Developing Web Services with Eclipse

Arthur Ryman
IBM Rational
ryman@ca.ibm.com
Abstract

The recently created Web Tools Platform Project extends Eclipse with a set of Open Source Web service development tools and APIs. This talk gives an overview of the project and focuses on its Web services support. The project is divided into two subprojects: Web Standard Tools and J2EE Standard Tools. The Web Standard Tools subproject contains support for XML Web Services, including tools based on standards defined by W3C, OASIS, WS-I and others. The J2EE Standard Tools subproject contains support for standards defined by JCP, such as JAX-RPC and JSR-109, and for reference implementations of these standards, such as Apache Axis.

The project contains both a set of tools for Web service developers and a set of APIs for Web service tool creators. The talk includes a demonstration of the tools.

This is an Introductory Level talk. It assumes some knowledge of Eclipse, Java, XML, and Web services.
My Background

- Software Development Manager at IBM Toronto Lab
  - Focus on Web Service, XML, and J2EE Tools
  - Rational Application Developer V6
  - WebSphere Studio Application Developer V4, V5
  - VisualAge for Java V1, V2, V3
- Leader of Web Standard Tools subproject, Eclipse Web Tools Platform project
- Editor of W3C WSDL 2.0 Core Language specification
- Committer on Apache Woden Project
Topics

- Eclipse Web Tools Platform (WTP) Project
- WTP Web Service Tools
- Web Service Programming Examples
- Call for Participation
Eclipse Web Tools Platform Project
Eclipse Web Tools Platform Project

- See: http://eclipse.org/webtools
- Extends the Eclipse Platform with tools and APIs for Web and J2EE application development
- Formally launched in June 2004
- Has two subprojects:
  - Web Standard Tools (WST)
  - J2EE Standard Tools (JST)
- Includes tools for HTML, XML, Web Services, J2EE, Data
- Includes Server tools for integrating application servers, e.g. Tomcat, Geronimo, JBoss, WebSphere, WebLogic
WTP Subproject Scopes

De Jure Standards
- HTML, XML, XSLT, CSS, JS, WSDL, SOAP, UDDI
- Servlet, JSP, EJB, JAX-RPC, JDBC, JAXP, JSF, J2EE

De Facto Standards
- SQL
- Java Technologies
- Web Technologies
- PHP
- Struts
- Hibernate
- Apache
- ObjectWeb
- SourceForge
WTP Architecture

- JST
  - J2EE Server
  - J2EE Project
  - JSP
  - Servlet
  - EJB
  - J2EE WS

- WST
  - Server
  - Internet
  - HTML
  - XML
  - Data
  - WS

- Eclipse Tools
  - EMF
  - GEF
  - JEM

- Eclipse Platform
  - Resource
  - JDT
  - Debug
WTP Roadmap

- WTP 0.7, July 2005 – End User Tools
- WTP 1.0, December 2005 – Platform APIs
- WTP 1.5, June 2006 – Java EE 5.0
The IBM Software Development Platform

- Analyst
  - WebSphere Business Integration Modeler & Monitor
  - Rational Software Architect

- Developer
  - Rational Web/App Developer
  - WebSphere Tools

- Tester
  - Rational Functional & Manual Tester
  - Rational Performance Tester

- Deployment Manager
  - Tivoli Configuration Manager
  - Tivoli Monitoring

- Customer Extensions
  - Project Manager
  - Executive

- 3rd Party ISV Tools
  - Rational Team Unifying Platform
  - Rational Portfolio Manager
WTP Web Services Tools
Web Service Tools in WTP:

WST

- Web Standard Tools
  - Web Service Explorer
  - WSDL/XSD Editor
  - Web Service Wizard
  - WS-I Test Tools
Web Service Tools in WTP:
JST

- J2EE Standard Tools
  - J2EE Explorer
  - JAX-RPC
  - JSR 109
  - Axis 1.2.1
Web Services Explorer: Discovery and Publish Web Services

- **Discovery**
  - Search UDDI Registries
  - Navigate WSIL Documents
  - Import WSDL into development project

- **Test**
  - Dynamic invocation based on WSDL
  - View SOAP messages

- **Publish**
  - Publish WSDL into UDDI Registries
Web Services Explorer Demo

1) Open Web Services Explorer
2) Open XMethods UDDI Registry
3) Find all stock quote services
4) Select Stock Quote service
5) Open WSDL page and GetQuote for IBM
6) Import WSDL into Workbench StockQuoteClient project as StockQuote.wsdl
Arthur Ryman — Developing Web Services with Eclipse
Details for **XMethods Registry** are shown below. Some items may be added, removed, or changed. Authentication may be required for changes. Other actions are listed at the bottom of this page.

