Using Rational XDE v2002 with IBM WebSphere Studio Application Developer v5

Dean Wampler

TP904, 1/2003

Rational®
the software development company

Introduction

Rational XDE™ v2002 installs into IBM WebSphere Studio Application Developer™ v4.0.3 and Application Developer Integration Edition™ v4.1.1. You can also install XDE v2002 "standalone", in which case the WebSphere Workbench Studio, an IBM-supported version of the open-source Eclipse IDE, is installed as well.

Support for the Application Developer v5.0 family of products is under development and not yet available. However, in the interim, it is possible to use XDE with Application Developer v5, if you run XDE separately. This whitepaper describes how to use the two tools together to develop Java and J2EE applications.

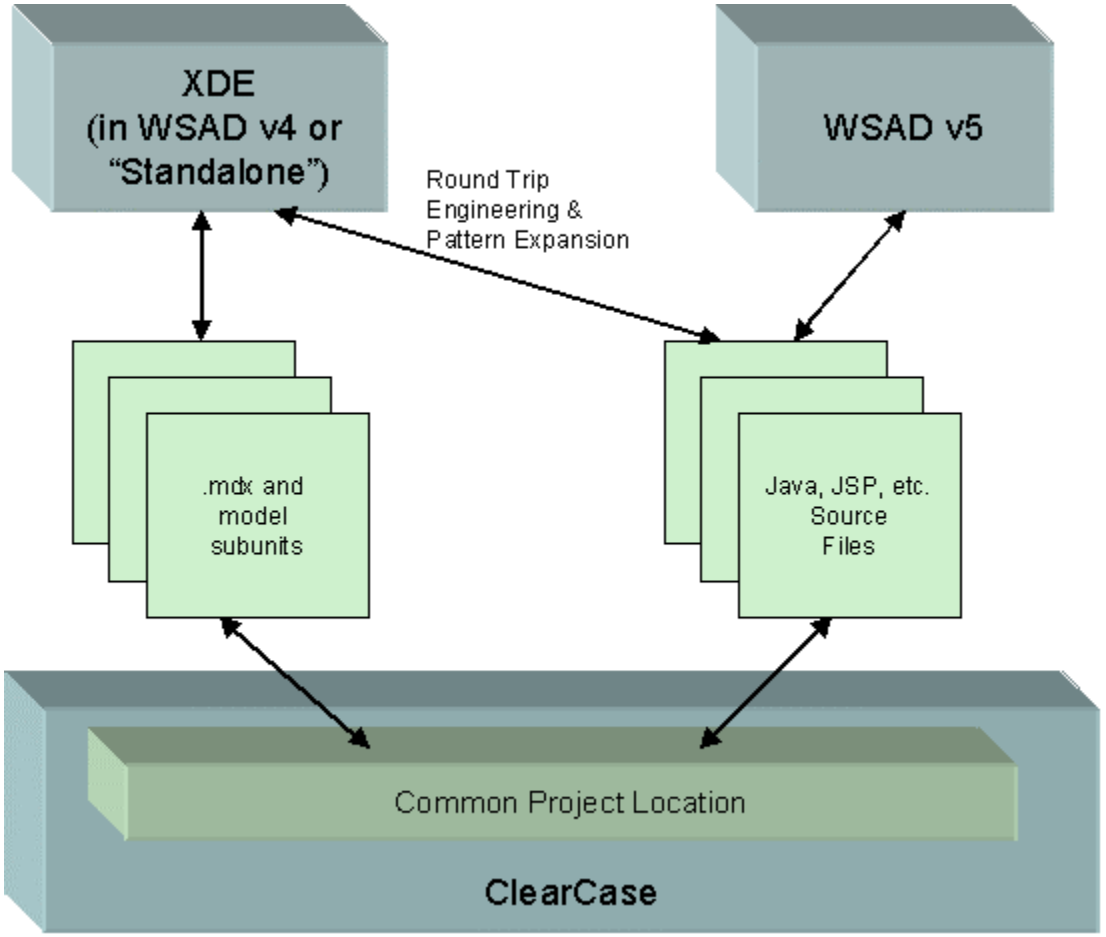
In this arrangement, XDE and Application Developer v5 share the same project directories. XDE is used for all modeling activities, including the application of Patterns and round-trip engineering between source code and models. Application Developer v5 is used for the rest of the code development and deployment.

If you are not using XDE for source code development, then it is easier to use separate projects for XDE and Application Developer v5. In this case, the procedures described in this document are not necessary.

Configuration management is handled through *both* tools. Models are managed in XDE, while all other files are *primarily* managed in Application Developer. However, XDE also needs write permission on source files and deployment descriptors when round-trip engineering is done between models and sources.

If you prefer to use separate projects locations for XDE and Application Developer v5, see Appendix B.

Here is a schematic view of how XDE and Application Developer v5 are used together. Both applications use a common project location, the contents of which are managed in ClearCase. XDE handles its own model files while Application Developer v5 handles source files. However, XDE patterns and round-trip engineering are also used to manipulate source files.



Getting Started

Install Application Developer v5

Install Application Developer v5 by following the directions provided with it. Also, you should check the IBM web site for any applicable updates and bug fixes. See http://www-3.ibm.com/software/ad/studioappdev/support/.

Note that it doesn't matter if you install XDE v2002 or Application Developer v5 first.

Install XDE v2002

If you have not already installed XDE, then install the "standalone" configuration. While you can use XDE inside Application Developer v4.0.3 or inside Application Developer Integration Edition v4.1.1, doing so will only consume extra resources at run time and the added features of these IDE's won't be useful if you are using Application Developer v5.

If you have already installed XDE v2002, either "standalone", inside Application Developer v4.0.3, or inside Application Developer Integration Edition v4.1.1[1], you don't need to make any changes to your installation. Don't uninstall XDE if you currently have it installed inside Application Developer v4.0.3 or Application Developer Integration Edition v4.1.1, with the intention of reinstalling XDE "standalone". The resource savings are not worth the effort.

No matter what configuration you choose, you should download and install the latest Service Release for XDE v2002. It contains important bug fixes and instructions for using XDE with ClearCase. See http://www.rational.com/support/downloadcenter/upgrades/xde.jsp.

Install and Setup Your Configuration Management System

Install your configuration management system and configure both IDE's to use it as appropriate.

For example, if you are using Rational ClearCase™, configure both Application Developer v5 and XDE v2002 to use it, following the instructions provided with each environment. For example, it is important to set up ClearCase so that it understands how to manage XDE Model files. This paper discusses some of the issues concerning ClearCase usage, since it will be necessary to have check-out and check-in capabilities in both Application Developer v5 and XDE v2002 if you work with source code and deployment descriptors in both environments, as opposed to using XDE just for "high-level" modeling. You will also need to create the appropriate "VOB's" (Versioned Object Bases or ClearCase repositories), "views" (ClearCase user workspaces), *etc*. See the ClearCase help in Application Developer for the full details. Additional information is available on the Rational Developer Network (http://www.rational.net). See, for example, "Team Development Support in IBM WebSphere Studio Application Developer Using Rational ClearCase".

