Developing Web Services with Open Source

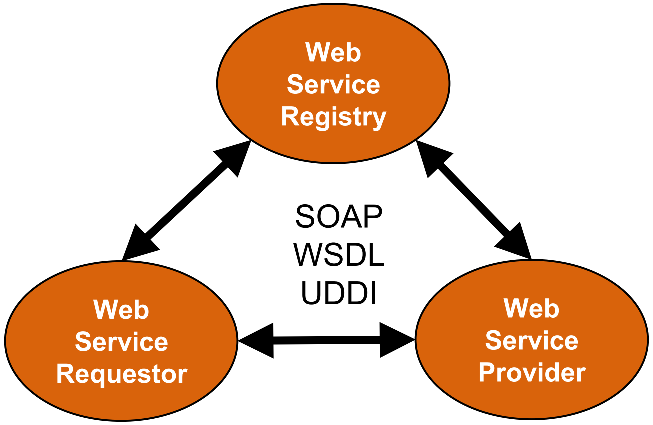
Chris Peltz

Senior Software Consultant
Hewlett Packard
Developer Resources Organization

HP WORLD 2003
Solutions and Technology Conference & Expo

Background

- Web Services have been positioned as a key enabler to EAI and B2B integration
- Web Services are:
  - self-contained, modular software components
  - compliant with open, industry standards (XML, HTTP)
  - can be located and invoked across the Internet
- Key benefits include:
  - lower overall integration costs
  - a higher degree of reusability
  - potential for new revenue streams



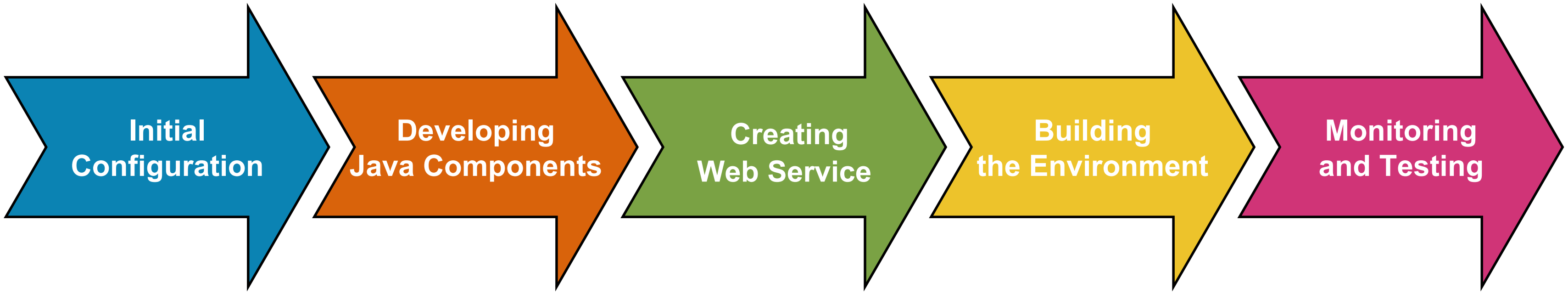
Introduction

- Many companies investigating the use of web services
- Cost required to get started might pose a huge barrier
  - high-priced platforms may not be an option
  - teams may look to open source to get started
- Presentation objectives:
  - present a lifecycle for web services development
  - introduce open source tools for development
  - share key learnings in using some of these tools

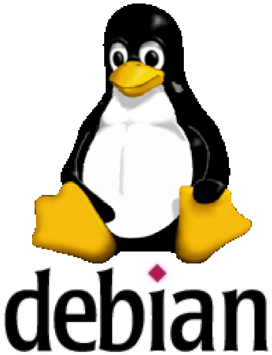
The hope is that you will gain valuable knowledge to

Approach

- Scenario
  - **HotSpell**, a new Internet-based startup company
  - creating a weather forecast service for consumers
  - has selected Java as underlying development platform
  - wishes to expose application as a web service
- You will be led through the entire software lifecycle to develop, deploy, and manage the service using open source tools



Initial Configuration: Selecting a Linux Distribution



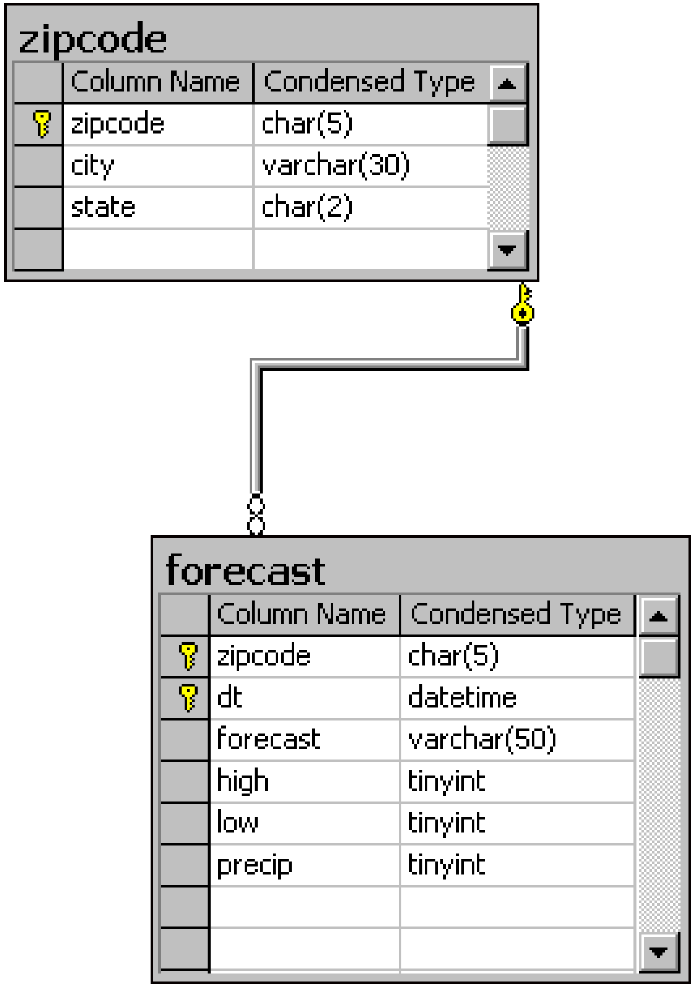
- Why Linux?
  - lower development and deployment costs
  - flexibility to distribute and modify source code
  - strong Internet-based capabilities
- Why Debian Linux?
  - most vendor-neutral (“open source only” policy)
  - provides an easy-to-use interface for installing packages
- Key learnings:
  - provides both Unix and Windows look-and-feel
  - Linux is not a **single** development platform
  - installation made easier with **aptitude** and HP’s PTK

Initial Configuration: Selecting a Database

- Evaluated two open source database offerings
  - PostgreSQL considered more robust in SQL support
  - MySQL designed for speed at the cost of features
  - we selected MySQL for ease-of-use and simplicity
- Installation experience:
  - installed MySQL 3.23.49 from www.mysql.com
  - Debian determined required dependencies for install
  - only change was resolving a TCP/IP connection problem
- Key learnings:
  - MySQL is a stable RDBMS for web-based applications
  - there are many different application packaging formats

Initial Configuration: Designing the Database

- Two simple tables were created to model the data
  - zipcode for city/state information
  - forecast for forecast information
- Data was loaded using the load data SQL command
- Permissions were added to allow user to query database tables



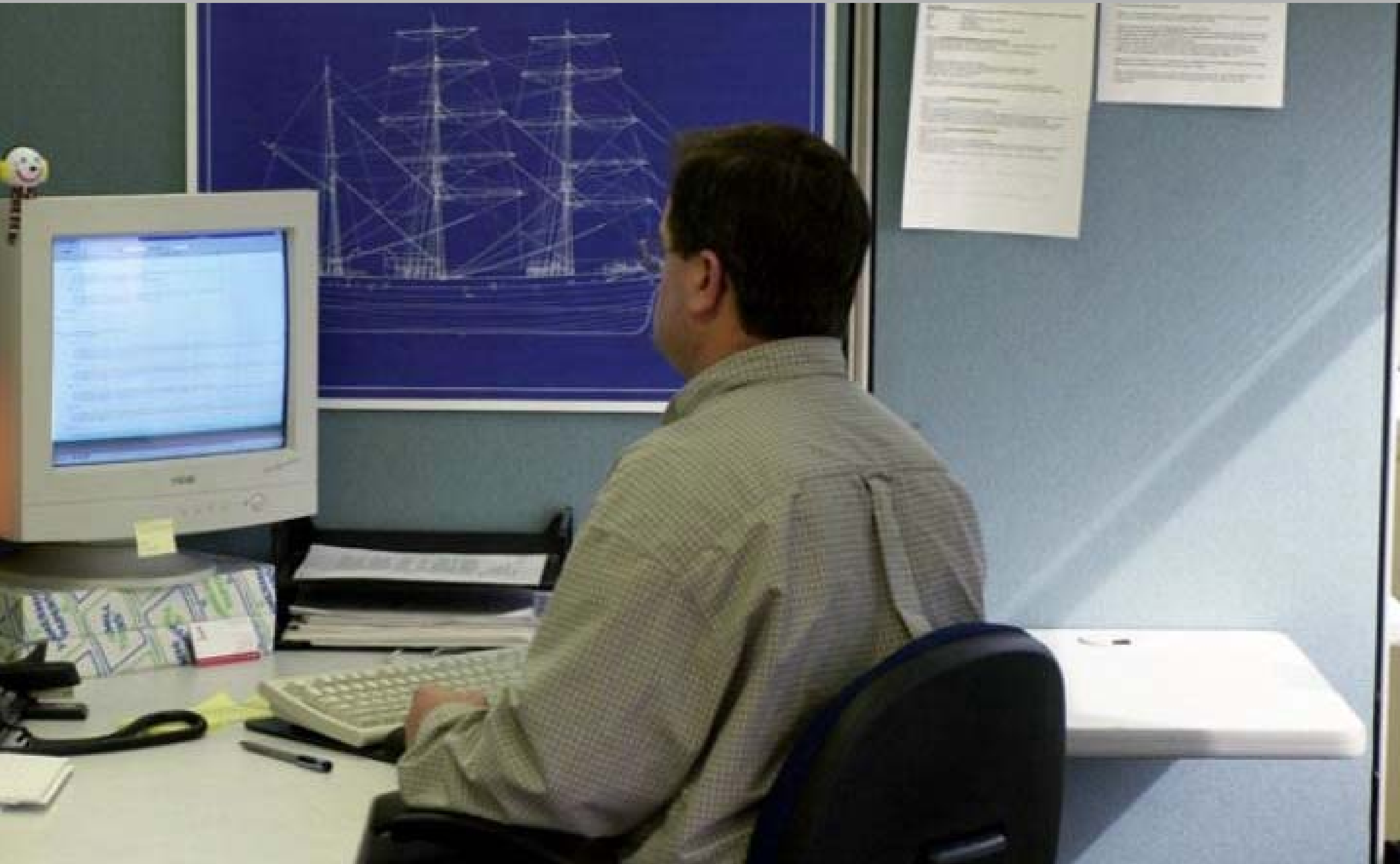
zipcode

	Column Name	Condensed Type
🔑	zipcode	char(5)
	city	varchar(30)
	state	char(2)

