J2EE Development with Rational XDE and IBM WebSphere Studio Application Developer

Version 2.2

Khawar Z. Ahmed

Rational Software Corporation White Paper
TP197A, 6/02

Rational®
the software development company

Table of Contents

Introduction

While Integrated Development Environments (IDEs) continue to make great strides in the facilities offered to Java software developers, today's software systems demand the power of capabilities such as visual modeling to meet the growing challenges posed by complexity and sheer scale.

Rational XDE extends the integrated development environment provided by IBM WebSphere Application Developer (WSS AD) by providing new and unique capabilities for the developers that can be accessed and used from within IBM WSS AD in a frictionless manner.

This paper discusses Rational® XDE™ 2002 Release 2 in the context of IBM WSS AD 4.0.3, and how to effectively leverage the capabilities of each for specific tasks carried out by J2EE developers. We walk-through a simple but complete J2EE application development consisting of HTML pages, JSPs, servlets and EJBs.

Overview of Rational XDE 2002

Rational XDE is an eXtended Development Environment. It brings world-class modeling capabilities to the fingertips of Java/J2EE developers by running within the IBM WSS AD IDE.

Here are some highlights of Rational XDE 2002:

- *Frictionless Design and Development:* With Rational XDE and IBM WSS AD, modeling becomes as much a part of the IDE as the code editor and debugger. Since you are using the same menus, gestures, and usage metaphors, the learning curve is accelerated. There is no need to switch between applications. Such a seamless integrated approach promotes the use of design, code generation and forward/reverse engineering as a daily function.
- *User-controllable Synchronization:* Rational XDE offers complete forward and reverse engineering support, and (user-configurable manual or auto-synchronization). You can use whichever approach suits your specific circumstances best and you are not restricted by artificial limitations imposed by the tool vendor.
- *Custom Patterns:* Rational XDE offers some powerful capabilities that allow the harnessing of the knowledge base within an individual developer or an entire organization. For example, Rational XDE offers a powerful patterns capability that supports visual creation of custom patterns and wizard driven application of such patterns. Various built in patterns, such as Gang Of Four (GoF patterns) are also available to ensure that you never have to start with a blank slate again. You can even share your patterns with others in RAS (reusable asset format)!
- *Code Templates:* Rational XDE also offers a code templates capability that can be used to automate tasks that need to be repeated. Combined, the patterns and code templates capabilities are so extensive that you can create complete applications entirely from a combination of patterns and code templates.
- *Free-form modeling:* Communication is often times one of the strongest needs of a software team. You can use Rational XDE's ability to create free-form models for improved communication of ideas with others.
- *Multiple-models:* Rational Software not only allows you to have multiple projects and models open simultaneously, you can also utilize the multi-models capability in Rational XDE to work at different levels of abstraction, as needed, while still maintaining traceability across model elements.

And of course, Rational XDE offers various other capabilities to IBM WSS AD developers. For more information, or to try out Rational XDE firsthand, you can visit http://www.rational.com/products/xde/index.jsp and download a free evaluation copy of Rational XDE.

In short, with Rational XDE, you are able to liberate yourself from the productivity-killing tasks you face every day with a faster, more integrated and more enjoyable software development experience.

IBM WebSphere Studio Application Developer Highlights

IBM WebSphere Studio Application Developer (IBM WSS AD) for Windows is designed for professional developers of Java and J2EE technology based applications.

IBM WSS AD highlights include:

- An integrated development environment for building J2EE applications consisting of HTML pages, JSP, servlets, and EJB
- An open, standards-based J2EE development environment
- A collaborative development environment with role-based perspectives for every team member
- A scalable and adaptable platform to meet growing application development needs
- Integration with best-of-breed plug-ins
- End to end local and remote testing

For more information about IBM WSS AD, please http://www.ibm.com/software/ad/studioappdev/

Getting Started

As Rational XDE runs within the IBM WSS AD, getting started with Rational XDE is very straightforward. You simply create an appropriate project, just as you would do when working with just WSS AD.

IBM WSS AD provides various types of projects. Among them, EJB, Web, and J2EE projects are most relevant to J2EE application development. As the name implies, an EJB project sets up a project that is used for Enterprise JavaBeans. Similarly, the web project is appropriate for applications that consist of web artifacts such as JSP, servlet, and/or HTML pages, along with any supporting Java classes. If your application requires both EJB and web projects, a better option is to use the J2EE project option. It actually creates up to four different projects (EJB, Web, Client Application and an Enterprise Project) depending on the options you specify.