Finally, the previously mentioned Service Release for XDE, available at http://www.rational.com/support/downloadcenter/upgrades/xde.jsp, contains important information on using XDE successfully with ClearCase. Please follow its instructions.

Decide on Your Project Structure

Before you begin working with either tool, decide on where you want to locate your projects.

First, let's clarify some terminology. In Eclipse-based IDE's, like XDE and Application Developer, the *workspace* is the directory structure that stores and manages the metadata for the different *projects* under development. The *projects* have their own directory structures, which may or may not be located inside the workspace area; you can put them elsewhere, if you wish.

[1] This whitepaper was prepared while running XDE inside Application Developer Integration Edition v4.1.1, as some of the screenshots will suggest.

So, several points should be considered when structuring your projects.

- XDE v2002 and Application Developer v5 cannot share the same workspace location.
- An XDE project and an Application Developer project *can* share the same location if one of the projects is a "Simple" project. The type of the other project is not restricted. Otherwise, collocation of projects causes "metadata collisions", as both XDE and Application Developer attempt to manage the same project data.
- Since most coding will be done in Application Developer v5, it is best to do most configuration management (CM) from within Application Developer v5, rather than XDE, to minimize confusion.
- Versioning of Model files should be managed from within XDE, especially if the models are decomposed into controlled units or the merging of different versions is required.
- If you are modeling source code and using round-trip engineering, then XDE needs check-in and check-out permissions for the source files.
- The Application Developer v5 *workspace* can be managed under CM, but this is not required. (Note that it still must be in a different location than the XDE workspace.) In either case, the data and artifacts for all *projects* in the workspace should be managed.

So, in this paper I use ClearCase for UCM for the projects, I place the XDE workspace in a "view-private" directory (*i.e.,* not managed by ClearCase, but located in a ClearCase "view"), and I place the Application Developer v5 workspace inside of ClearCase. Also, only "Simple" projects are used in XDE, while Java and Web Projects are used in Application Developer 5. I use the following directories for the workspaces and projects.

Tool	Location
XDE workspace	d:\ viewStore\wsadxde_ws[2]
Application Developer v5 workspace	d:\viewStore\wsadxde\dwamplerTestPVOB\workspace
Projects	d:\viewStore\wsadxde\dwamplerTestPVOB\workspace*myprojects*

"d:\viewStore\wsadxde" is the location of a ClearCase snapshot view I created for this paper. It is a view for my test PVOB (Project Versioned Object Base) called "dwamplerTestPVOB". Hence, "d:\viewStore\wsadxde\ dwamplerTestPVOB" is the root location for all my managed files. The Application Developer workspace is under ClearCase and the projects are under that workspace. The XDE workspace is in a "view private" directory, "d:\viewStore\wsadxde_ws". This directory is in the view's directory structure, but I won't manage it with ClearCase.

Configure Application Developer v5

Start Application Developer

Invoke Application Developer v5 and specify the desired workspace directory in the initial prompt for the workspace location. You can also run the "wsad.exe" command from a command window prompt and specify the argument "-data *workspace_directory*".

Set ClearCase Preferences

If you are using ClearCase as your UCM system, a few preferences should be set in Application Developer v5.

1. Invoke "Windows > Preference".
2. Select "Team > Rational ClearCase".
3. Check the box for "Automatically connect to ClearCase on startup."

Create Your Projects

Create your Application Developer projects in the normal way. In the "File" menu, invoke the "New > Project" command, select the appropriate project type and walk through the New Project Wizard as you normally would. For the project type, follow the guidance in this table:

[2] This is one of the possible workspace locations used by the "xde_cc.bat" driver script that is installed by the XDE 2002 service release. The default location depends on your configuration. The script is discussed more fully below.

Your Project:	Project Category (Left-Hand Side)	Project Type (Right-Hand Side)
Web Site or Web Tier for J2EE Application	Web	Web Project
EJB Tier	EJB	EJB Project
Full J2EE Project with EJB and Web Tiers, *etc.*	J2EE	Enterprise Application Project[3]
Other Java development	Java	Java Project

For larger projects, you may choose to create many projects, partitioned by subsystem or other boundaries. For example, a large J2EE project may have several Web and EJB Projects, all of which are subordinate to the same Enterprise Application Project.

For this whitepaper, I use an example Enterprise Application Project named "LoanApp", for a bank loan application, with subordinate Web and EJB projects named "LoanAppWeb" and "LoanAppEJB", respectively. I also use J2EE 1.3, the latest standard supported by Application Developer v5 and XDE. For completeness, I add a generic Java project of "utilities", called "JavaUtils".

The steps provided in this whitepaper are specific to my example, but they should give you the detailed information you need for your own projects.

If you didn't exit and restart Application Developer v5 after setting the previous ClearCase preference, then manually connect to ClearCase before proceeding.

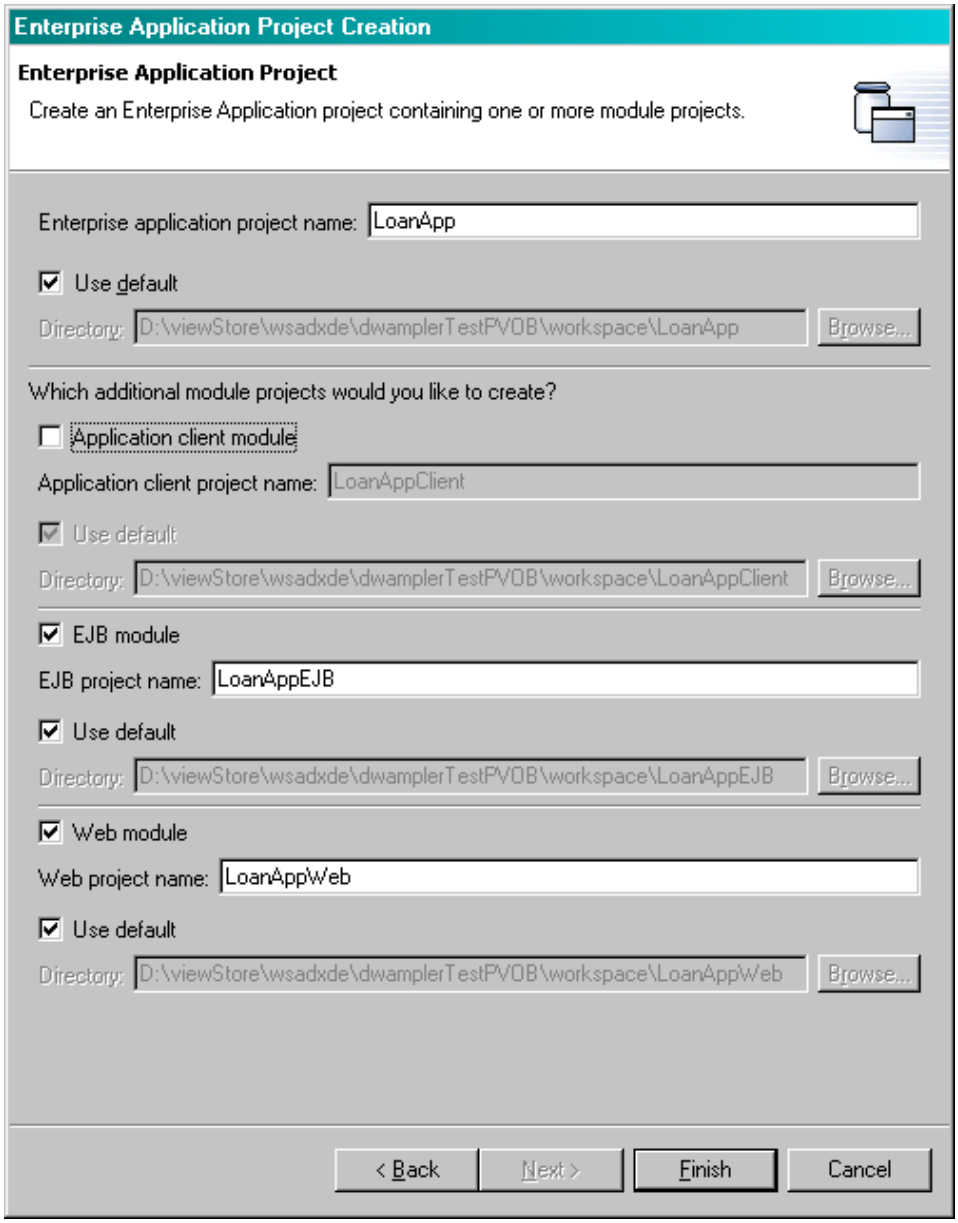
1. Invoke "ClearCase > Connect to ClearCase".