forecast

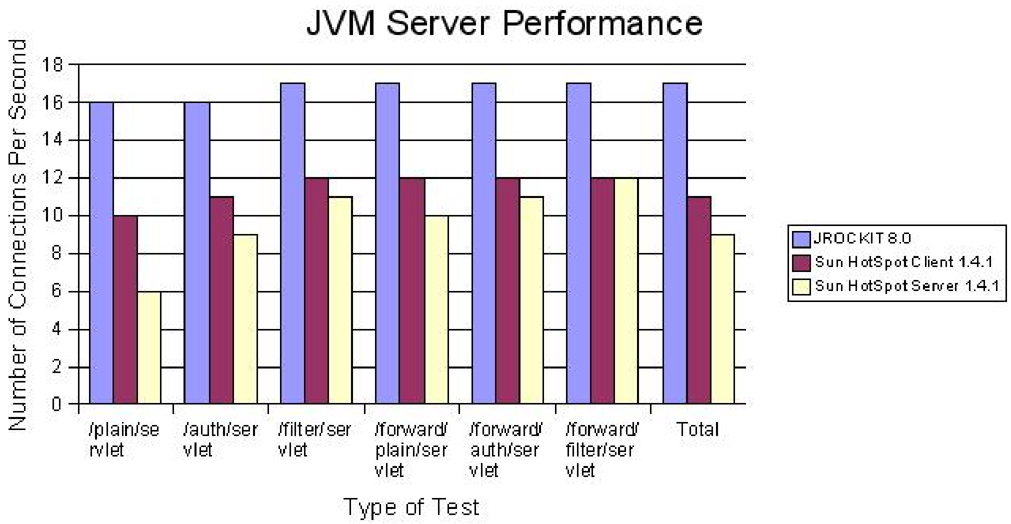
	Column Name	Condensed Type
🔑	zipcode	char(5)
🔑	dt	datetime
	forecast	varchar(50)
	high	tinyint
	low	tinyint
	precip	tinyint

Developing the Java Components



Selecting a Java Environment

- Linux JDK was required to run the application and tools
  - J2SE JDK from Sun (java.sun.com)
  - Blackdown JDK (www.blackdown.org)
  - BEA WebLogic JRockit (www.bea.com)
- JRockit selected for performance reasons
  - MxN threading model for Java threads
  - less memory and context switching
  - higher scalability for thread-intensive applications



Selecting a Java IDE

- IDE
  - Integrated Development Environment
  - provides tools to edit, compile, and debug applications
- Several open source IDEs for Java available
  - NetBeans (www.netbeans.org)
  - Eclipse (www.eclipse.org)
- Eclipse selected because of prior experience with tool
  - HP is Eclipse board member
  - Eclipse plug-ins available for OpenCall and IUM

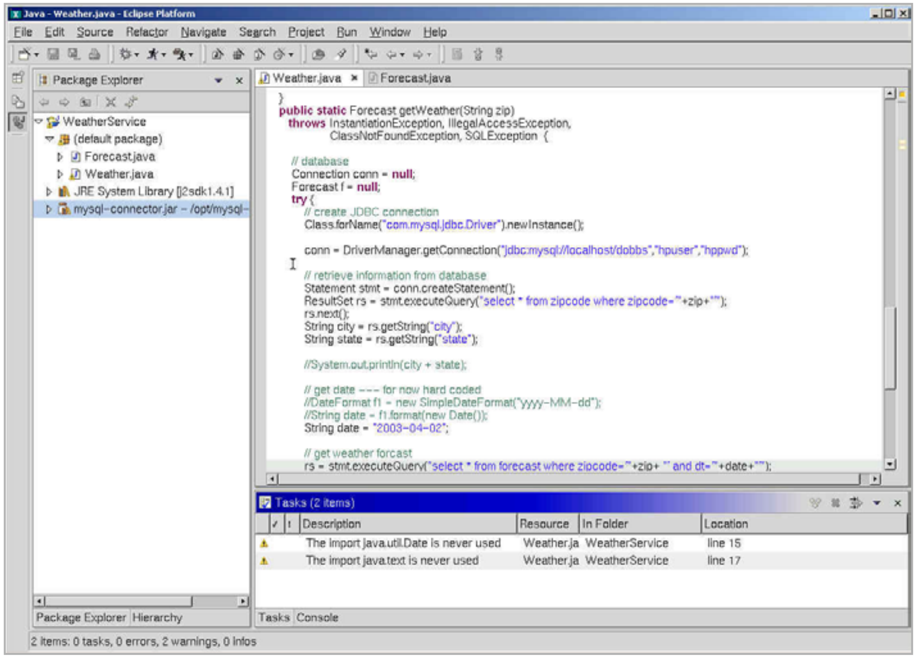
v.s.

- *both open source*
- *both provide Java-based IDEs*
- *Eclipse is Swing-based*
- *NetBeans is SWT-based*

An Overview of Eclipse

Eclipse's strength lies in its ability to easily integrate third-party tools into the development environment

- Key features
  - syntax highlighting editor
  - incremental code completion
  - source-level debugger
  - class navigator
  - file/project manager
  - integration with source control systems
  - task-oriented development through **perspectives**



```
}
public static Forecast getWeather(String zip)
  throws InstantiationException, IllegalAccessException,
           ClassNotFoundException, SQLException {

   // database
   Connection conn = null;
   Forecast f = null;
   try {
     // create JDBC connection
     Class.forName("com.mysql.jdbc.Driver").newInstance();

     conn = DriverManager.getConnection("jdbc:mysql://localhost/dobbs","hpuser","hppwd");

     // retrieve information from database
     Statement stmt = conn.createStatement();
     ResultSet rs = stmt.executeQuery("select * from zipcode where zipcode='"+zip+"'");
     rs.next();
     String city = rs.getString("city");
     String state = rs.getString("state");

     //System.out.println(city + state);

     // get date --- for now hard coded
     //DateFormat f1 = new SimpleDateFormat("yyyy-MM-dd");
     //String date = f1.format(new Date());
     String date = "2003-04-02";

     // get weather forcast
     rs = stmt.executeQuery("select * from forecast where zipcode='"+zip+ "' and dt='"+date+"'");
```

Tasks (2 items)

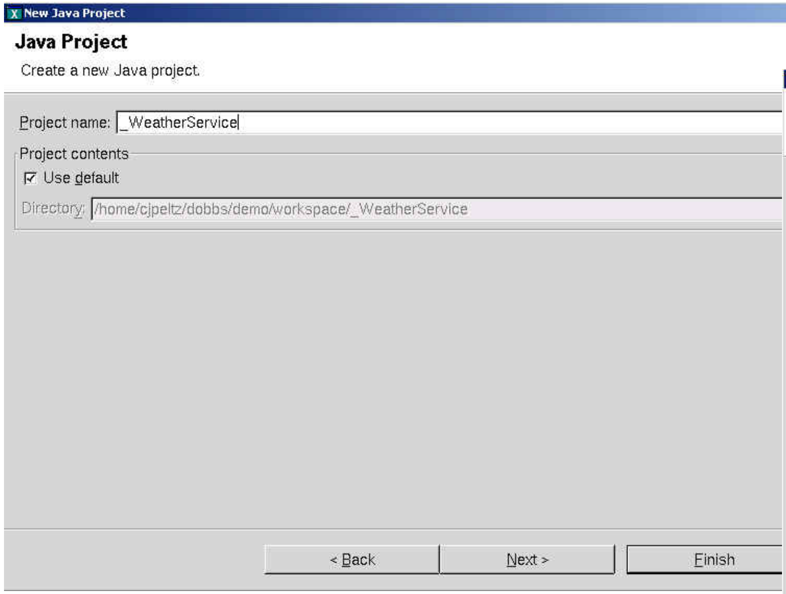
	✓	!	Description	Resource	In Folder	Location
⚠			The import java.util.Date is never used	Weather.ja	WeatherService	line 15
⚠			The import java.text is never used	Weather.ja	WeatherService	line 17

2 items: 0 tasks, 0 errors, 2 warnings, 0 infos

The Eclipse Platform

Developing with Eclipse

- Installation and configuration was straightforward
  - downloaded the Linux version from www.eclipse.org
  - installed in /opt and updated PATH to include binary
- Key development steps:



New Java Project

Java Project

Create a new Java project.