To simplify your modeling setup, Rational XDE provides modeling projects that parallel the WSS AD. These are accessed via the Modeling projects category in the WSS AD project creation wizard.

We also need to use the appropriate J2EE version since Rational XDE supports J2EE 1.2 as well as J2EE 1.3 whereas IBM WSS AD currently only supports J2EE 1.2. By default, when Rational XDE is used with IBM WSS AD, it sets the J2EE version to J2EE 1.2.

Since the focus of this document is to show J2EE application development with Rational XDE and IBM WSS AD, we will start off by creating a J2EE project and setting up Rational XDE for use with IBM WSS AD.

Steps to create the project:

1. Go to the main WSS AD menu bar and choose File>New>Project.
2. In the wizard, choose Modeling on the left and Enterprise Application Modeling Project on the right. Click Next.
3. In the Enterprise application project name field, type in XDEProject as the name. XDE automatically fills in the other project names.
4. Uncheck the Application client project name checkbox. Click Finish.
5. Go to Window>Preferences>Rational XDE>Code-Model Synchronization and ensure that AutoSync is checked and set to Review the Conflict option. Click Apply.

At this point, your navigator should have the project structure similar to the one shown in figure 1.

Figure 1: Project structure

Setting up a projects for modeling

To model J2EE applications within IBM WSS AD, you need to create one or more Rational XDE projects that contain models within the project. Rational XDE provides basic model types that serve this purpose and creates them automatically for you if you use the Modeling projects category when creating new projects.

Since we created an Enterprise application modeling project, we already have models defined within each project. No additional steps are required for modeling.

The following summarizes the model types and their usage:

- *Java Code Model:* Appropriate for EJBs, servlets, Java Beans and other Java language artifacts.
- *Java Content Model:* Used for modeling artifacts that do not require code generation and synchronization. Typically used for modeling patterns. We will not use this type of model in our sample application.
- *Virtual Directory Model:* Suitable for modeling web artifacts such as JSP and HTML pages.
- *JSP Tag Lib Model:* As the name implies, used for modeling JSP tag libraries. We will not use this type of model in our sample application.

You can also add additional models to a project if you need to. This is done as follows:

- In the Navigator, select the desired project
- Right click and choose New>Model
- In the dialog, choose the category on the left and the type of model on the right
- Click Finish

At this point, the contents of your model explorer view should look similar to that in Figure 2.

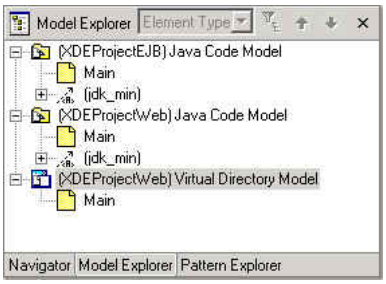

Figure 2: Models required for the Sample Application

We also need to setup J2EE as a reference model so Rational XDE knows where to look for the classes provided as part of the J2EE SDK and resolve references appropriately. You need to do the following: To add the J2EE 1.2 Framework Model as a reference model to each Java Code Model:

1. In the Model Explorer, right click on (XDEProjectEJB) Java Code Model and select More Java Actions>Referenced Models
2. In the Referenced Models dialog Other Models pane, click the plus sign above the pane
3. In the resulting Select Reference Models dialog, click the Browse button

4. Navigate to and select: C:\Program Files\Rational\XDE\Addins\J2EE\J2EE 1.2 Framework Model\J2EE 1.2 Framework.mdx
5. Click Open
6. Provide a description for the reference model e.g. J2EE1.2 Model
7. Click OK
8. Highlight J2EE1.2 Model in the Other Models pane
9. Click the button in the middle of the dialog to add the J2EE1.2 Model to the Referenced Models list on the left
10. Click OK
11. In the dialog asking if you want to synchronize all cross-model relationships at this time, click Yes(Synchronize)
12. Repeat for (XDEProjectWeb) Java Code Model, skipping steps 2-7

Designing the user interface

IBM WSS AD and Rational XDE provide an ideal extended development environment for designing and implementing web components comprised of HTML pages and JavaServer Pages (JSP). You model the components in Rational XDE's modeling perspective and then switch to the web perspective for taking care of the detailed implementation and testing.