Create the "LoanApp" Project

1. Start Application Developer v5, if it is not already running, using the appropriate workspace location.
2. In the "File" menu, invoke the "New > Project" command.
3. Select "J2EE" on the left-hand side and "Enterprise Application Project" on the right-hand side. Click "Next".
4. Click "Select J2EE 1.3 Enterprise Application project".[4] Click "Next".
5. Use "LoanApp" as the project name.
6. Uncheck the "Application client module", which isn't used in this example.
7. Accept all the other defaults.
8. The wizard should look like this:

[3] When you select Enterprise Application Project, several projects are actually created. A top-level project is created that manages the Enterprise ARchive (EAR), which contains any EJB JAR, Web ARchive (WAR), and other JAR files you choose to deploy.

[4] The processes described in this whitepaper are equally valid for J2EE 1.2 projects.



9. Click “Finish”.
10. If you are using ClearCase, six dialogs will get presented after the projects are created, two for each new Project. The first dialog in each pair asks for permission to add the created files and folders to ClearCase. Accept all defaults, although you may wish to uncheck the “Keep checked out” item on each prompt, if you don’t plan to do additional work now in the projects. (I’ll assume in what follows that they are left checked out.) The second dialog in the pair asks you to pick a ClearCase “activity”. Select an appropriate activity for your project.

Three projects are created:

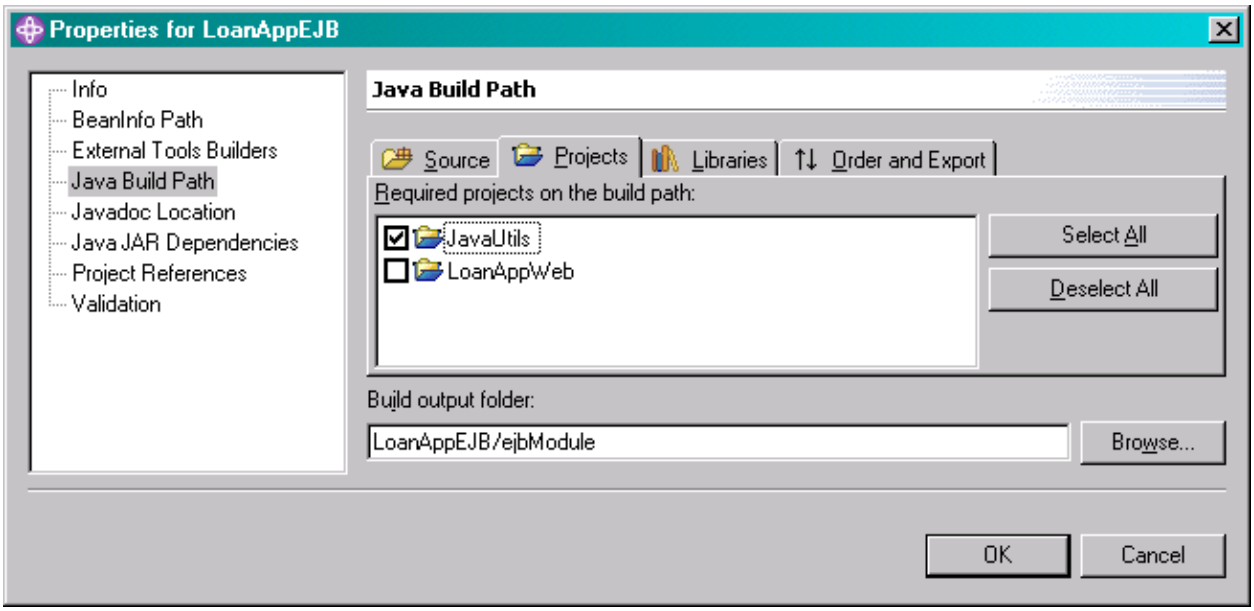
Project Name	Purpose
LoanApp	The Enterprise Application project that is the “parent” of “LoanAppEJB” and “LoanAppWeb”. It manages the EAR that will be created, including all the subordinate JAR’s that it will contain.
LoanAppEJB	The EJB-Tier, including all EJB’s and supporting Java classes and files. It manages the EJB-JAR that will be included in the “LoanApp” EAR.
LoanAppWeb	The Web-Tier, including all JSPs, Servlets, JSP tag libraries, and other supporting Java classes and files. It manages the WAR that will be included in the “LoanApp” EAR.