Project name: _WeatherService

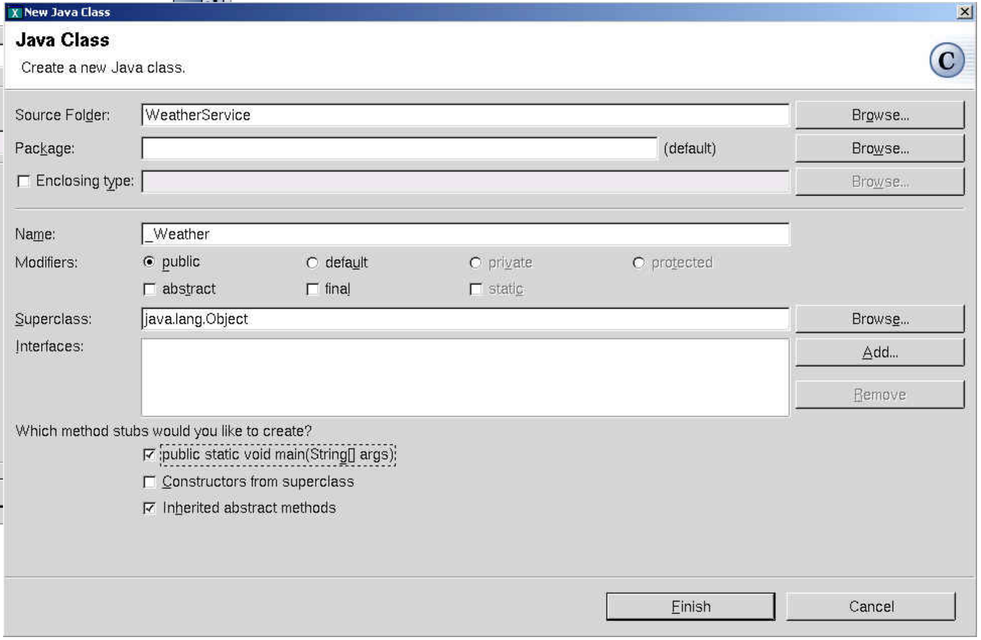
Project contents

☑ Use default

Directory: /home/cjpeltz/dobbs/demo/workspace/_WeatherService

(1) Create Project

(2) Create Java Classes



New Java Class

Java Class

Create a new Java class.

Source Folder: WeatherService

Package: (default)

□ Enclosing type:

Name: _Weather

Modifiers: ● public ○ default ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

Interfaces:

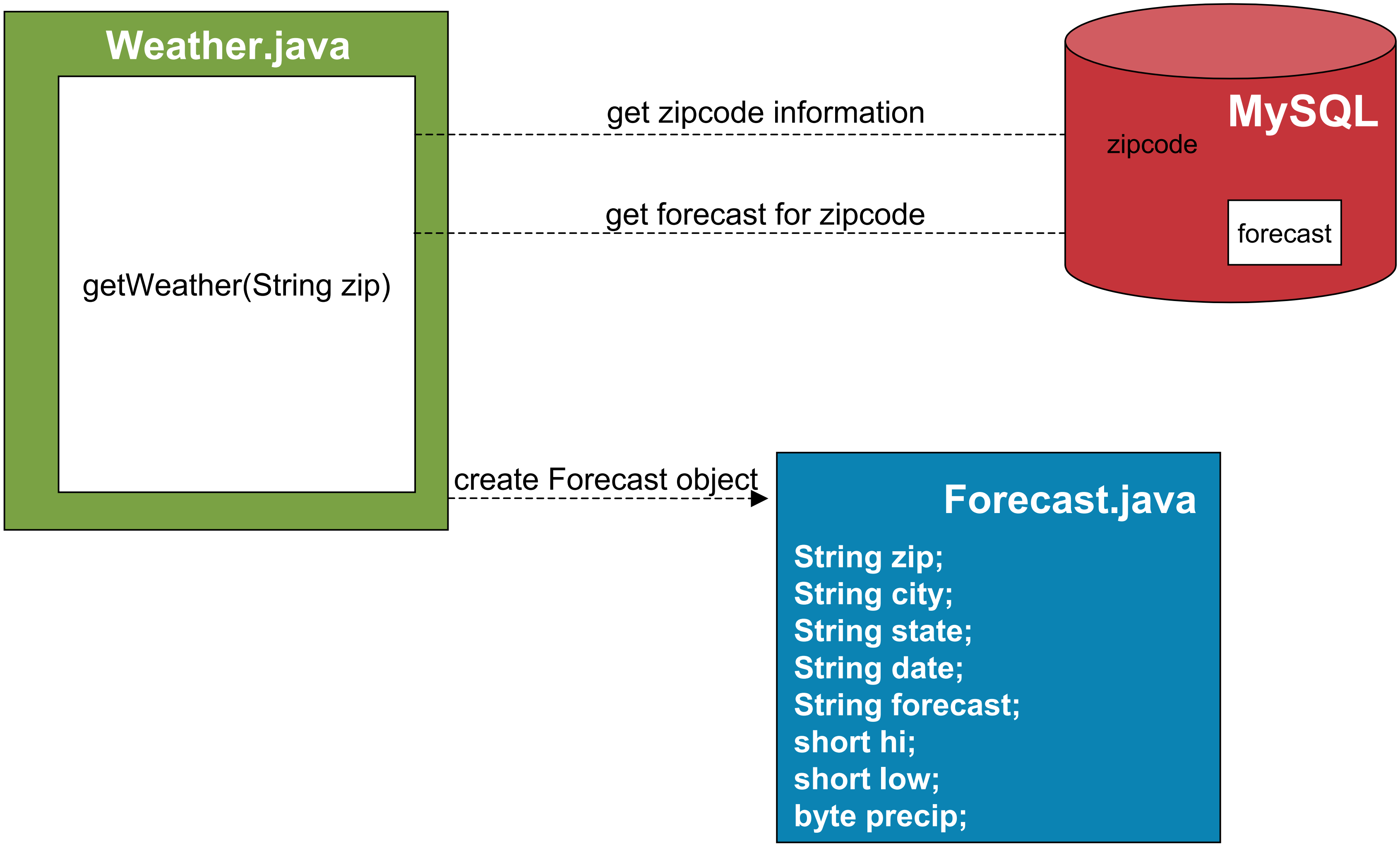
Which method stubs would you like to create?

☑ public static void main(String[] args)

□ Constructors from superclass

☑ Inherited abstract methods

Creating the Java Classes

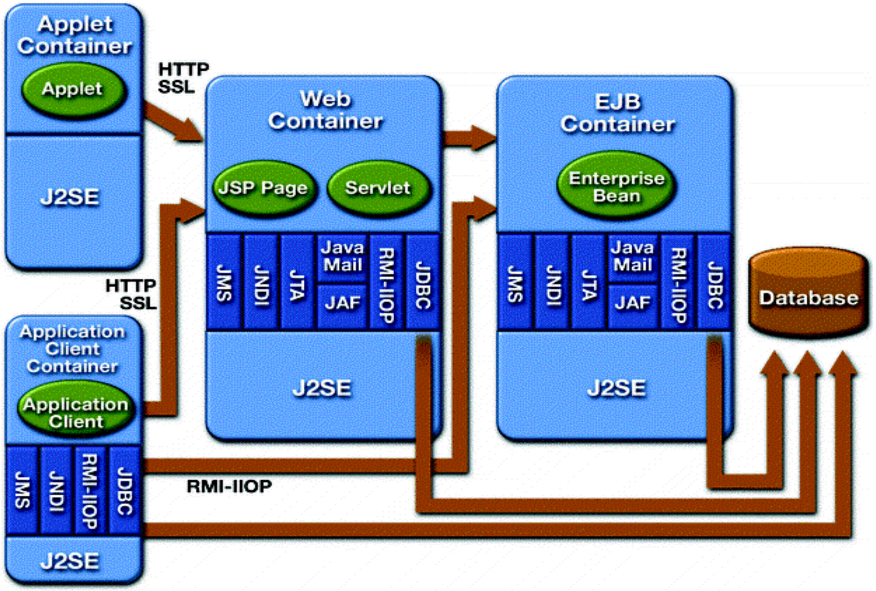


String zip;
String city;
String state;
String date;
String forecast;
short hi;
short low;
byte precip;

Developing the Web Service

The J2EE Web Container

- The web services runtime requires a J2EE web container
- We selected Tomcat
  - widely used open source servlet engine
  - default container for Apache products
- Installing Tomcat:
  - downloaded Tomcat 4.1.24 from jakarta.apache.org
  - configured environment variables
- Starting Tomcat:
  - startup scripts provided
  - Tomcat plug-in for Eclipse

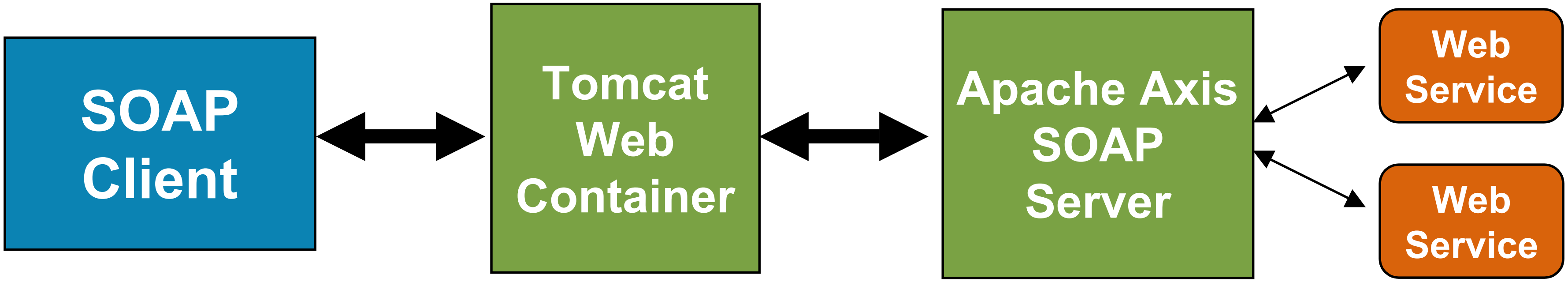


A J2EE Container provides:

- *lifecycle management*
- *security*
- *deployment*
- *runtime service*

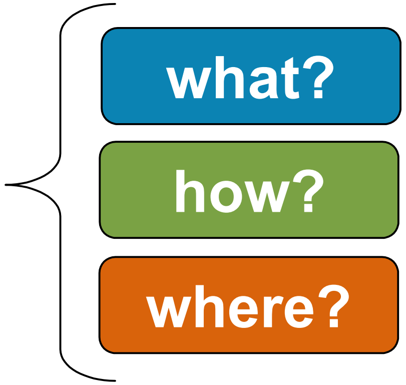
The Web Services Container

- SOAP defines the XML message format for web services
- A Web Services Container:
  - manages the routing and receiving of SOAP messages
  - maps received SOAP messages to back-end components
  - provides tools for creating and deploying web services
- Apache Axis (www.apache.org/axis) was the open source platform chosen for this application



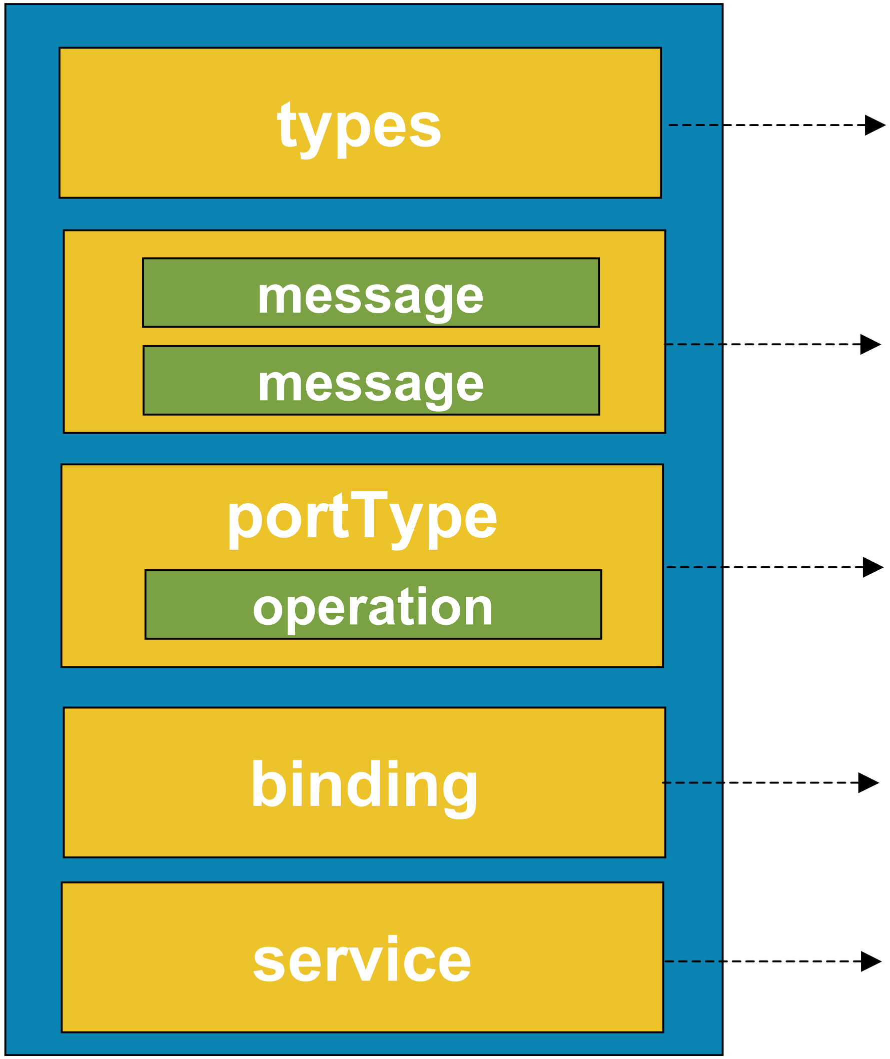
Designing the Web Service Interface

- WSDL
  - Web Services Description Language
  - defines the “signature” of the web service
  - XML-based, independent of platform



- Two approaches for designing a WSDL
  - design WSDL first, then map to business objects
  - have the WSDL be automatically generated from code
- The “WSDL First” approach is usually recommended for complex document exchanges
  - for our simple demo, only one method was exposed
  - we relied on Apache Axis tools to generate WSDL

Our Weather.wsdl



XML schema, mapping to the Forecast object

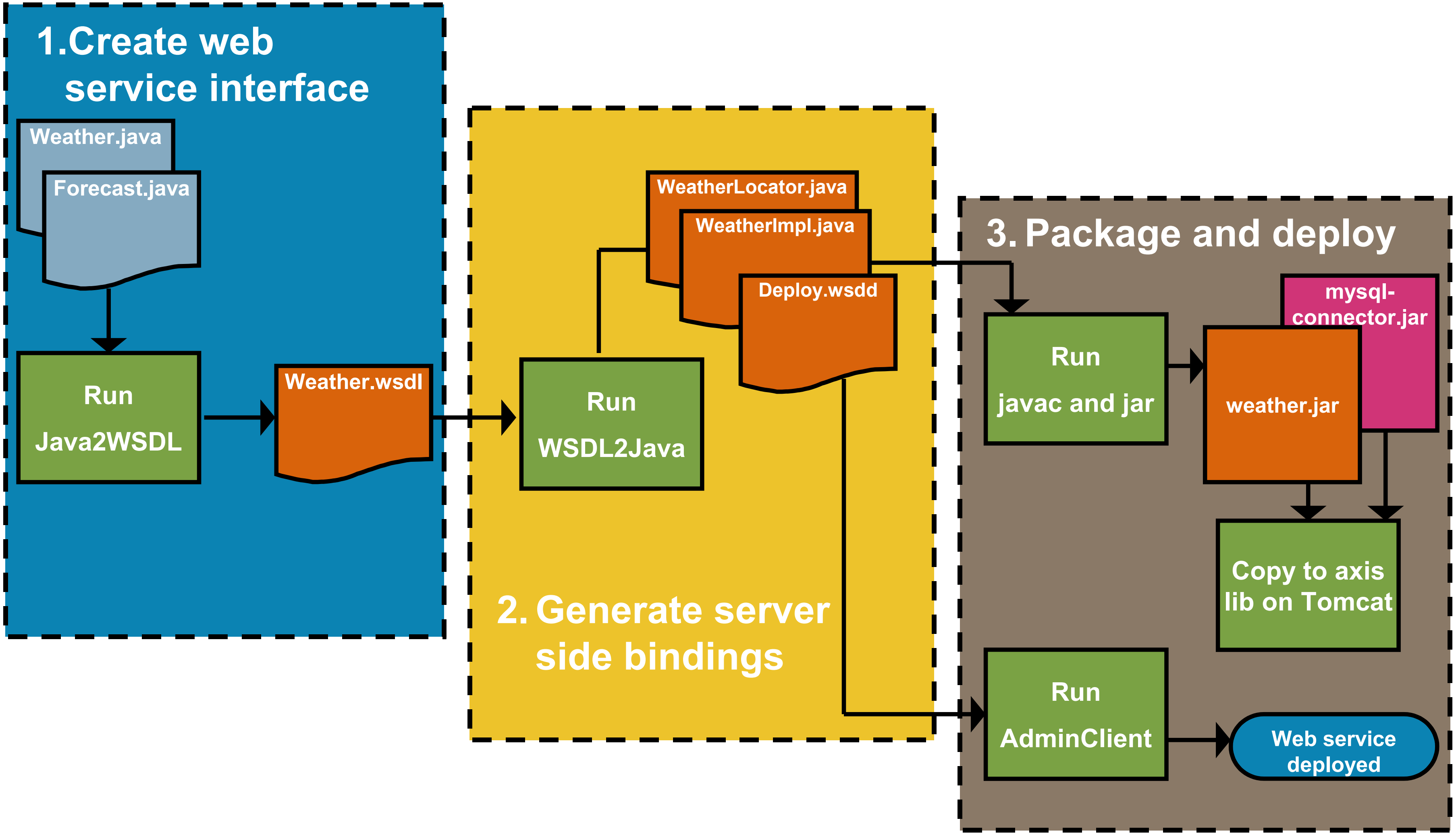
contains a request message and a response message for the service

defines one operation, getWeather, containing the request/response pair

defines specific bindings to SOAP protocols

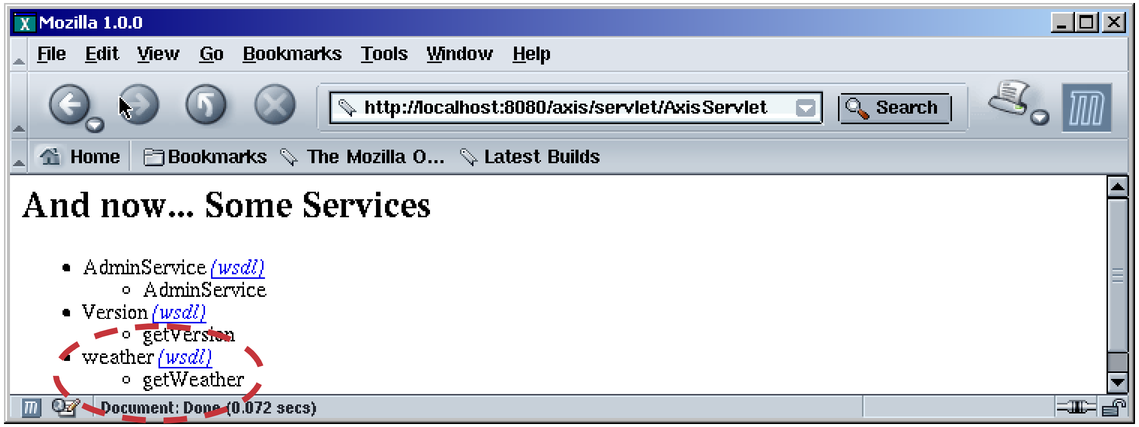
provides the URL reference for invoking the service

Using Apache Axis



Our Experience

- Existing code may not support web services model
- Apache Axis provided a sufficient development platform
  - mostly command-line as compared to other tools
- Generated server-side bindings were not "complete"
  - logic was added to invoke the original Java classes
- Deployment process was very straightforward


Mozilla 1.0.0
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/axis/servlet/AxisServlet
Search
Home Bookmarks The Mozilla O... Latest Builds

And now... Some Services

- AdminService (wsdl)
  - AdminService
- Version (wsdl)
  - getVersion
- weather (wsdl)
  - getWeather

Document: Done (0.072 secs)

Creating an Automated Build Process

Why Do You Need a Build Process?