For our sample application, we require a rather simple interface that consists of the following:

- An HTML page with a form for obtaining the user inputs
- A JSP to build the response based on results provided via the JavaBean

By convention, IBM WSS AD requires that all JSP and HTML pages in a web project be placed in a package called *webApplication*. Rational XDE creates this for you automatically, so you don't need to do anything here. Recall that since we are dealing with JSP and HTML pages, we need to use the Virtual Directory Model. We also create a diagram within our model for modeling the user interface. Creating the appropriate JSP/HTML model element on the diagram is easily done using the Rational XDE web toolbox. The resulting diagram is shown in Figure 3.

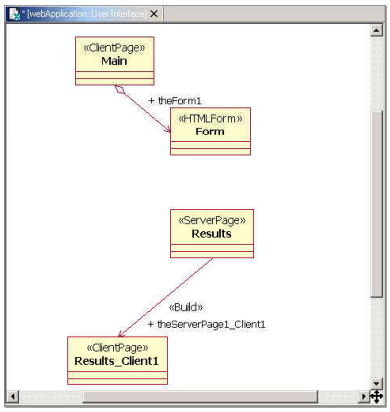

Figure 3: User Interface components

Steps to create the user interface:

1. In the Model Explorer, select (XDEProjectWeb)Virtual Directory Model
2. Right click and choose Add Diagram>Class
3. Rename the diagram User Interface
4. Click on the Web toolbox
5. Drag and drop an HTML Form on to the User Interface diagram
6. In the Add Form dialog, type in Main in the Owner field and leave other options as is
7. Click OK. This creates a client page Main that contains the HTML form Form1 model element
8. Drag and drop a Server Page on to the User Interface diagram

9. Rename it Results
10. In the Model Explorer, expand the Results model element
11. Drag and drop the Results_Client1 page nested in Results server page on to the User Interface diagram

Form processing

The sample application also requires some supporting classes that will be used to handle the form submission and results. Specifically, we will use:

- A servlet to act as a controller and handle the incoming form and obtain data required for the response (we will deal with the actual processing of the input later)
- A JavaBean to store the results obtained by the servlet and passed on to the JSP

In IBM WSS AD, supporting Java classes such as servlets and JavaBeans need to be placed in the *source* directory of the web project. Rational XDE automatically takes care of creating this folder and mapping the code generation to it when you create the modeling project. All you need to do is to create a servlet and a Java class in the Java Code Model of the web project. You can do so using Rational XDE's Java toolbox or via the model explorer, and then just reference them as required in the Virtual Directory Model. The updated diagram is shown in Figure 4.

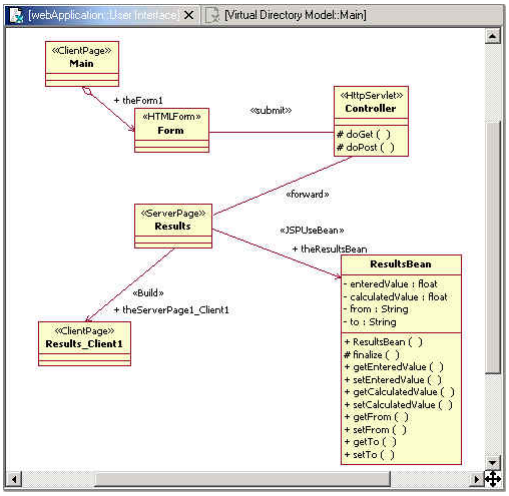

Figure 4: Updated diagram with servlet and JavaBean

Steps to create the servlet and JavaBean:

1. In the Model Explorer, select (XDEProjectWeb) Java Code Model
2. Right click and choose Add Java>HTTP Servlet
3. In the Model Explorer, rename the servlet to Controller
4. Select the (XDEProjectWeb) Java Code Model
5. Right click and choose Add Java>Class
6. Rename the class to ResultsBean
7. Drag and drop Controller from the Model Explorer to the User Interface diagram
8. Drag and drop ResultsBean from the Model Explorer to the User Interface diagram
9. In the User Interface diagram, select the ResultsBean
10. Right click and choose Add Java>Bean Property
11. In the bean property dialog, type in enteredValue for name and set Type Expression to float
12. Create the following additional bean properties: calculatedValue of type float, to of type String, from of type String