Create the "JavaUtils" Project

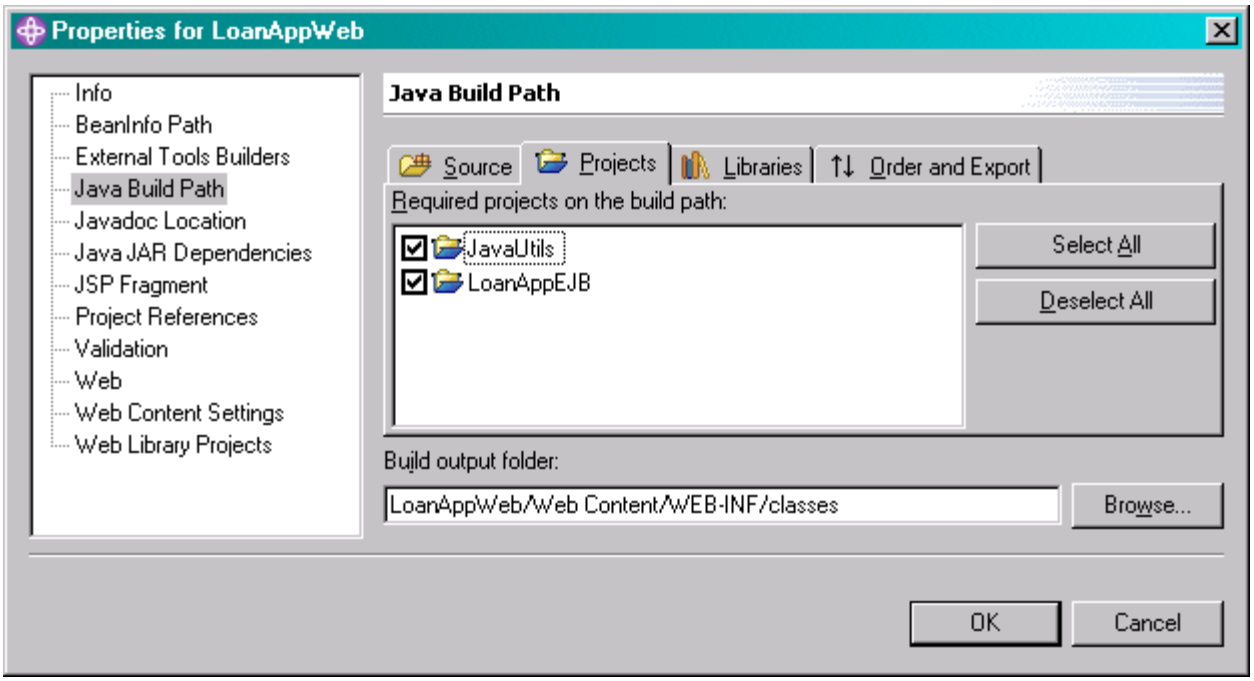
1. In the "File" menu, invoke the "New > Project" command.
2. Select "Java" on the left-hand side and "Java Project" on the right-hand side. Click "Next".
3. Use "JavaUtils" as the project name.
4. Uncheck the "Use default" item.
5. In the "Directory:" field, type "D:\viewStore\wsadxde\projects\JavaUtils".
6. Click "Finish".

Both the Web and EJB projects will use the utilities in "JavaUtils". I need to define the appropriate project dependencies for it.

1. In the Package Explorer or the Navigator, select "LoanAppEJB".
2. Right click and invoke the "Properties" command.
3. Select "Java Build Path" on the left-hand side,
4. Select the "Projects" tab.
5. Check "JavaUtils".
6. The wizard should look like this:



7. Click "OK".
8. Repeat the process for "LoanAppWeb", but before clicking "OK", also select "LoanAppEJB", so that the web-tier elements can refer to EJB's:



Configure XDE v2002

Invoke XDE Using "xde_cc.bat"

Invoke XDE using the "xde_cc.bat" script provided by the XDE 2002 service release discussed previously. Using this script to launch XDE is important because it manages many setup chores, including a number of checks to ensure that your ClearCase environment is set up properly. The workspace location it uses depends on your environment. As discussed above, the XDE workspace isn't managed under ClearCase, so the location used is not very important. Unless you need to use a specific location for other reasons, I recommend using the default value used by the script.[5]

Turn Off Automatic Builds

If you use Application Developer v5 to do your builds, as recommended[6]:

1. In the "Window" menu, invoke the "Preferences" command.
2. In the Preferences dialog, select "Workbench" on the left.
3. Uncheck the option to "Perform build automatically on resource modification"

Turn Off Automatic Code-Model Synchronization

This step not only minimizes overhead, but it makes configuration management easier. You invoke synchronization and CM commands only as needed.

1. While still in the Preferences dialog, select "Rational XDE > Code-Model Synchronization".
2. Uncheck the "Autosync" option.

[5] If you still wish to use a different directory, it may be necessary to edit the script. At the time of this writing, the script does not offer a command-line option for the workspace. In an editor, search for the place where the string "$eclipse_ws" is defined. Set the value appropriately (and carefully!!).

[6] If you use "simple" projects as described below, changing this preference isn't really necessary, because simple projects do not have any awareness of the Java build facilities. However, changing this preference is harmless and worth doing in case you add other Java-type projects later and you forget to change the preference at that time.

Use Manual Subunit Policy

Model "subunits" can be stored in separate files, so they can be managed separately under CM. This feature should not be done automatically, but manually as desired.

1. While still in the Preferences dialog, select "Rational XDE > File Storage".
2. In the "Store in Subfiles" menu, select "Manual.

Allow ClearCase Check-ins of Identical Versions (optional)

If you are using ClearCase and you do blanket check-ins of changes, you will get error dialogs when a checked-out file is unchanged. You can suppress these errors by enabling check-ins of identical versions. This does waste some space in the VOB, however. To enable this option:

1. While still in the Preferences dialog, select "Rational ClearCase".
2. Click the "Advanced Options" button.
3. Check the "Checkin even if identical" item.

Choose Consistent Naming Preferences

If you plan to create EJB's in both XDE and Application Developer, then you should consider using the same naming conventions for the elements in EJB's. The default conventions are slightly different in XDE and Application Developer.

Application Developer doesn't provide a way to specify default naming conventions. Instead, you specify the element names each time you create an EJB.

If desired, the XDE conventions can be changed as follows.[7]

1. In the "Window" menu, invoke the "Preferences" command.
2. In the Preferences dialog, select "Rational XDE > Java > EJB/Servlet Preferences" on the left.
3. Select the model to change in the "Settings for:" pop-down menu.
4. The following suffix naming conventions are used in XDE and Application Developer 5:

EJB Element	XDE Suffixes	Appl. Dev. Suffixes
Remote	*None*	*None*
Local	*None*	Local
Remote Home	Home	Home
Local Home	Home	LocalHome
Implementation	Bean	Bean
Primary Key	PrimaryKey	Key
EJB Name	EJB	*None*

Create Your Projects

In XDE, I want to create projects that correspond to the Application Developer projects so that I can model the application and develop it using XDE's Patterns engine. In this case, the "LoanAppEJB", "LoanAppWeb", and "JavaUtils" projects are mirrored in XDE. Since the parent project "LoanApp" only manages the EAR and XDE does not model EAR's, it is not required to create a mirror project for "LoanApp".

However, Enterprise Application Projects like "LoanApp" are convenient places to keep other models, such as those for use cases, analysis, data modeling, *etc*. Hence, in your application, you may wish to mirror your Enterprise Application Project anyway. Since my example does not have any of these other types of models in "LoanApp", I won't mirror it.

[7] You must wait to do these steps until after you have created the EJB project, as described below.