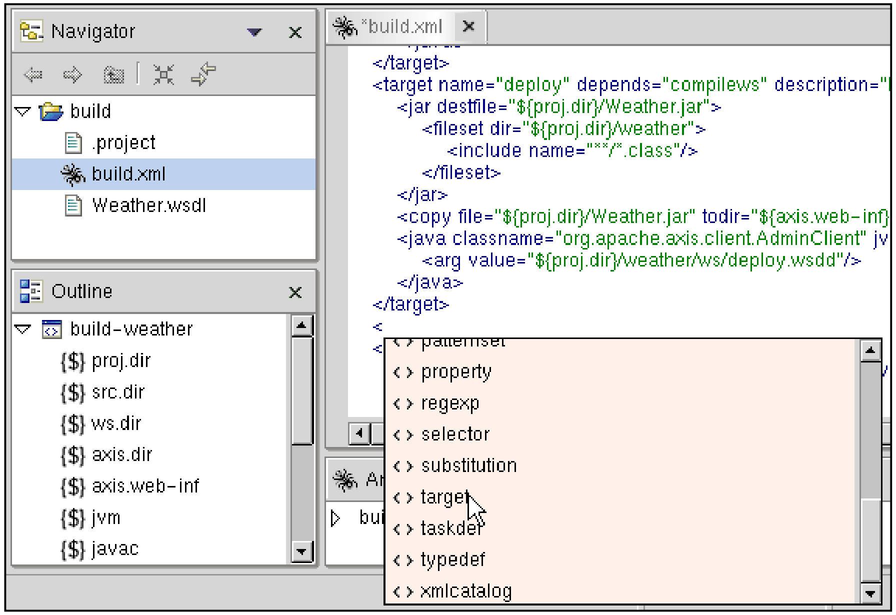
- Tools are available for creating web services
  - for Apache Axis, process is mostly command-line
  - can be time consuming if components are rebuilt
- A build process can automate many of these steps
  - can greatly enhance developer productivity
- Consider an eXtreme Programming (XP) methodology
  - "continuous integration" – deploy early and often
  - single commands to build and test the web services
  - automated builds are conducted a few times a day

Extreme Programming

An Introduction to Apache Ant

“A Java-based build tool designed to be cross-platform, easy-to-use, extensible, and scalable”[1]

- Highly portable across operating systems and platforms
- Component-based model makes it easy to extend
- Can build Java and web services components
- Fully integrated with the Eclipse environment
- See ant.apache.org for more information

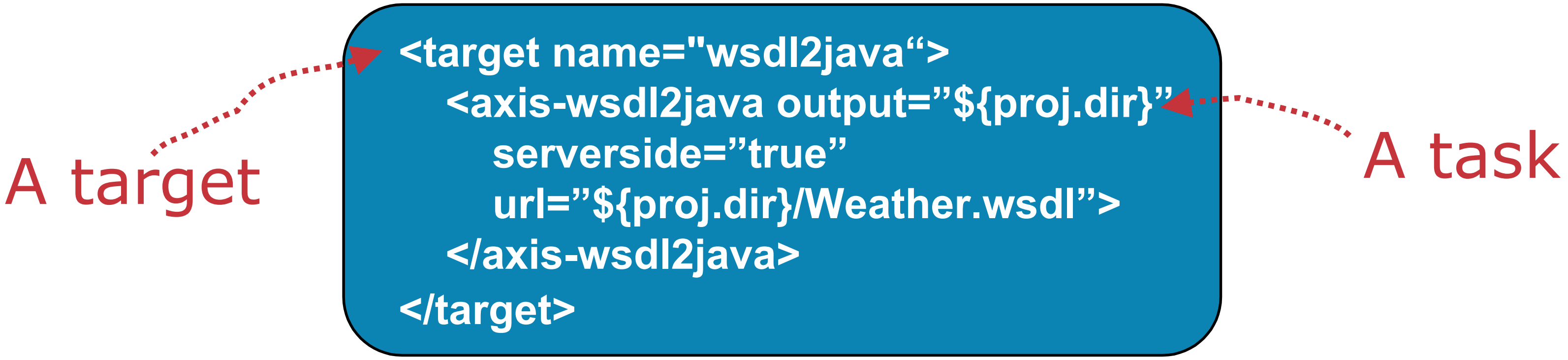


```
</target>
<target name="deploy" depends="compilews" description="
   <jar destfile="${proj.dir}/Weather.jar">
      <fileset dir="${proj.dir}/weather">
         <include name="**/*.class"/>
      </fileset>
   </jar>
   <copy file="${proj.dir}/Weather.jar" todir="${axis.web-inf}
   <java classname="org.apache.axis.client.AdminClient" jv
      <arg value="${proj.dir}/weather/ws/deploy.wsdd"/>
   </java>
</target>
<
```

[1] Source: “Java Development With Ant”

The Build Script

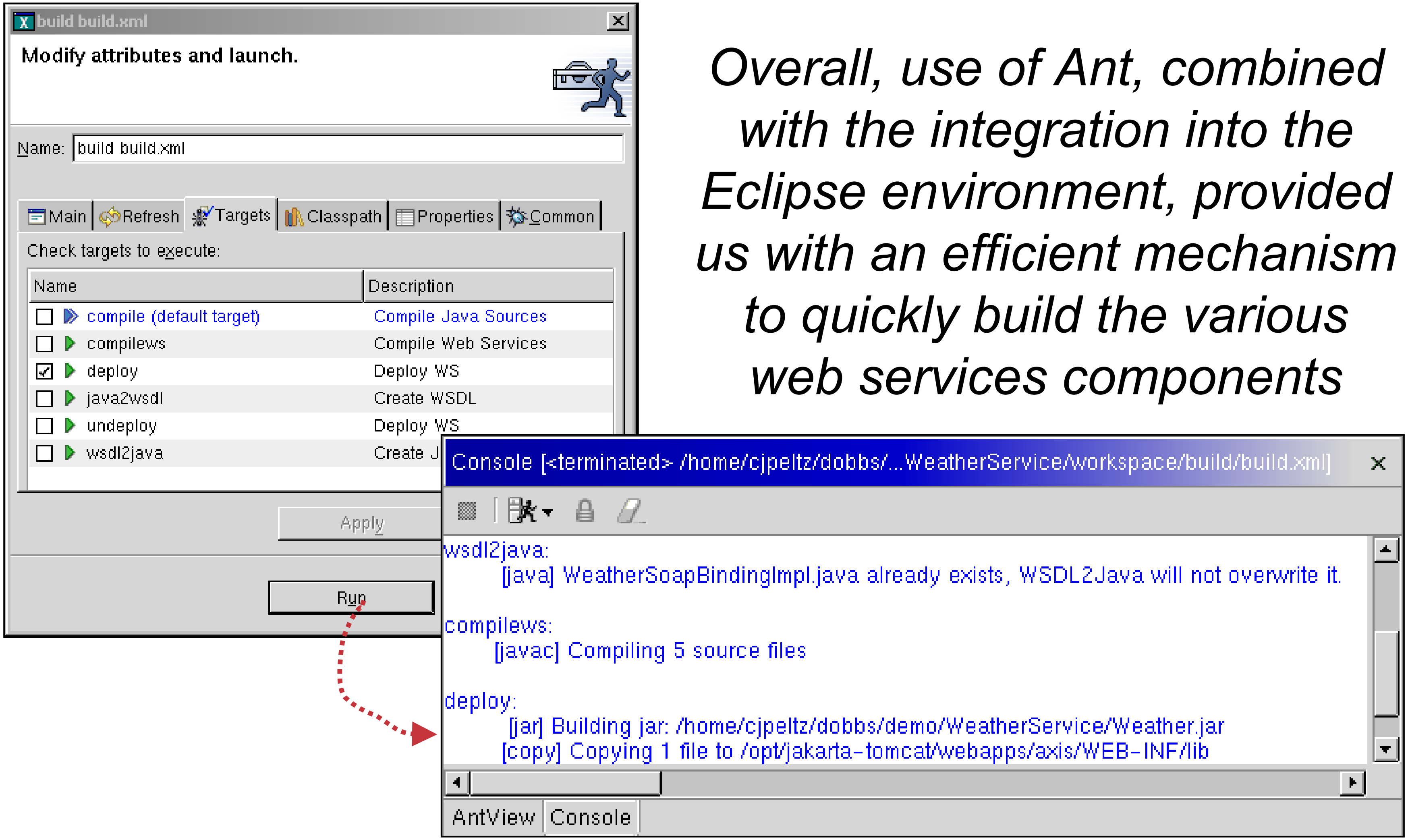
- A build project can contain multiple targets
  - a target represents a specific step in the build process
  - a target can have dependencies on other targets
- Targets contain tasks
  - creating, deleting, and copying files
  - compiling and packaging Java classes
- Apache Axis provides Ant tasks, e.g.:



```
<target name="wsdl2java“>
    <axis-wsdl2java output=”${proj.dir}”
        serverside=”true”
        url=”${proj.dir}/Weather.wsdl”>
    </axis-wsdl2java>
</target>
```

Running the Build Script

Overall, use of Ant, combined with the integration into the Eclipse environment, provided us with an efficient mechanism to quickly build the various web services components



build build.xml

Modify attributes and launch.