13. In the Web toolbox, select the JSPUseBean Relationship tool
14. Draw a relationship from the Results server page to the ResultsBean JavaBean
15. In the Add UseBean dialog, set Scope to session. Use defaults for the rest
16. Click OK
17. In the UML Class toolbox, select the Association tool
18. Draw a relationship from the Form to the Controller and from the Controller to the Results server page[1]
19. Select the association between the form and the Controller and via the properties editor, type in *submit* in the stereotype field
20. Select the association between the Controller and the Results server page and via the properties editor, type in *forward* in the stereotype field
21. In the Model Explorer, select (XDEProjectWeb)Virtual Directory Model
22. Right click and choose Generate Code

EJB Modeling

You can use Rational XDE and IBM WSS AD together to model and implement both entity beans and Session Beans. Rational XDE provides specific capabilities that simplify creation and implementation of EJBs. For instance, Rational XDE provides you with a simplified representation of the EJB for use in the model, but allows you the flexibility of viewing and manipulating the individual EJB pieces as required. As another example, consider the addition of a business method. In Rational XDE, when you add a business method using Add Java>Business method action, it automatically adds it to the EJB implementation class and the remote interface and it keeps them synchronized as you make changes.

Certain conventions need to be adhered to when you use IBM WSS AD with Rational XDE for EJB creation and development. Specifically:

- You should create and place the EJBs in their own sub-package and place the sub-package in the ejbModule package
- For consistency, it is best to follow the established Java convention of specifying a package structure for avoiding naming conflicts <org type>.<org name>.<application name> e.g. com.rational.xde
- You must synchronize the XDE modeling perspective with the IBM WSS AD provided J2EE perspective to ensure that EJB deployment descriptor changes are appropriately propagated.

In our sample application, although the Controller servlet could just as easily handle the calculations required for the response, for the purposes of illustrating the capabilities of Rational XDE and IBM WSS AD, we will use a stateless session bean to perform the conversions. The MyConversion EJB we use is a simple one. It has two business methods, namely KilometersToMiles that takes a float value as input and returns a float value indicating the calculated value in miles, and MilesToKilometers that takes a float value as input and returns a float value indicating the calculated value in kilometers.

Details of the Conversion EJB are shown in the diagram below.

[1] This and the following steps are optional. These relationships serve to clarify how the different elements relate to each other. No code is generated for these relationships.

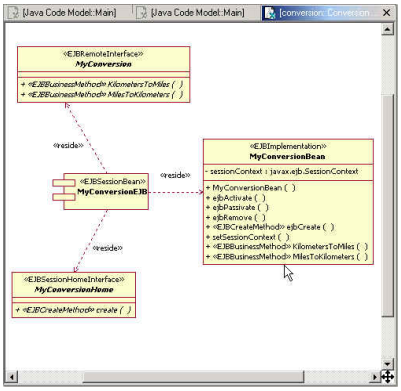

Figure 5: MyConversion EJB

Steps to create the EJB:

1. In the Model Explorer, select (XDEProjectEJB)Java Code Model
2. Right Click and choose Add Java>Package
3. Rename the package com
4. In the Model Explorer, select com package, right click and choose Add Java>Package
5. Rename the package conversion
6. Select the conversion package
7. Right click and choose Add Java>Stateless session bean
8. Rename the EJB MyConversion.
9. In the Model Explorer, expand MyConversion EJB
10. Double click on the MyConversion Overview diagram to open it
11. Select the MyConversion EJB on the MyConversion Overview diagram
12. Select MyConversionEJB component on the diagram
13. Right click and choose Add Java>Business Method
14. Go to the remote interface (or implementation class) on the diagram and click on the newly added business method
15. Rename it to the following: MilesToKilometers(m: float): float
16. Repeat the above and add the following business method: KilometersToMiles(km: float): float
17. Select the MyConversionBean implementation class in the Model Explorer and expand it
18. Locate the MilesToKilometers operation and select it
19. Right click and choose Browse code
20. In the method body, type in the following: return m*1.609347f;
21. Locate the KilometersToMiles method
22. In the method body, type in the following: return km*0.62f;
23. Click Save
24. In the Model Explorer, select the (XDEProjectEJB) Java Code Model
25. Right click and choose Update WSS AD from XDE

Implementing the User Interface

Once you have captured the initial design of the web-based user interface in the modeling perspective, you can use IBM WSS AD's web perspective to refine and complete your user interface. IBM WSS AD's web perspective provides a Page Designer that allows you to graphically layout HTML pages as well as JSPs, view and edit the source code for it and preview it within the IBM WSS AD environment.