As discussed previously, I use the same project location for an XDE project and its corresponding Application Developer project. To do this, I must use a "simple" project for the XDE project. Then, I add the models I want in each project. So, continuing with my example, I create the following XDE mirror projects and models.

Application Developer Project:	XDE Project Type (New Project Wizard)	XDE Project Name	XDE Models
Web Project "LoanAppWeb"	Simple > Project	"LoanAppWeb"	Java Code Model, Virtual Directory Model JSP Tag Library Model
EJB Project "LoanAppEJB"	Simple > Project	"LoanAppEJB"	Java Code Model
Enterprise Application Project "LoanApp"	*None required*	*N.A.*	*N.A.*
Java Project "JavaUtils"	Simple > Project	"JavaUtils"	Java Code Model

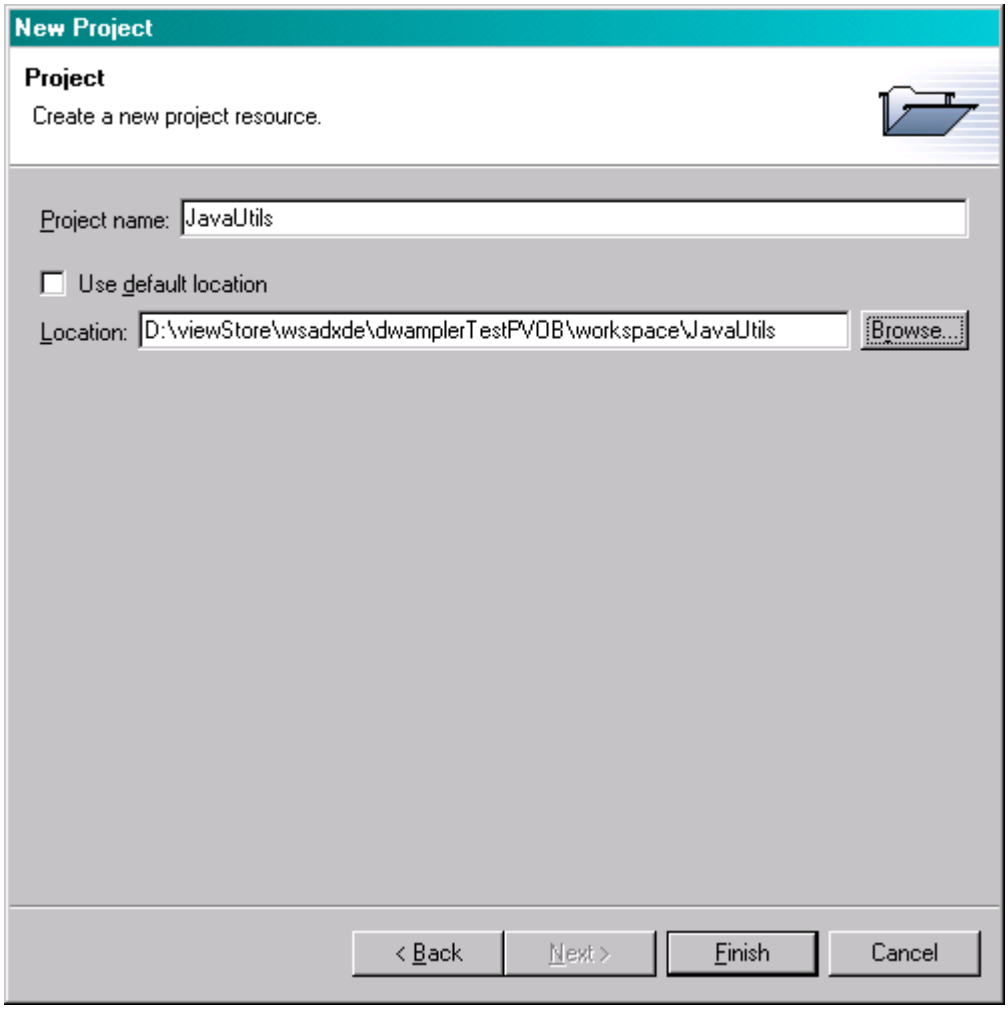
Note that the project names are the same as the corresponding Application Developer project names. This is actually not required; you can adopt a different naming convention, if you choose, and still share the same location.

Here are the steps.

> **Note:** If you are running XDE inside Application Developer v4 or Integration Edition v4.1, in the New Project wizard, you will see "EJB Modeling Project", "Web Modeling Project", and "Enterprise Application Modeling Project" types in the "Modeling" category. They are not designed for developing J2EE 1.3 applications, only J2EE 1.2 applications.
>
> If you have used XDE previously with either Application Developer v4.X IDE and you are migrating projects of these types to v5, see Appendix A for more information.