Name: build build.xml

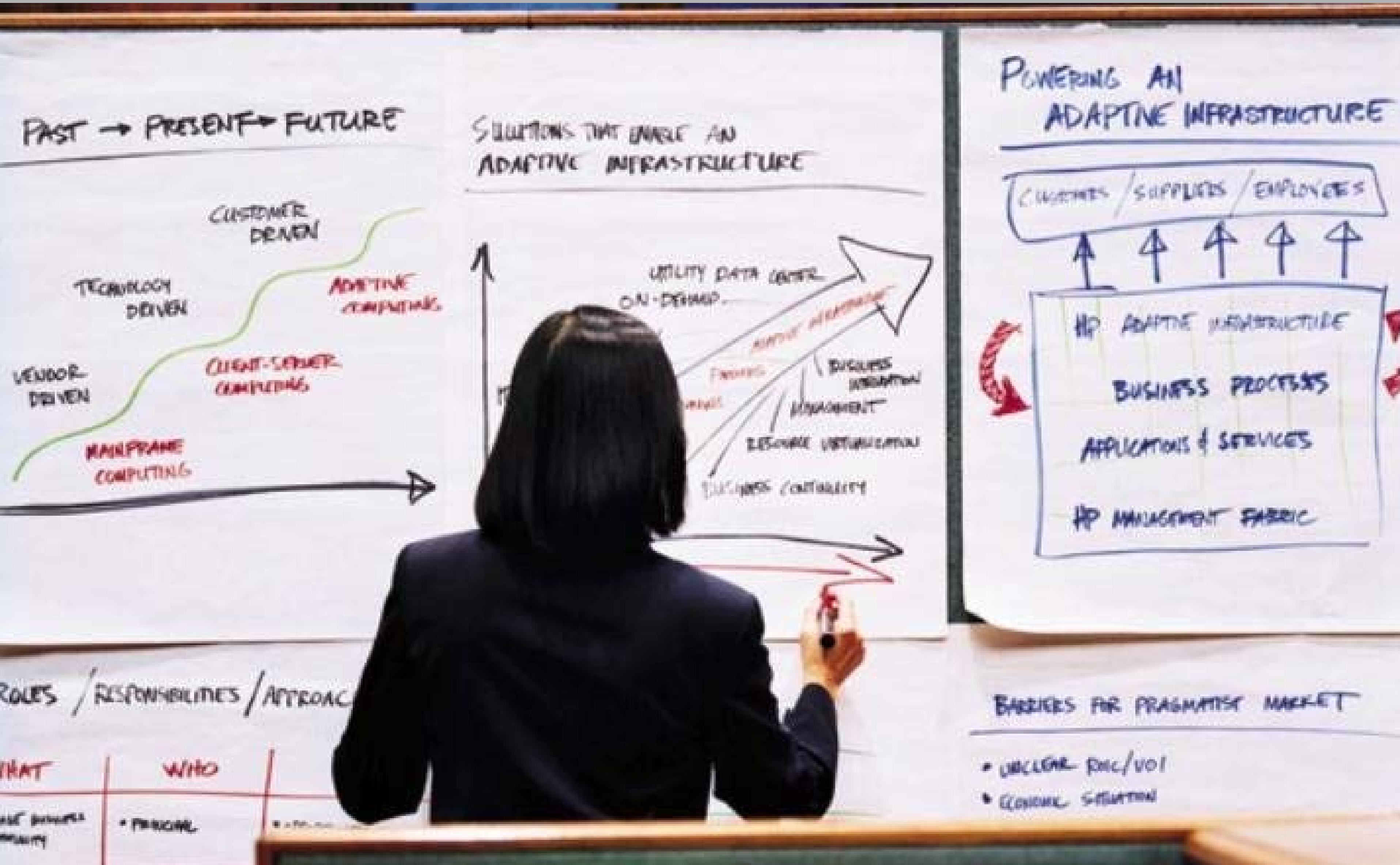
Check targets to execute:

	Name	Description
□	compile (default target)	Compile Java Sources
□	compilews	Compile Web Services
☑	deploy	Deploy WS
□	java2wsdl	Create WSDL
□	undeploy	Deploy WS
□	wsdl2java	Create J

Console [<terminated> /home/cjpeltz/dobbs/...WeatherService/workspace/build/build.xml]

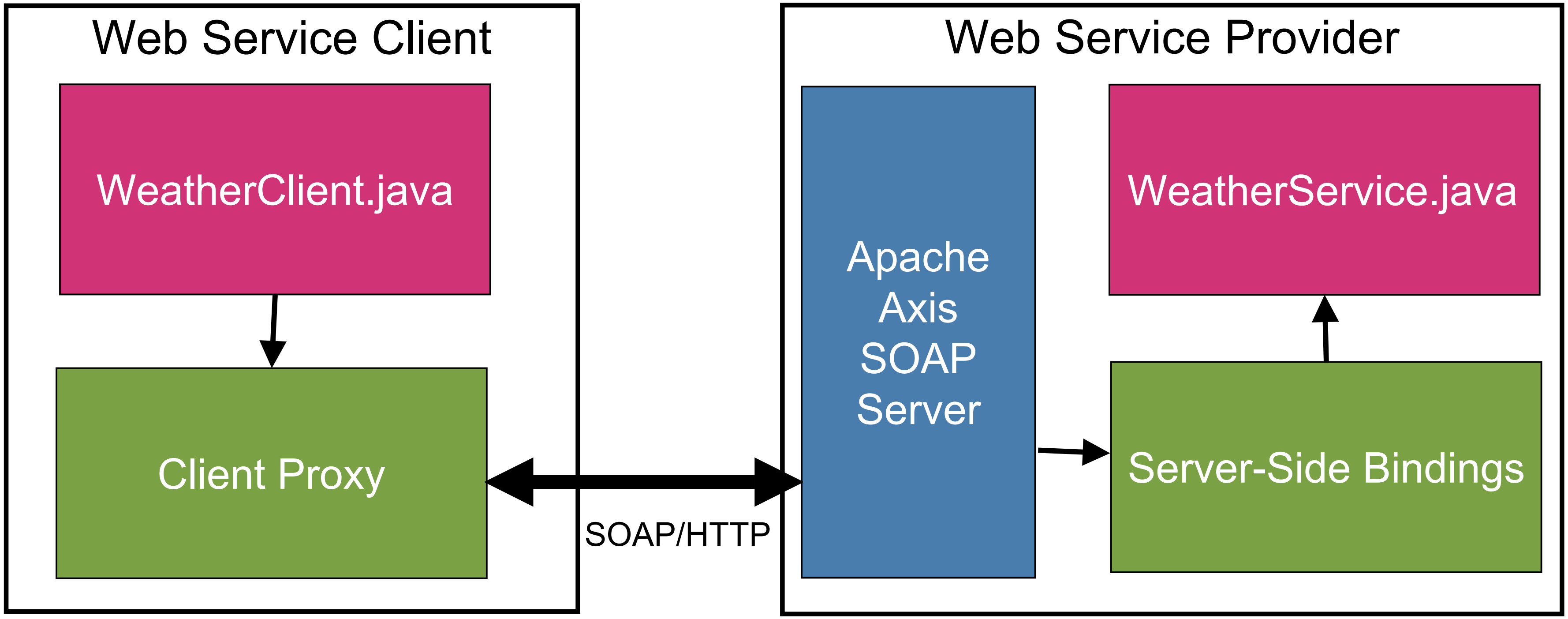
```
wsdl2java:
      [java] WeatherSoapBindingImpl.java already exists, WSDL2Java will not overwrite it.

compilews:
     [javac] Compiling 5 source files

deploy:
       [jar] Building jar: /home/cjpeltz/dobbs/demo/WeatherService/Weather.jar
      [copy] Copying 1 file to /opt/jakarta-tomcat/webapps/axis/WEB-INF/lib
```

Testing the Web Service



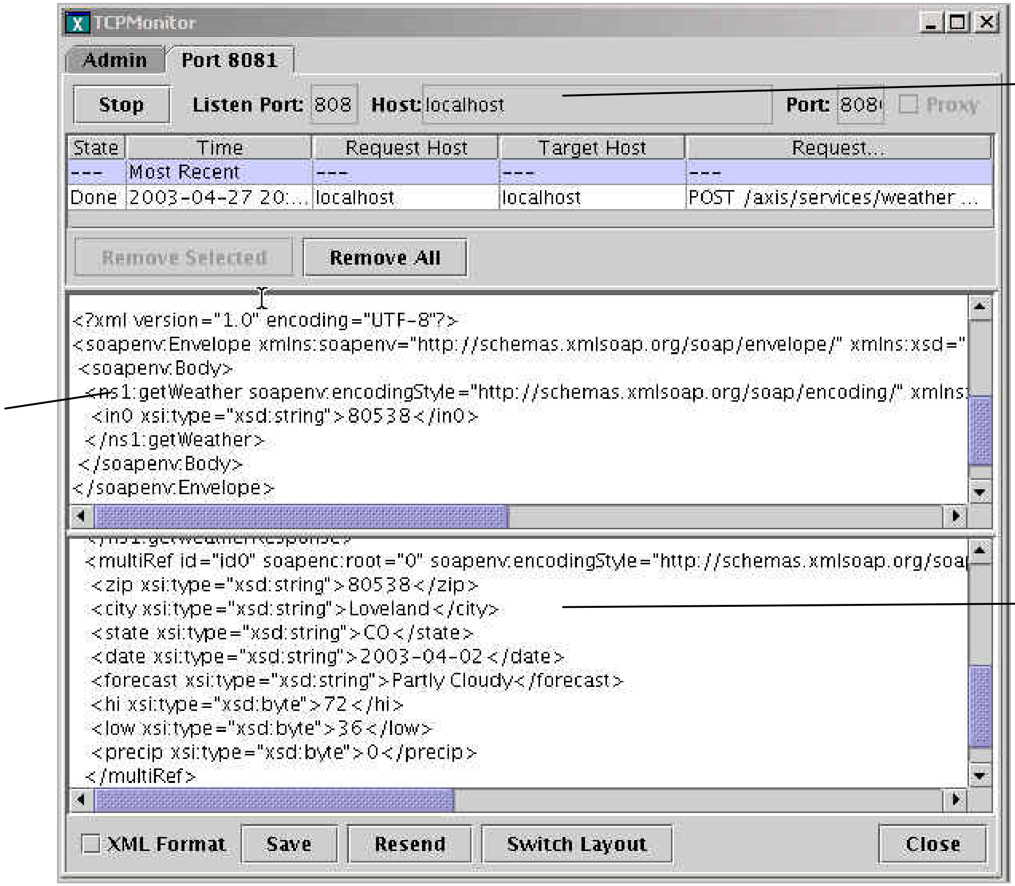
Invoking the Service

- Client proxies isolate SOAP processing code
- Apache Axis automatically creates these components
- We had to write additional logic to use the proxy



Monitoring the Web Service

Apache Axis provides a TCP Monitor tool that monitors SOAP requests and responses



Listen Port: 808 Host: localhost Port: 808 □ Proxy

State	Time	Request Host	Target Host	Request...
---	Most Recent	---	---	---
Done	2003-04-27 20:...	localhost	localhost	POST /axis/services/weather ...