You can also continue to take advantage of the modeling aspects of HTML pages as well as JSP because changes you made in the web perspective are reflected in the modeling perspective as soon as you issue the reverse engineer command in the modeling perspective.

Our sample project is quite simple, and so are the initial HTML pages as well as the page built by the JSP. The completed pages are shown in figure 7and figure 8.

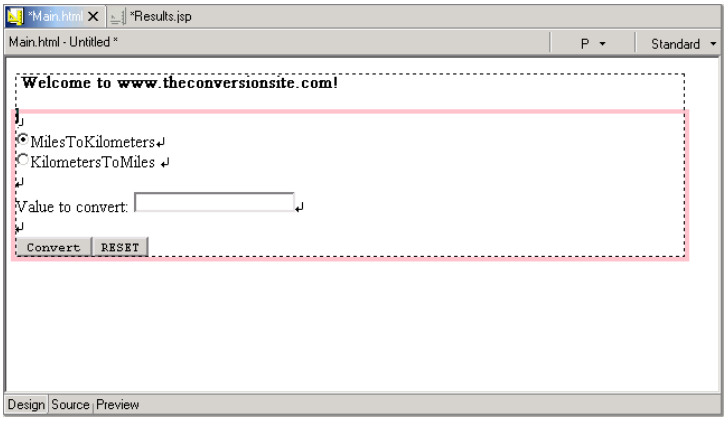

Figure 6: Main Page for accepting user input

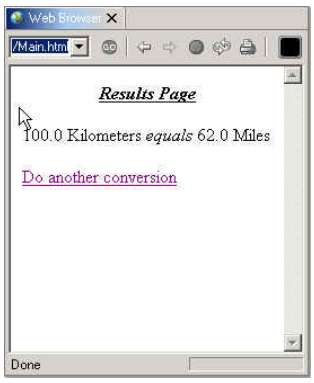

Figure 7: The desired response page

Steps to implement the user interface:

1. Make sure you generated the code (last step in the form processing section)
2. Go to Perspective>Open>Other and choose web
3. Expand XDEWebProject>webApplication>WEB-INF
4. Select Main.html
5. Right click and choose Open with>Page Designer
6. click on the Source tab
7. Copy the contents of Listing 1 at the end of this article (you can also design the page interactively using the Page Designer Design tab) and replace the <form></form> tags entirely in the Main.html source window.
8. Click Save
9. Click on the Design tab to adjust layout as desired
10. Click Save
11. Select Results.jsp
12. Right click and choose Open with>Page Designer
13. click on the Source tab
14. Copy the entire contents of Listing 2 immediately after the UseBean tag

15. Click on the design page to view the page layout (you should see a page similar to one in Figure 8 but with place holders instead of actual values)
16. Click Save
17. Close the Main.html and Results.jsp page design editor windows

Completing the EJB

We have already implemented the logic required for the actual conversions. We will use the IBM WSS AD provided access bean wrappers capability for simplifying the use ofMyConversion EJB. You use the access bean just as you would a Java Bean, thereby making your EJB use simpler. The code generated for the access bean takes care of accessing the EJB, including looking up the EJB, etc.

Thus, completing the EJB essentially consists of generating the wrapper access bean for the EJB.

Steps to complete the EJB:

1. Click on the J2EE perspective
2. Click on the J2EE view tab
3. Expand EJB Modules>XDEProjectEJB
4. Select the MyConversion EJB
5. Right click and choose Open with EJB Extension Editor
6. Click on the bindings tab
7. In the left pane, expand XDEProjectEJB
8. Select MyConversion
9. Make sure the JNDI name at the top of the window is set to: ejb/MyConversionHome
10. Click Save
11. Close the EJB Extension Editor
12. In the J2EE view, reselect MyConversion
13. Right click and choose New>Access Bean
14. Choose Java bean wrapper and click Next
15. In the drop down list, choose XDEProjectEJB and click Next
16. In the Enterprise Bean drop down list, choose MyConversion and click Next
17. Select create() in the drop down list and click Finish

Implementing the servlet controller

Now that we have generated the access bean, implementing the servlet becomes rather simple since there is no need to write EJB lookup and access logic.