<table>
<thead>
<tr>
<th>Registry Name</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMethods Registry</td>
<td>Edit</td>
</tr>
</tbody>
</table>

**Inquiry URL**

http://uddi.xmethods.net/inquire

**Status**

IWAB0139I The registry named XMethods Registry located at http://uddi.xmethods.net/inquire
Select a result to see more details or select a set of results and click a button to perform an operation.

### Services

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock Quote</td>
<td>IMPLEMENTATION: msnet</td>
</tr>
<tr>
<td>Stock Quotes</td>
<td>IMPLEMENTATION: apachesoap</td>
</tr>
</tbody>
</table>

**Status**

IWAB0149I Number of services found: 2
Details for **Stock Quote** are shown below. Some items may be added, removed, or changed. Authentication may be required for changes. Other actions are listed at the bottom of this page.

**Service Key**
C9E6B64F-426E-D27E-9E0771163564

**WSDL URL**
http://www.webservicex.com/stockquote.asmx?WSDL

**Status**
IWAB0149I Number of services found: 2
Invoke a WSDL Operation

Enter the parameters of this WSDL operation and click Go to invoke.

Endpoints

```
http://www.webservicex.com/stockquote.asmx
```

**symbol** string

IBM

Go Reset

Status

```
<StockQuotes>
<Stock>
<Symbol>IBM</Symbol>
<Last>83.05</Last>
<Date>8/19/2005</Date>
<Time>3:14:10 PM</Time>
</Stock>
</StockQuotes>
```
Web Services Explorer

Actions

Import WSDL To workbench

Select a workbench project to save the WSDL file into. You may also change the name of the WSDL file.

Workbench project: StockQuoteClient

WSDL file name: StockQuote.wsdl

Go  Reset
WSDL/XSD Editor:
Design Web Services

- Graphical and Source editing modes
- Seamless integration for editing inline XSD
- Content Assist
- Pop-up actions
- Binding Wizard
- Validator, including WS-I profiles
- Extendible for WSDL extension elements
WSDL Editor Demo

1) Open **StockQuote.wsdl** in WSDL Editor
2) Go into graphical view of XML schema for messages
3) Navigate into GetQuote element
4) Switch to Source tab
5) Navigate using linked Outline and Properties views
Developing Web Services with Eclipse
Developing Web Services with Eclipse
An XML fragment showing a Web Service definition:

```xml
<s:element name="GetQuote">
    <s:complexType>
        <s:sequence>
            <s:element maxOccurs="1" minOccurs="0" name="symbol" type="s:string"/>
        </s:sequence>
    </s:complexType>
</s:element>

<s:element name="GetQuoteResponse">
    <s:complexType>
        <s:sequence>
            <s:element maxOccurs="1" minOccurs="0" name="GetQuoteResult" type="s:string"/>
        </s:sequence>
    </s:complexType>
</s:element>

<s:element name="string" nillable="true" type="s:string"/>
</s:schema>
```

The `GetQuote` element takes a `symbol` parameter and returns a `GetQuoteResult`. The `GetQuoteResponse` element includes a `GetQuoteResult` response.
Web Services Wizard: Create and Access Web Services

- Supports generate/deploy/test/publish lifecycle
- Configures project, server, and SOAP engine
- Highly extensible
  - SOAP engines
  - Code generators
  - Test facilities
Web Services Wizard:
Create and Access Web Services

- Code generation
  - WSDL to client proxy
  - WSDL to server skeleton
  - Java to WSDL

- Test facilities
  - JSP test client
  - Web Service Explorer
Web Service Client Wizard Demo

1) Run Web Service Client Wizard to generate Java proxy and JSP test page with TCP/IP monitor

2) Wizard adds Web application to Tomcat 5.0 server and installs Axis 1.2.1 SOAP engine

3) Select methods to include in JSP test page

4) Test getQuote() method using IBM

5) View SOAP messages in TCP/IP monitor
Web Services

Review your Web service options and make any necessary changes before proceeding to the next page.

- Client proxy
  
  Client proxy type: Java proxy

- [ ] Test the Web service
- [x] Monitor the Web service
- [ ] Overwrite files without warning
- [x] Create folders when necessary
- [ ] Check out files without warning

< Back  Next >  Finish  Cancel
Arthur Ryman — Developing Web Services with Eclipse
Web Service Client Test
Do you want to test the generated proxy?