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="
 <soapenv:Body>
  <ns1:getWeather soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:
   <in0 xsi:type="xsd:string">80538</in0>
  </ns1:getWeather>
 </soapenv:Body>
</soapenv:Envelope>
```

```
  </ns1:getWeatherResponse>
  <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soa
   <zip xsi:type="xsd:string">80538</zip>
   <city xsi:type="xsd:string">Loveland</city>
   <state xsi:type="xsd:string">CO</state>
   <date xsi:type="xsd:string">2003-04-02</date>
   <forecast xsi:type="xsd:string">Partly Cloudy</forecast>
   <hi xsi:type="xsd:byte">72</hi>
   <low xsi:type="xsd:byte">36</low>
   <precip xsi:type="xsd:byte">0</precip>
  </multiRef>
```

Proxy Configuration

getWeather SOAP request

Forecast SOAP Response

Testing the Service

- Important considerations:
  - create graphical interfaces to test web services
  - design a test framework usable by non-developers
  - build tests into the development process early on
  - test security, reliability, interoperability, and scalability
  - consider the use of automated testing tools

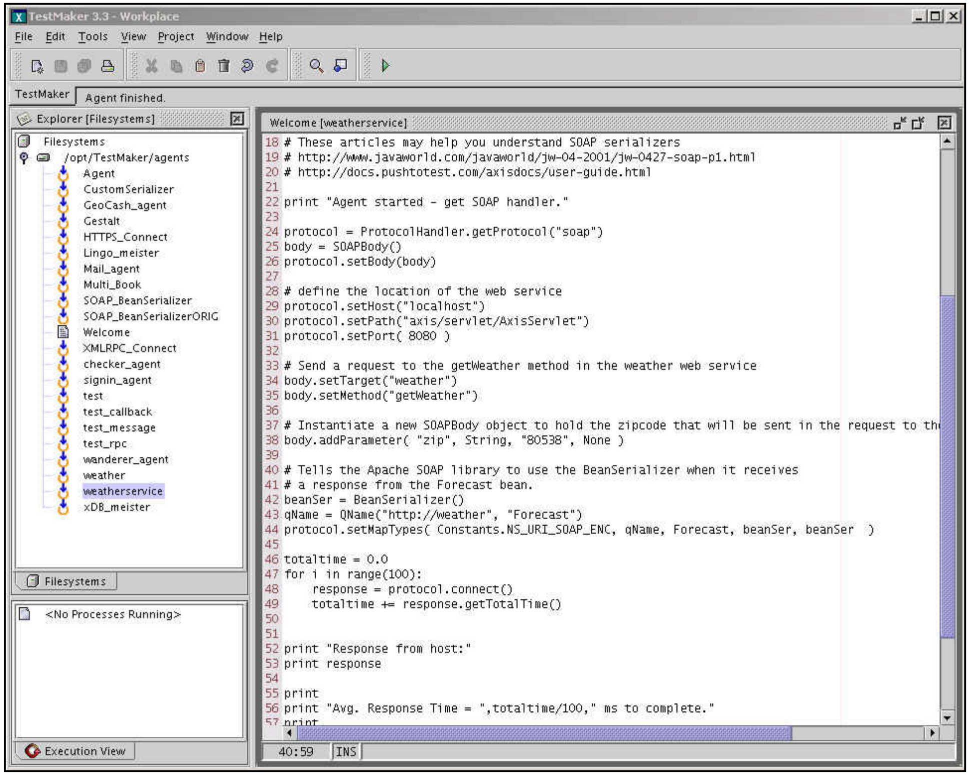
Open Source Testing Tools

- **JUnit**: general framework for testing Java code
- **Grinder**: tool for load-testing web applications
- **Anteater**: Ant-based testing tool with SOAP support
- **PushToTest**: specifically targeted at web services

PushToTest TestMaker

- An open source web services testing tool
- Robust graphical environment and scripting language
  - tool can generate test case from given WSDL
  - scripts written used Jython (Python for Java)
  - comes with library to simplify creation of web services tests
- Allows you to test functionality and scalability of a web service
  - validate SOAP messages received
  - configure stress tests with multiple virtual clients

PushToTest TestMaker



```
# These articles may help you understand SOAP serializers
# http://www.javaworld.com/javaworld/jw-04-2001/jw-0427-soap-p1.html
# http://docs.pushtotest.com/axisdocs/user-guide.html

print "Agent started - get SOAP handler."

protocol = ProtocolHandler.getProtocol("soap")
body = SOAPBody()
protocol.setBody(body)

# define the location of the web service
protocol.setHost("localhost")
protocol.setPath("axis/servlet/AxisServlet")
protocol.setPort( 8080 )

# Send a request to the getWeather method in the weather web service
body.setTarget("weather")
body.setMethod("getWeather")

# Instantiate a new SOAPBody object to hold the zipcode that will be sent in the request to th
body.addParameter( "zip", String, "80538", None )

# Tells the Apache SOAP library to use the BeanSerializer when it receives
# a response from the Forecast bean.
beanSer = BeanSerializer()
qName = QName("http://weather", "Forecast")
protocol.setMapTypes( Constants.NS_URI_SOAP_ENC, qName, Forecast, beanSer, beanSer  )

totaltime = 0.0
for i in range(100):
    response = protocol.connect()
    totaltime += response.getTotalTime()


print "Response from host:"
print response

print
print "Avg. Response Time = ",totaltime/100," ms to complete."
print
```

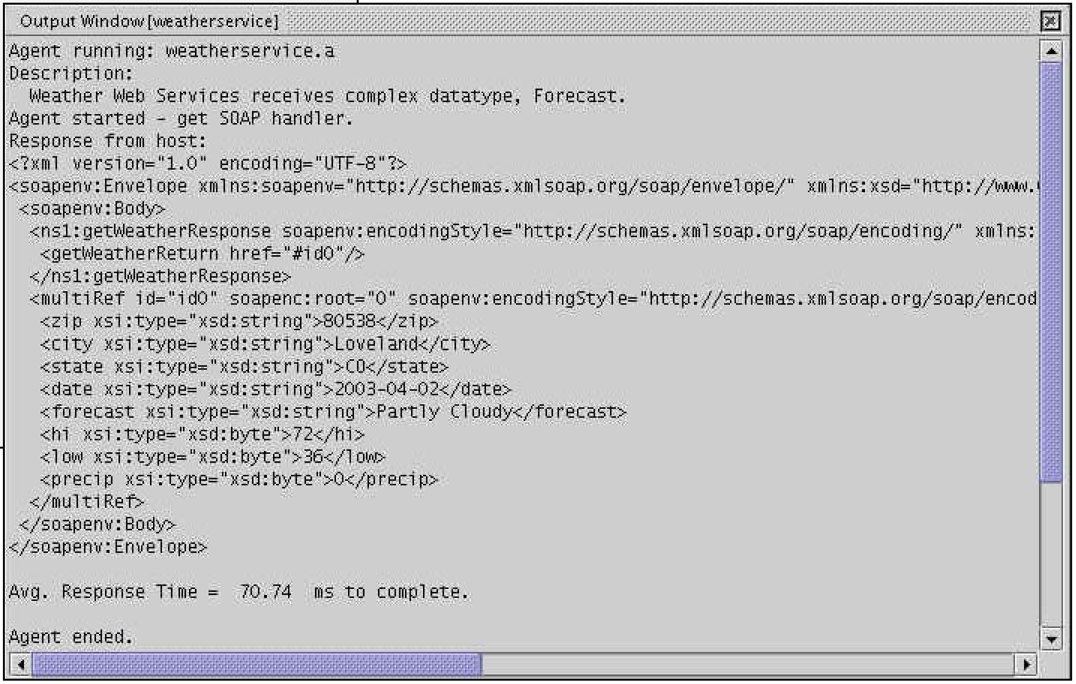
www.pushtotest.com

Running the Test

```
# create protocol for Axis servlet
protocol.setHost("localhost")
protocol.setPath("axis/servlet/AxisServlet")
protocol.setPort( 8081 )

# construct SOAP message with getWeather request
body.setTarget("weather")
body.setMethod("getWeather")
body.addParameter( "zip", String, "80538", None )

# invoke service 100 times
totaltime = 0.0
for I in range(100):
     response = protocol.connect()
     totaltime +=
     response.getTotalTime()