The following needs to be done to complete the Controller servlet:

- Obtain the incoming user data
- Call appropriate access bean methods
- Place the results in the ResultsBean
- Forward the request on to the Results JSP to process it

Listing 3a, 3b and 4 show the implementation details for the Controller servlet.

Steps to implement the servlet:

1. Click on the web perspective
2. Expand XDEProjectWeb>source
3. Double click on Controller.java to open it in an editor

4. Copy the source code in listing 3a (variable declaration and init() method code) to within the servlet class definition e.g. immediately after the opening brace for the servlet class
5. Replace the doGet() method with the source code in listing 3b
6. Copy the source code in listing 4 immediately before the last closing brace }. This defines a new method called getResults
7. Choose File>Save
8. Select XDEProjectWeb in the Navigator, right click and choose properties
9. Click on Java build path
10. For required projects on the build path, check XDEProjectEJB and click OK
11. Eliminate any unresolved reference errors by highlighting the unresolved class name in the source code, right-clicking and choosing Add Import from the context menu.
12. Save any changes to source code via File>Save
13. In the navigator, double click on XDEProjectWeb/webApplication/WEB-INF/web.xml file
14. Click on the servlet tab
15. Click Add at the bottom left
16. Select Controller in the list
17. Click OK
18. Click Add near the URL mapping pane
19. Type in Controller for the URL mapping
20. Click Save
21. Close web.xml editor

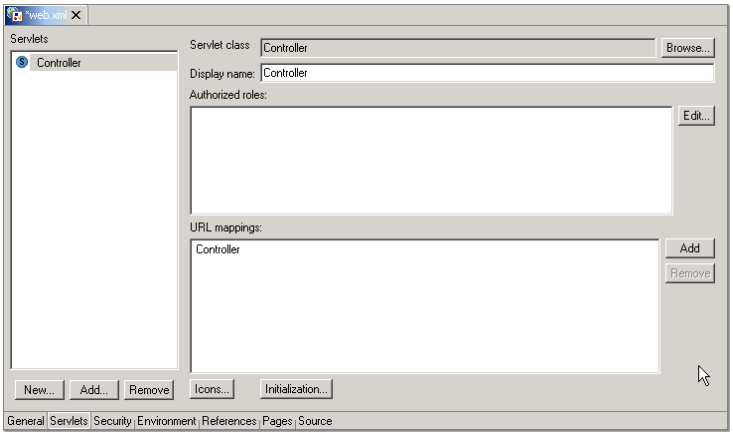

Figure 8: Setting up the Controller servlet URL

Deploying to the WebSphere Test Environment

IBM WSS AD provides a test environment for deploying and testing EJB, servlet and JSP. To simplify EJB testing, it also provides an EJB test client that allows you to instantiate EJBs and test individual methods defined for the EJB via auto-generated JSP pages. This eliminates the need to have or create a web interface simply to test an EJB.

You can also use Rational XDE's deploy capability to deploy to the WebSphere App server from the modeling perspective. It is recommended to use the IBM WSS AD deployment capabilities and the test encvironment when using Rational XDE within IBM WSS AD due to its more extensive support for deployment and testing

Steps to deploy to theWebSphere test Environment:

1. Click on the J2EE Perspective
2. Click on the J2EE view tab
3. Expand EJBModules
4. Select XDEProjectEJB

5. Right click and choose Generate>Deploy and RMIC code
6. In the dialog, check MyConversion
7. Click Finish
8. Right click and choose run on server. Wait till the server startup has completed before proceeding further
9. Click on the Web perspective
10. Expand XDEProjectWeb>webApplication
11. Select Main.html
12. Right click and choose run on server. This will open up the Main page in a browser
13. In the browser, choose one of the business methods
14. Enter a numeric amount in the value to convert field
15. Click submit
16. Click on Do another Conversion to repeat the test with another value or different conversion

Working with Existing Java Code

In this document, we concentrated on working through a complete sample project created from scratch. In real life, you may have situations that require the use and incorporation of existing code into a model.

This is relatively straightforward to do and requires the following steps:

- Create a model file in the project where the source files are located. To do this, select the project, right click and choose New>Model. Choose Java>Java Code Model for EJB and other Java artifacts and Web>Virtual Directory Model for web components such as JSP and Web>JSP Tag Library Model for taglibs.
- Add the desired source files to the model for modeling purposes. To do so, select the model in the Model Explorer and from the context menu, choose More Java Actions>Add/Remove Modeled files. At this point, you can click on Add Files to add additional source files for modeling.