Create the "JavaUtils" Project

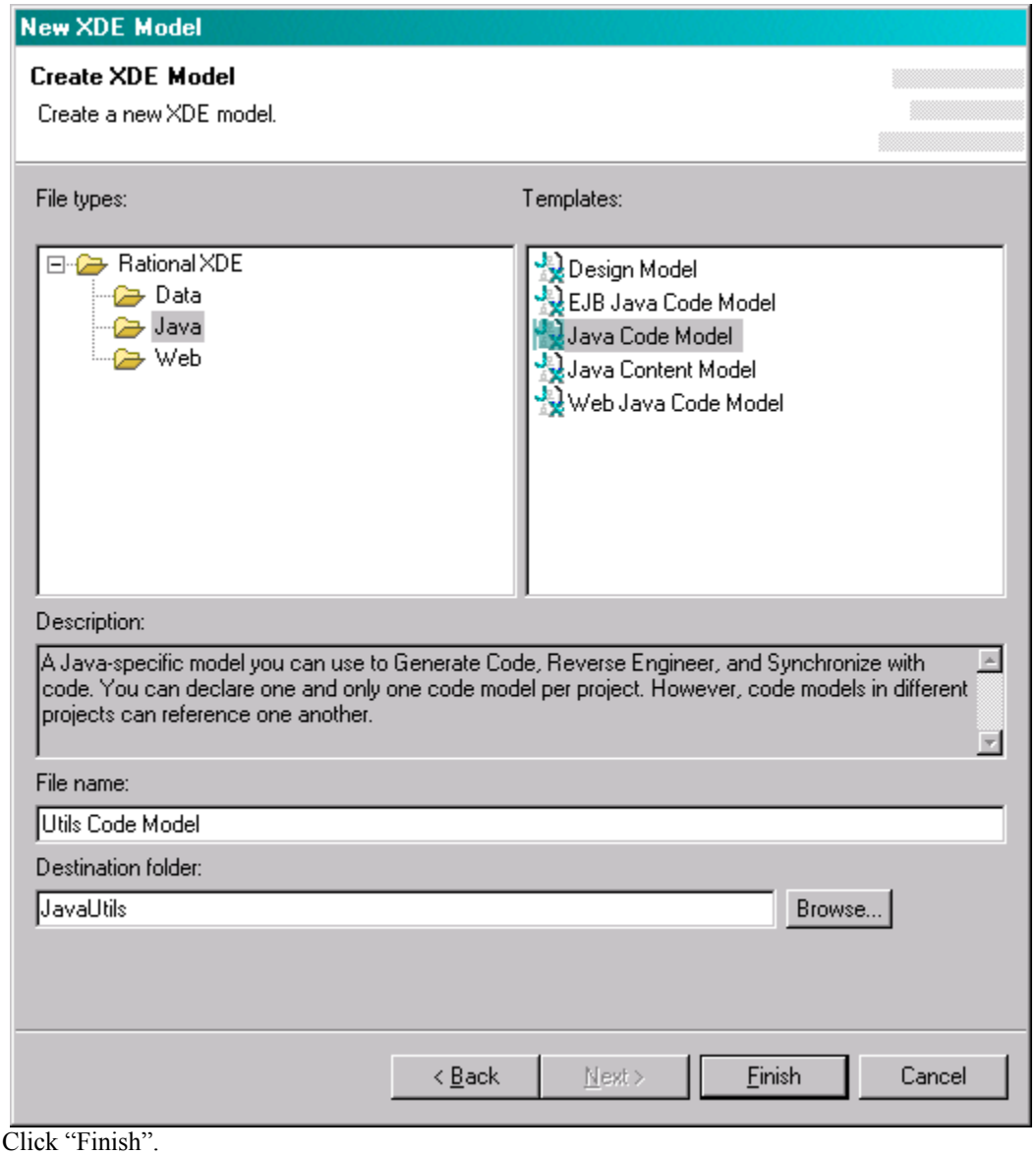
1. Start XDE, if it is not already running, using your chosen workspace location.
2. In the "File" menu, invoke the "New > Project" command.
3. Select "Simple" on the left-hand side and "Project" on the right-hand side. Click "Next".
4. Use "JavaUtils" as the project name.
5. Uncheck "Use default location".
6. For the Location, browse to "D:\viewStore\wsadxde\dwamplerTestPVOB\workspace\JavaUtils". (This is the same location as the Application Developer "JavaUtils" project. Note that the parent directory is "wsadxde", not "wsadxde_ws", which is the XDE workspace directory.).
7. The wizard should look like this:



8. Click “Finish”.

Now create a Java Code Model in the project. It can be helpful to give the models unique names, so I name it “Utils Code Model”.

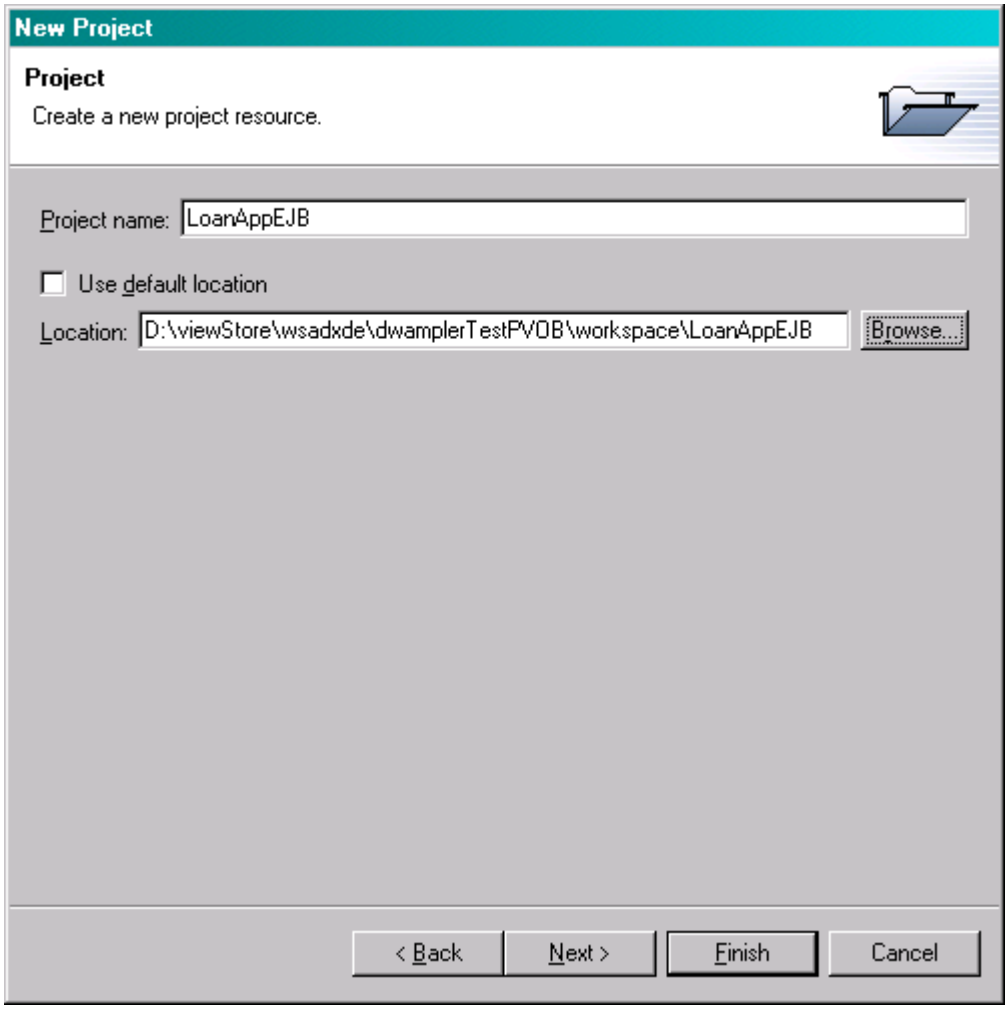
1. Invoke “File > New > Other…”.
2. In the wizard, select “Modeling” on the left-hand side and “Model” on the right-hand side. Click “Next”.
3. Select “Rational XDE > Java” on the left-hand side and “Java Code Model” on the right-hand side.
4. For “File Name:”, use “Utils Code Model”.
5. For “Destination Folder:”, use “JavaUtils”.
6. The wizard should look like this:



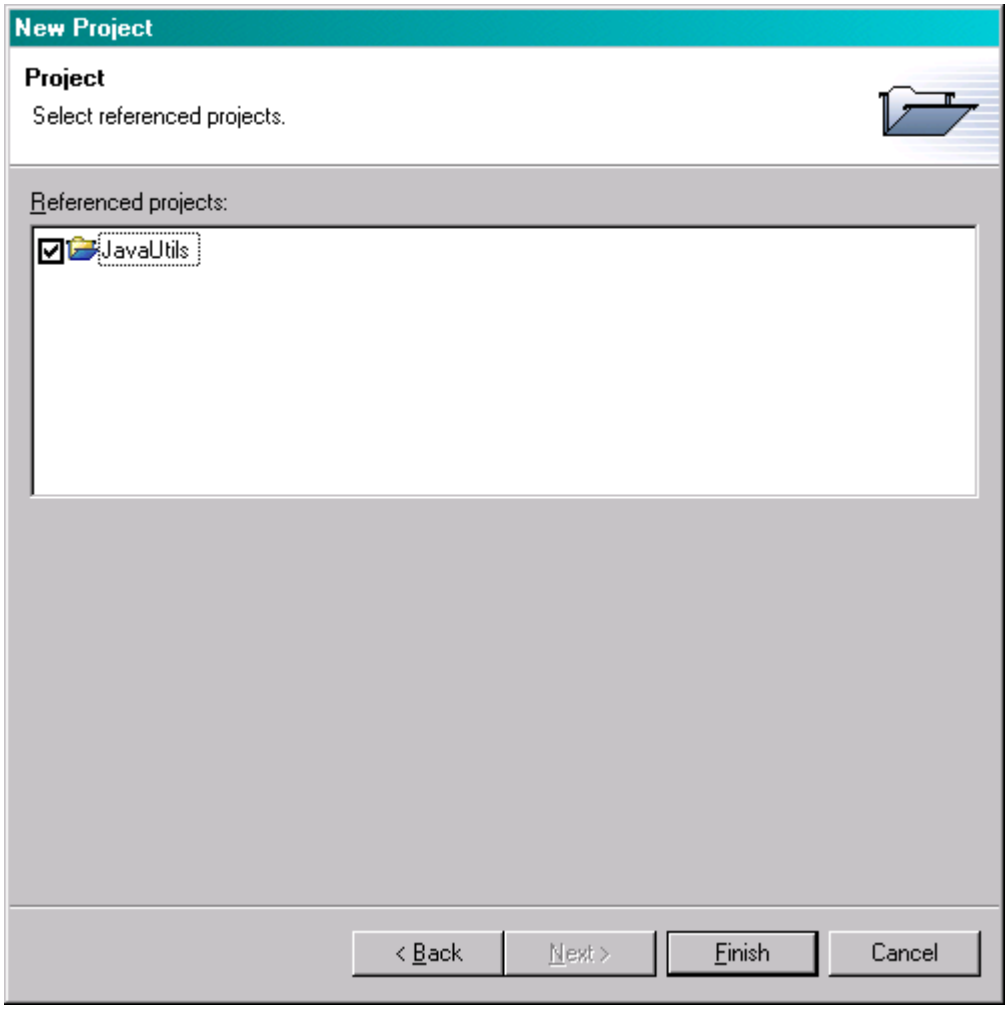
9. Click “Finish”.

Create the “LoanAppEJB” Project

1. In the “File” menu, invoke the “New > Project” command.
2. Select “Simple” on the left-hand side and “Project” on the right-hand side. Click “Next”.
3. Use “LoanAppEJB” as the project name.
4. Uncheck “Use default location”.
5. For the Location, browse to “D:\viewStore\wsadxde\dwamplerTestPVOB\workspace\LoanAppEJB”
6. The wizard should look like this:



7. Click “Next” (*not* “Finish”!).
8. Check the box “JavaUtils” in the “References Projects:” section.
9. The wizard should look like this:



10. Click “Finish”.

Now create a Java Code Model in the project. The procedure is almost the same one I just used for “JavaUtils”. I name this model “EJB Code Model”.

7. Invoke “File > New > Other…”
8. In the wizard, select “Modeling” on the left-hand side and “Model” on the right-hand side. Click “Next”.
9. Select “Rational XDE > Java” on the left-hand side and “Java Code Model” on the right-hand side. (Ignore the “EJB Java Code Model” shown.)
10. For “File Name:”, use “EJB Code Model”.
11. For “Destination Folder:”, use “LoanAppEJB”.
12. Click “Finish”.
13. In the Model Explorer, select the newly created “EJB Code Model”.
14. Right click and invoke “More Java Actions > Set Source Root”.
15. In the “Root:” field, type or browse to “D:\viewStore\wsadxde\projects\LoanAppEJB\ejbModule”. This changes the model to use the same root of the source tree that Application Developer v5 will use.

If you want to change the naming preferences used for EJB’s, as discussed previously, do it now. Don’t forget to select the “EJB Code Model” in the pop-down menu at the top.

Create the “LoanAppWeb” Project

Repeat the same steps you just followed for “LoanAppEJB”, with the following changes:

1. Name the project “LoanAppWeb”.
2. For the Location, use “D:\viewStore\wsadxde\projects\LoanAppWeb”.
3. For the Referenced Projects, use both “JavaUtils” and “LoanAppEJB”.

Now, a web modeling project can have three different types of models:

Model Type	Purpose
Java Code Model	Used for Servlets and supporting Java classes, except for JSP tag libraries.
Virtual Directory Model	Used for JSPs, HTML pages, images, *etc.*
JSP Tag Library Model	Used for modeling JSP tag libraries.

I create a Java Code Model *and* a Virtual Directory Model in the project. I don't need a JSP Tag Library Model for my example. The procedure is similar to the ones I just used for the other projects.

16. Invoke "File > New > Other…"
17. In the wizard, select "Modeling" on the left-hand side and "Model" on the right-hand side. Click "Next".
18. Select "Rational XDE > Java" on the left-hand side and "Java Code Model" on the right-hand side. (Ignore the "Web Java Code Model" shown.)
19. For "File Name:", use "Web Code Model".
20. For "Destination Folder:", use "LoanAppWeb".
21. Click "Finish".
22. In the Model Explorer, select the newly created "Web Code Model".
23. Right click and invoke "More Java Actions > Set Source Root".
24. In the "Root:" field, type or browse to "D:\viewStore\wsadxde\projects\LoanAppEJB\Java Source". This changes the model to use the same root of the source tree that Application Developer v5 will use.

The Virtual Directory Model:

25. Invoke "File > New > Other…"
26. In the wizard, select "Modeling" on the left-hand side and "Model" on the right-hand side. Click "Next".
27. Select "Rational XDE > Web" on the left-hand side and "Virtual Directory Model" on the right-hand side. (Ignore the "Web Virtual Directory Model" shown.)
28. For "File Name:", keep the default "Virtual Directory Model".
29. For "Destination Folder:", use "LoanAppWeb".
30. Click "Finish".
31. In the Model Explorer, select the newly created "Virtual Directory Model".
32. Right click and invoke "More Java Actions > Set Source Root".
33. In the "Root:" field, type or browse to "D:\viewStore\wsadxde\projects\LoanAppWeb\Web Content". This changes the model to use the same root of the source tree that Application Developer v5 will use.
34. In the Navigator, open the "LoanAppWeb" project folder.
35. Select the folder "webApplication", right click and invoke "delete". (This folder is the default used by XDE for the web content. The "Web Content" folder created by Application Developer 5 is used instead.)

Add the Projects to ClearCase

If you are using ClearCase, add the projects to source control.