- Test the generated proxy

Test Facility: Web service sample JSPs
JSP project: StockQuoteClient
EAR project:
Folder: sampleStockQuoteSoapProxy
JSP folder: /StockQuoteClient/WebContent/sampleStockQuoteSoapProxy

Methods:
- getEndpoint()
- getEndpoint(java.lang.String)
- getStockQuoteSoap()
- getStockQuote(java.lang.String)
Methods

- `getEndpoint()`
- `setEndpoint(java.lang.String)`
- `getStockQuoteSoap()`
- `getQuote(java.lang.String)`

Inputs

symbol: IBM

Result

```xml
<StockQuotes><Stock><Symbol>IBM</Symbol><Last>82.76</Last><Date>8/19/2005</Date><Time><%Value>13:39:49</%Value></Time><%Value>71.85 - 99.10</%Value><%Range>71.85</%Range><%Range>99.10</%Range><%Change>5.119</%Change><%P-E>15.85</%P-E><%Name>INTL BUSINESS MAC</%Name></Stock></StockQuotes>
```
Arthur Ryman — Developing Web Services with Eclipse
WS-I Test Tools:
Test Interoperability of Web Services

- Developed in Eclipse Web Service Validation Tools (WSVT) Project
- Supports WS-I Basic Profile 1.0/1.1, Simple SOAP Binding Profile 1.0, Attachments Profile 1.0
- WSDL 1.1 Validator
- SOAP 1.1 Message Monitor/Analyzer
WS-I SOAP Message Monitor/Analyzer

Normal Message Flow

Requestor

Monitor
- Interceptor
- Logger

Monitor Config File

Web Service

SOAP Messages

XML Schema

Analyzer
- WSDL
- UDDI

Test Assertion Document

Message Log

Conformance Report
WS-I Test Tools Demo

1) Specify level of WS-I compliance in Preferences page
2) Save SOAP messages from TCP/IP to a WS-I log file
3) Specify WSDL file that describes messages
4) View WS-I errors and warnings in Problems view
Specify the WS-I Message Log File Name

Select a folder and specify a message log file name

Enter or select the parent folder:

StockQuoteClient

- StockQuoteClient
  - JavaSource
  - WebContent

File name: log.wsimsdl

Advanced >>
Arthur Ryman — Developing Web Services with Eclipse
J2EE Web Services: Deploy Web Services

- Web Services appear as first class objects in J2EE Explorer
- Content assist for deployment descriptor source editors
- JAX-RPC code generators
- JSR 109 support
- Axis 1.2.1 adaptor
Web Service Programming
Examples
Development Scenarios

- Accessing Web Services
- Creating Web Services
  - Bottom-Up
  - Top-Down
Accessing Web Services

- The preceding demo generated a JSP test client for the Stock Quote service.
- We’ll now code a JSP client application that accesses it.
- The Web Service wizard generated JAX-RPC compliant client code and a convenience wrapper.
- We’ll use this code in our client.
JAX-RPC Client Code
JAX-RPC Client Code

- package NET.webserviceX.www
  - targetNamespace="http://www.webserviceX.NET/
- StockQuoteLocator.java – Service locator
- StockQuote.java – Service interface
  - <wsdl:service name="StockQuote">
- StockQuoteSoap.java – Remote interface
  - <wsdl:portType name="StockQuoteSoap">
    - <wsdl:operation name="GetQuote">
- StockQuoteSoapStub.java – Client stub wrapper for Call object
- StockQuoteSoapProxy.java – Client proxy convenience wrapper
Web Client Version 1:

getQuote.jsp

1) Create a JSP
   1) get a query parameter, “symbol”,
   2) create a service proxy, and
   3) invoke the “getQuote” operation

2) Select getQuote.jsp and invoke Run As->Run on Server
   1) The Web app is added to the server,
   2) the server is started, and
   3) a Web browser is opened on the appropriate URL for getQuote.jsp
J2EE - getQuote.jsp - Eclipse SDK

```html
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Get Quote</title>
</head>
<body>
<h1>Get Quote</h1>

String symbol = request.getParameter("symbol");
String default_symbol = symbol == null ? "IBM" : symbol;

<p>Enter a stock symbol and click the button to get the latest price.</p>
<form action="getQuote.jsp">
Symbol: <input type="text" name="symbol" value="" size="4"> <input type="submit" value="">

String price = "n/a";
if (symbol != null) {
    NET.webserviceX.www.StockQuoteSoapProxy proxy =
        new NET.webserviceX.www.StockQuoteSoapProxy();
    price = proxy.getQuote(symbol);
}

Price: <%= price %>
</form>
</body>
</html>
```
Get Quote

Enter a stock symbol and click the button to get the latest price.

