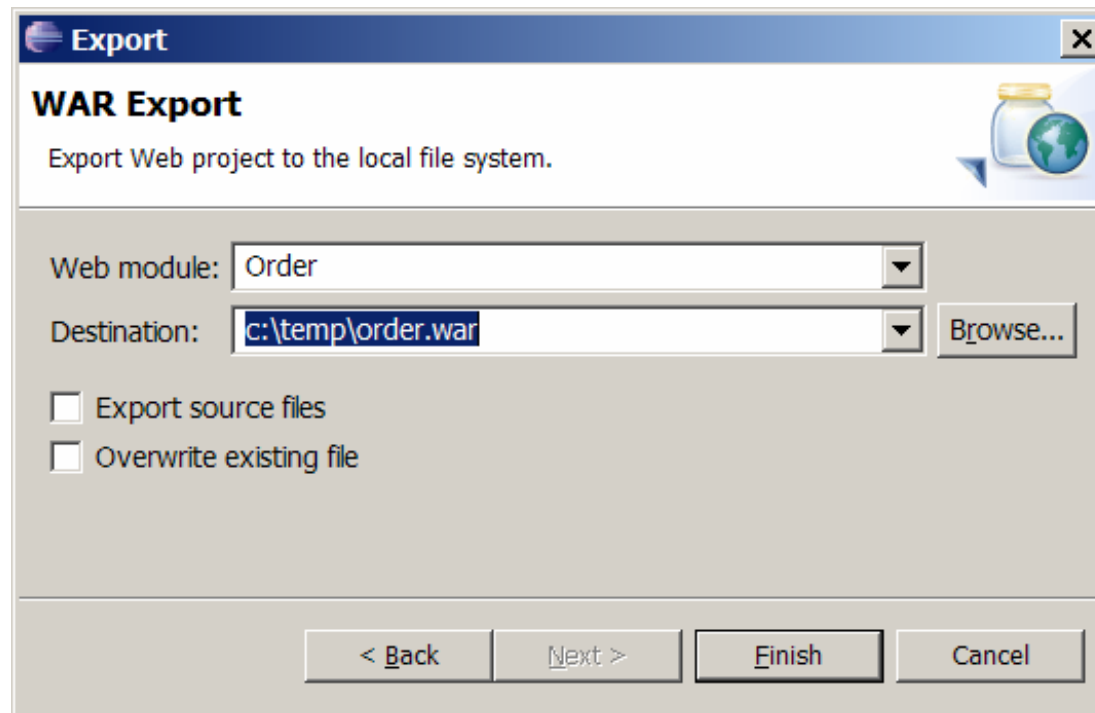
- WTP Overview
- Web Services Overview
- Consuming
- Producing Web Services (bottom up)
- Testing
- Deployment
- Conclusion



# Package



- File > Export > WAR file



The image shows a screenshot of an 'Export' dialog box titled 'WAR Export'. The dialog box has a blue header bar with the title 'Export' and a close button. Below the header, the title 'WAR Export' is displayed in bold, followed by the instruction 'Export Web project to the local file system.' and a small icon of a jar and a globe. The main area contains two dropdown menus: 'Web module:' with 'Order' selected, and 'Destination:' with 'c:\temp\order.war' selected. To the right of the 'Destination:' dropdown is a 'Browse...' button. Below these are two checkboxes: 'Export source files' and 'Overwrite existing file', both of which are unchecked. At the bottom of the dialog box are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.



# Deployment



- Container specific
- Admin console
- Deployment directory
- Ant task
- Server View





# Agenda



- WTP Overview
- Web Services Overview
- Consuming
- Producing Web Services (bottom up)
- Testing
- Deployment
- Conclusion



# WTP Resources



- [www.eclipse.org/webtools/](http://www.eclipse.org/webtools/)
- [www.projst.com](http://www.projst.com)
- Tutorials
  - <http://www.eclipse.org/webtools/community/communityresources.html#tutorials>
- Articles
  - <http://www.eclipse.org/webtools/community/communityresources.html#articles>
  - Build rich Internet applications - <http://www-128.ibm.com/developerworks/edu/os-dw-os-laszlo-i.html>
- New Group
  - <news://news.eclipse.org/eclipse.webtools>



# Contact Information



- <http://www.juddsolutions.com>
- [cjudd@juddsolutions.com](mailto:cjudd@juddsolutions.com)
- Blog
  - <http://blogs.apress.com/authors.php?author=Christopher%20Judd>
- Pro Eclipse JST
  - <http://www.projst.com>
  - <http://www.apress.com/book/bookDisplay.html?bID=447>
- Enterprise Java Development on a Budget
  - <http://www.apress.com/book/bookDisplay.html?bID=197>





# Questions ?

Please fill out your evaluations.

Slides can be found at

<http://www.juddsolutions.com/ew2006/>