Summary

Rational XDE and IBM WSS AD offer a powerful new combination for developing J2EE applications. This paper only offers a basic introduction to their capabilities. There are numerous other capabilities, such as custom design patterns, free form modeling and code templates that can dramatically improve communication and speed within a team.

You are encouraged to download a Rational XDE evaluation copy from www.rational.com and try it out for yourself to see how Rational XDE and IBM WSS AD can revolutionize your software development activities.

Source Code Listings

Listing 1: Main.html source

```
<form action=Controller method=Get enctype="application/x-www-form-urlencoded"
name=Form1>
<p>
<b>Welcome to www.theconversionsite.com!</b></p>
<p>
<br>
<INPUT TYPE="radio" CHECKED NAME="optionsgroup" VALUE="1">MilesToKilometers<br>
<INPUT TYPE="radio" NAME="optionsgroup" VALUE="2">KilometersToMiles <br>
<br>
Value to convert: <INPUT TYPE="text" SIZE="20" NAME="Value"><br>
<br>
<INPUT TYPE="submit" VALUE="Convert" NAME="Convert"><INPUT TYPE="reset" NAME="Reset"></p>
```

```
</form>
```

Listing 2: Results.jsp source

```
<BODY>
<P align="center"><B><I><u>Results Page<BR>
</u></I></B></P>
<p>
<%=ResultsBean.getEnteredValue()%> <%=ResultsBean.getFrom()%> <I>equals</I>
<%=ResultsBean.getCalculatedValue()%> <%=ResultsBean.getTo()%>
<p>
<P>
<a HREF="Main.html">Do another conversion</a>
</BODY>
```

Listing 3a: Controller.java field declaration and init() source

```
MyConversionAccessBean cabean = null;

public void init() throws javax.servlet.ServletException
{
        cabean = new MyConversionAccessBean();
}
```

Listing 3b: Controller.java doGet() source

```
protected void doGet(javax.servlet.http.HttpServletRequest httpRequest, javax.servlet.http.HttpServletResponse
httpResponse) throws javax.servlet.ServletException, java.io.IOException
   {
        javax.servlet.http.HttpServletRequest req = httpRequest;
        float enteredValue = 1.0f;
        float resValue = 1.0f;
        int option;

        option = new Integer(req.getParameter("optionsgroup")).intValue();
        enteredValue = new Float(req.getParameter("Value")).floatValue();
        ResultsBean rb = new ResultsBean();
        try
        {
                rb = getResults(option, enteredValue);
        }
        catch (Exception e)
        {
                System.out.println("Exception encountered in Controller.getResults method!");
                System.out.println(e.getClass());
                System.out.println(e.getMessage());
                e.printStackTrace(System.out);
        }
        javax.servlet.http.HttpSession session = req.getSession(true);
        session.setAttribute("ResultsBean", rb);

        RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("Results.jsp");
```

```
        dispatcher.forward(req, httpResponse);
    }
```

Listing 4: Controller.java getResults() method definition

```
protected ResultsBean getResults(int opt, float val)
                throws java.rmi.RemoteException, javax.naming.NamingException, javax.ejb.CreateException
        {
        ResultsBean        rbean = new ResultsBean();
        if (opt==1)
                {
                rbean.setCalculatedValue(cabean.MilesToKilometers(val));
                rbean.setTo("Kilometers");
                rbean.setFrom("Miles");
                rbean.setEnteredValue(val);
                }
        else
                {
                rbean.setCalculatedValue(cabean.KilometersToMiles(val));
                rbean.setFrom("Kilometers");
                rbean.setTo("Miles");
                rbean.setEnteredValue(val);
                }
        return rbean;
        }
```

About the author: *Khawar Ahmed is in the Rational XDE Technical Marketing Team, focused on the Java Platform. He is the author of "Developing Enterprise Java Applications with the J2EE and UML", Addison-Wesley, 2001. He can be reached at kahmed@rational.com*

Rational®

the software development company

Corporate Headquarters
18880 Homestead Road
Cupertino, CA 95014
Toll-free: 800-728-1212
Tel: 408-863-9900
Fax: 408-863-4120
E-mail: info@rational.com
Web: www.rational.com

For International Offices: www.rational.com/worldwide