Symbol IBM Get Quote

Price: IBM 83.238/19/2005 11:21am+2.0881 3683.3081 362981200132.8B81.15+2.56%71 85 - 99.105.11915.85INTL BUSINESS MAC
Processing XML

- This service has a poorly designed interface
  - XML is returned as an escaped string
  - No schema for result
- The client needs to parse the result to extract the price, *etc.*
- Client application can parse XML using:
  - DOM,
  - SAX,
  - Java data binding code (JAXB, *etc.*), or
  - server-side XSLT
- For fun, we’ll use browser-side XSLT
Web Client Version 2: \textit{getQuote-xsl.jsp}

1) View example of XML response \texttt{ibm-quote.xml}

2) Create XSLT \texttt{StockQuotes.xsl}
   1) Generate \texttt{<form>} to get symbol
   2) Generate \texttt{<table>} to present stock quote

3) Create JSP \texttt{getQuote-xsl.jsp}
   1) Insert \texttt{<?xml-stylesheet?>} processing instruction
   2) Return unparsed XML

4) Run on Server to view result
<?xml version="1.0" encoding="utf-8"?>
<StockQuotes type="text/xsl" href="StockQuotes.xsl">
  <Stock>
    <Symbol>IBM</Symbol>
    <Last>61.30</Last>
    <Date>8/16/2005</Date>
    <Time>4:02pm</Time>
    <Change>-1.20</Change>
    <Open>82.07</Open>
    <High>82.43</High>
    <Low>81.09</Low>
    <Volume>4395200</Volume>
    <MktCap>129.7B</MktCap>
    <PreviousClose>82.50</PreviousClose>
    <PercentageChange>-1.45%</PercentageChange>
    <AnnRange>71.85 - 99.10</AnnRange>
    <Earnings>5.119</Earnings>
    <P-E>16.12</P-E>
    <Name>INTL BUSINESS MAC</Name>
  </Stock>
</StockQuotes>
<xsl:template match="Stock">
  <tr>
    <td>
      <xsl:value-of select="Symbol" />
    </td>
    <td>
      <xsl:value-of select="Last" />
    </td>
    <td>
      <xsl:value-of select="Date" />
    </td>
    <td>
      <xsl:value-of select="Time" />
    </td>
  </tr>
</xsl:template>
<?xml version="1.0" encoding="utf-8"?>
<xml-stylesheet type="text/xsl" href="StockQuotes.xsl"/>

String quote = "<StockQuotes/>";
String symbol = request.getParameter("symbol");
if (symbol != null) {
    quote = proxy.getQuote(symbol);
}

<%= quote %>
Get Quote

Enter a stock symbol and click the button to get the latest price.

Symbol: IBM  Get Quote

Symbol Last Date Time
IBM  83.27  8/19/2005  11:40am
Bottom-Up Web Service Creation

- Any “reasonable” Java class can be easily deployed as a Web service
- This approach is very appealing to Java programmers since it lets them be immediately productive
- The WSDL is generated from the Java
- The result is acceptable if the methods use “tame” argument types, however object graphs are problematic
- Top-Down design is recommended to achieve the cleanest and most interoperable Web service interfaces
Bottom-Up Service: BUService

1) Create a new Web project: BUService
2) Create a data object to represent the result: BUStock.java
3) Create a business object to take a symbol and return a stock quote for it: BUQuoter.java
4) Use the Web service wizard to deploy it. Use rpc-encoded style for fun so we can see WS-I errors. Generate and monitor a JSP test client.
5) Test the service, view the messages in the monitor, and validate the SOAP messages for WS-I conformance. Note the errors caused by rpc-encoded style.
Web Services

Review your Web service options and make any necessary changes before proceeding to the next page.

- Service
  - Web service type: Bottom up Java bean Web Service
  - Start Web service in Web project
  - Launch the Web Services Explorer to publish this Web service to a UDDI Registry
  - Generate a proxy
    - Client proxy type: Java Proxy
  - Test the Web service
  - Monitor the Web service
  - Overwrite files without warning
  - Create folders when necessary
  - Check out files without warning
  - Do not show me this dialog box again.
Web Service

Service Deployment Configuration

Choose from the list of runtimes and deployment servers, or use the default settings.

Select the service project and the EAR project with which you want it to be associated. If an EAR or project does not exist or is currently unassociated, it will be created and associated as required when you click Next.