# print response
sprint "Avg. Response Time ="
sprint totaltime/100
sprint "ms to complete."
```

TestMaker Test Script

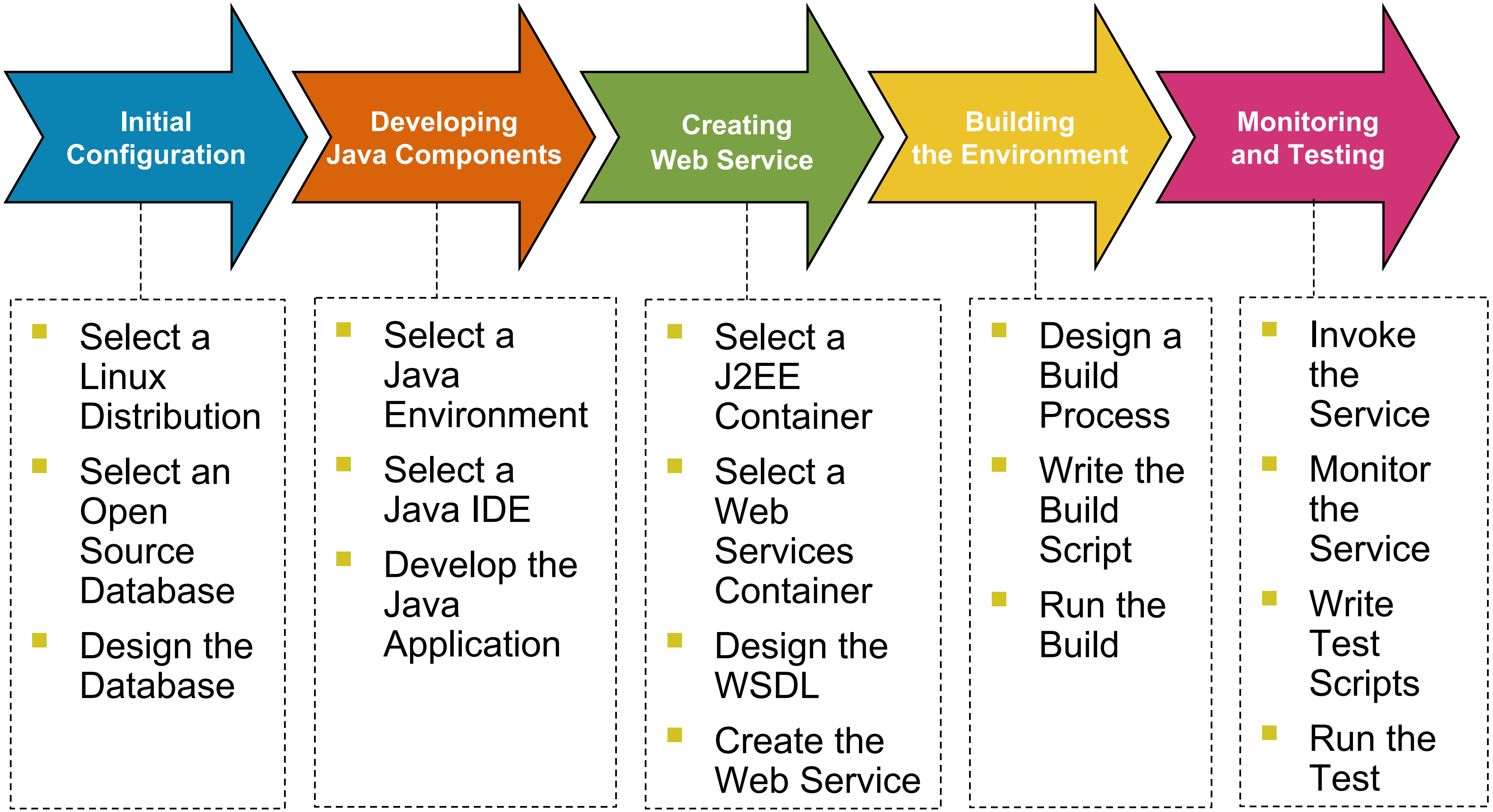
TestMaker Output



```
Agent running: weatherservice.a
Description:
  Weather Web Services receives complex datatype, Forecast.
Agent started - get SOAP handler.
Response from host:
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.
 <soapenv:Body>
  <ns1:getWeatherResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:
   <getWeatherReturn href="#id0"/>
  </ns1:getWeatherResponse>
  <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encod
   <zip xsi:type="xsd:string">80538</zip>
   <city xsi:type="xsd:string">Loveland</city>
   <state xsi:type="xsd:string">CO</state>
   <date xsi:type="xsd:string">2003-04-02</date>
   <forecast xsi:type="xsd:string">Partly Cloudy</forecast>
   <hi xsi:type="xsd:byte">72</hi>
   <low xsi:type="xsd:byte">36</low>
   <precip xsi:type="xsd:byte">0</precip>
  </multiRef>
 </soapenv:Body>
</soapenv:Envelope>

Avg. Response Time =  70.74  ms to complete.

Agent ended.
```

Conclusion

Let’s Review…



Initial Configuration

- Select a Linux Distribution
- Select an Open Source Database
- Design the Database

Developing Java Components

- Select a Java Environment
- Select a Java IDE
- Develop the Java Application

Creating Web Service

- Select a J2EE Container
- Select a Web Services Container
- Design the WSDL
- Create the Web Service

Building the Environment

- Design a Build Process
- Write the Build Script
- Run the Build

Monitoring and Testing

- Invoke the Service
- Monitor the Service
- Write Test Scripts
- Run the Test

Conclusion

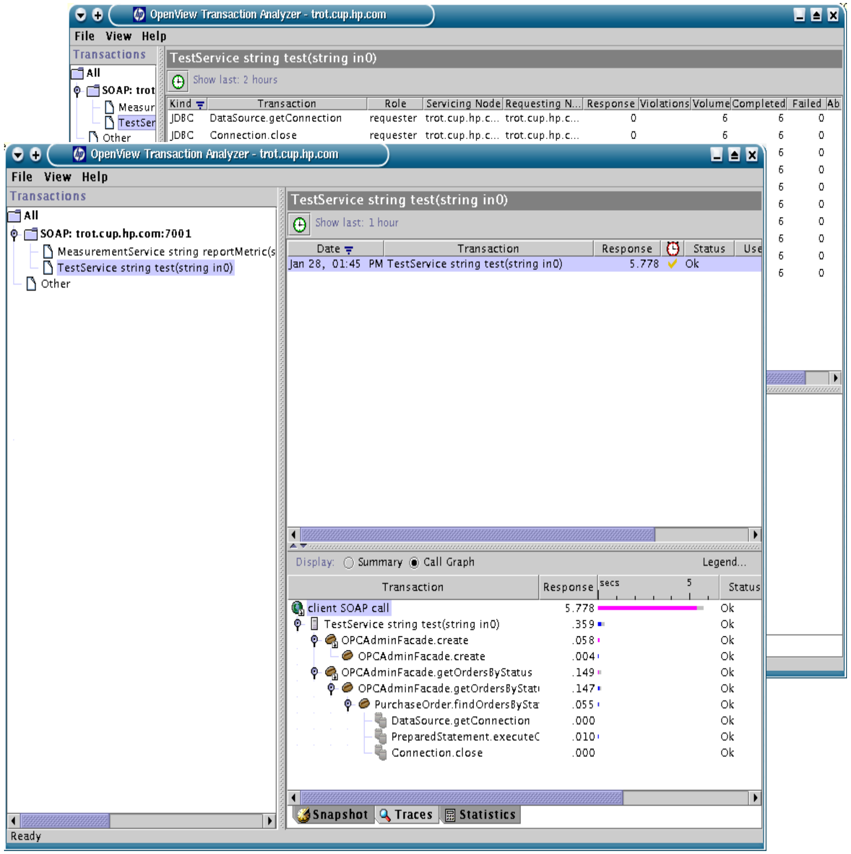
- Review:
  - outlined a process for creating web services
  - presented some open source tools that could be used
- Our key learnings:
  - were a few technical hurdles that had to be overcome
  - we found these tools were a boost to our productivity
  - surprised by the integration between tools
  - process to locate/install Linux packages straightforward
  - tools generally worked out of the box

Overall, the open source environment was very reliable, stable, and usable for building web services

What’s Next?

Monitoring, managing, and tracking
the web services platform and web services

- HP Openview offers products that integrate with Apache Axis:
  - OV SPI for Apache Axis captures information about the web service platform
  - OVTA will support Axis for diagnosing performance bottlenecks in web services



OpenView Transaction Analyzer - trot.cup.hp.com

TestService string test(string in0)

Show last: 2 hours

Kind	Transaction	Role	Servicing Node	Requesting N...	Response	Violations	Volume	Completed	Failed	Ab
JDBC	DataSource.getConnection	requester	trot.cup.hp.c...	trot.cup.hp.c...	0		6	6	0	
JDBC	Connection.close	requester	trot.cup.hp.c...	trot.cup.hp.c...	0		6	6	0	
								6	0	
								6	0	
								6	0	
								6	0	
								6	0	
								6	0	
								6	0	
								6	0	

OpenView Transaction Analyzer - trot.cup.hp.com

File View Help
Transactions
All
SOAP: trot.cup.hp.com:7001
MeasurementService string reportMetric(s
TestService string test(string in0)
Other

TestService string test(string in0)

Show last: 1 hour

Date	Transaction	Response	⏰	Status	Use
Jan 28, 01:45 PM	TestService string test(string in0)	5.778	✓	Ok	

Display: ○ Summary ● Call Graph

Transaction	Response	secs 5	Status
client SOAP call	5.778		Ok
TestService string test(string in0)	.359		Ok
OPCAdminFacade.create	.058		Ok
OPCAdminFacade.create	.004		Ok
OPCAdminFacade.getOrdersByStatus	.149		Ok
OPCAdminFacade.getOrdersBySta	.147		Ok
PurchaseOrder.findOrdersBySta	.055		Ok
DataSource.getConnection	.000		Ok
PreparedStatement.executeC	.010		Ok
Connection.close	.000		Ok

References

- Linux and HP (**www.hp.com/linux**)
  - for more information about HP’s Linux strategy
- HP OpenView (**www.openview.hp.com**)
  - for more information about OpenView support for web services and open source
- HP DSPP Developer Edge (**www.hp.com/go/developers**)
  - for more information about Linux, open source, and Java development
- HP Dev Resource Central (**devresource.hp.com**)
  - for more information about web services development, web services management, and HP’s Eclipse initiatives

HP WORLD 2003

Solutions and Technology Conference & Expo

Interex, Encompass and HP bring you a powerful new HP World.

interex
shared knowledge • shared power

encompass
AN HP USER GROUP

hp
invent