Server-Side Deployment Selection:

- Web service runtime: Apache Axis
- Server: Tomcat v.5.0 Server @ localhost
- J2EE version: 1.4

Service project: BService
Service module: BService
Service EAR project:
Service EAR module:

Client-Side Environment Selection:

- Web service runtime: Apache Axis
- Server: Tomcat v.5.0 Server @ localhost
- J2EE version: 1.4

Client type: Web
Client project: BServiceClient
Client module: BServiceClient
Client EAR project:
Client EAR module:
Web Service Java Bean Identity

Web Service Java Bean Identity

Web service URI: http://localhost:8080/BUService/services/BUQuoter
WSIDL Folder: D:/workspaces/CS5/BUService/WebContent/wsdl
WSIDL File: BUQuoter.wsdl

Methods

- ✓ getQuote(java.lang.String)

Select All | Deselect All

Style and Use

- Document/Literal (Wrapped)
- Document/Literal
- ✓ RPC/Encoded

Define custom mapping for package to namespace.

< Back | Next > | Finish | Cancel
Methods

- `getEndpoint()`
- `setEndpoint(java.lang.String)`
- `getBUQuoter()`
- `getQuote(java.lang.String)`

Inputs

- `symbol`: IBM

Result

```
return:

  name: International Business
  last: 100.0
  symbol: IBM
  dateTime: Aug 21, 2005
```
Top-Down Web Service Creation

- Business is transacted by exchanging documents – purchase orders, receipts, application forms, insurance claims, building permits, etc.
- For best interoperability, treat Web services as document interchange, not distributed objects
- Model documents using XSD, and operations using WSDL
- Generate Java from WSDL
Top-Down Service: TDSERVICE

1) Create a new Web project: TDSERVICE
2) Create an XML schema for the Stock quote result: **TDStock.xsd**
3) Create a WSDL for the quote service with an operation that takes a symbol and returns a quote: **TDQuoter.wsdl**. Use the WSDL Binding wizard to generate document-literal SOAP binding style this time.
4) Use the Web service wizard to generate the service.
5) Fill in the implementation of the service: **TDQuoterSOAPImpl.java**
6) Create a new Web project: TDSERVICEClient
7) Select /wsdl/TDQuoterSOAP.wsdl, create a JSP test client, and monitor the service.
8) Test the service and validate the SOAP messages for WS-I conformance. There should be no errors this time.
Arthur Ryman — Developing Web Services with Eclipse
Web Services

Review your Web service options and make any necessary changes before proceeding to the next page.

Service

- Web service type: Top down Java bean Web Service
  - Start Web service in Web project
  - Launch the Web Services Explorer to publish this Web service to a UDDI Registry

- Generate a proxy
  - Client proxy
    - Client proxy type: Java Proxy

- Test the Web service
- Monitor the Web service
  - Overwrite files without warning
  - Create folders when necessary
- Check out files without warning

- Do not show me this dialog box again.
* TDQuoterSOAPImpl.java

```java
package org.example.www.TDQuoter;

import java.util.Calendar;

public class TDQuoterSOAPImpl implements org.example.www.TDQuoter.TDQuoter_PortType {
        String symbol = getQuoteRequest.getSymbol();
        float last = 100;
        Calendar dateTime = new GregorianCalendar();
        String name = "International Business Machines";

        Stock stock = new Stock(symbol, last, dateTime, name);
        GetQuoteResponse response = new GetQuoteResponse(stock);

        return response;
    }
}
```
**Methods**

- `getEndpoint()`
- `setEndpoint(java.lang.String)`
- `getTDQuoter_PortType()`
- `getQuote(org.example.www.TDQuoter)`

**Inputs**

`getQuoteRequest`:

- `symbol`: IBM

**Result**

`returnp`:

- `stock`:
  - `name`: International Business Machines
  - `last`: 100.0
  - `dateTime`: Aug 21, 2005
  - `symbol`: IBM
Developing Web Services with Eclipse

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <getQuoteRequest xmlns="http://www.example.org/TDQuote"
      xmlns:ns1="http://www.example.org/TDQuote" xsi:type="xsd:string">
      IBM</ns1:Symbol>
      <ns1:Last>100.0</ns1:Last>
      <ns1:DateTime>2005-06-21T8:21:06.390Z</ns1:DateTime>
      <ns1:Stock>International Business Machines</ns1:Name>
    </getQuoteRequest>
  </soapenv:Body>
</soapenv:Envelope>
```
Summary

- The Eclipse Web Tools project has a rich, extensible set of Web service tools
  - Explorer, Graphical WSDL and XSD editors, Wizard, WS-I Test Tools
- Discovery, explore, and access remote services
- Create services from Java (bottom-up)
- Create services from WSDL and XSD (top-down)
- Check for WS-I conformance
Call for Participation

- Become a WTP user and tell your friends
- Test WTP and report bugs
- Write tutorials, articles
- Fix bugs
- Contribute enhancements
- Become a committer
- Develop plug-ins based on WTP
- Attend EclipseCon 2006