1. If you have not yet connected XDE to ClearCase, invoke "ClearCase > Connect to Rational ClearCase".
2. Select any project in the Navigator, right click, and invoke "ClearCase > Refresh Status". All three project folders should be changed to show that they are managed under ClearCase and they are already checked out.
3. Select each of the four new models, right click, and invoke "ClearCase > Add to Source Control". Answer the prompts appropriately, such as the prompt for the "activity" to use.

Developing Software in XDE and Application Developer

Now that I have created the projects in both IDE's, I examine how XDE and Application Developer v5 can be used together to develop Java and J2EE applications. In this example, I create J2EE software in XDE, work with it in Application Developer v5, then do more modeling in XDE, thereby completing the "circle".

For best results, a few general principles and recommendations should be considered.

- ***Never*** modify the software in both environments simultaneously. Work in one environment, update the other one and work there. This practice avoids the need to merge changes from both environments. The steps described below don't provide for merging changes together.[8]
- When you have changed the UCM status of one or more elements in one IDE, refresh the status in the other IDE before using it again.
    - In XDE, use the context menu command "ClearCase > Refresh Status".
    - In Application Developer, use the context menu command "Team > Refresh Status".
- Validate your XDE models regularly to catch occurrences of known potential problems. (Select a model in the Model Explorer, then right click and "Validate".) This is especially important before you check in changes to UCM.
- If the memory requirements are too great when running both XDE and Application Developer v5, then try turning off XDE "Add-ins" that you are not using. Here is a list of some of the Add-ins that may be optional in your circumstances.

XDE Add-in	Purpose
Data Modeler *and* Data Modeler - Preferences	Database modeling
J2EE	EJB-tier and some aspects of web-tier modeling
RequisitePro	Interface to requirements management using Rational RequisitePro™
Web	Web-tier modeling
Web Publisher	Creation of a snapshot of models in the form of a web site

- EJB's and Servlets can be created quickly using XDE Patterns. In particular, the patterns can be used to transform platform-independent classes to EJB's, thereby giving you traceability between analysis/design and implementation constructs, for example.
- In general, using the patterns bundled in XDE, like "Gang of Four" and Sun's Core J2EE patterns (downloadable from the Rational Developer Network – http://www.rational.net) help improve architectural and code quality.
- You can create your own patterns in XDE to improve productivity by automating repetitive and error-prone tasks.
- Creating and modifying Entity BMP/CMP fields and EJB methods are usually faster in XDE.
- Application Developer has better facilities for developing the Web tier.
- Assembling the application, including the definitions of the deployment descriptor elements (*e.g.,* Security Roles, Transactions, *etc*.) and the properties specific to WebSphere Application Server deployment are easier to manage in Application Developer v5.
- Editing code, deployment to test or production application servers, and testing are best done in Application Developer v5.

Develop Software in XDE

I begin the example usage scenario by creating software in XDE.

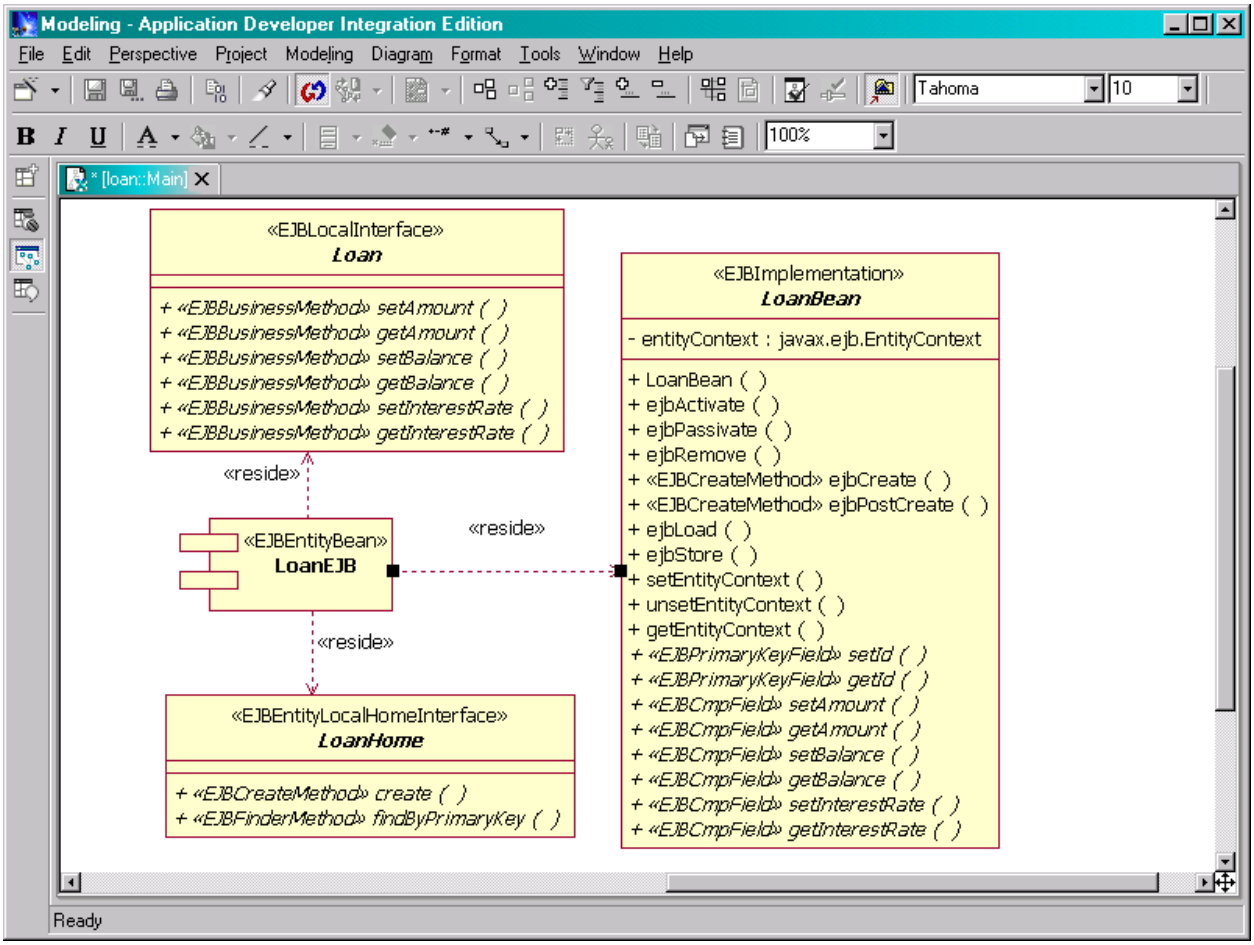
1. Start XDE if necessary.
2. If you have software changes to import into XDE, proceed with the steps in the section Update XDE with Changes, then return here.

In this example, I create two EJB's, an HTML page, a JSP page, a Servlet, and several "utility" classes. For brevity, the steps are not repeated in detail here, but the items are described in the following table. Consult the online help for more information about creating these items in XDE.

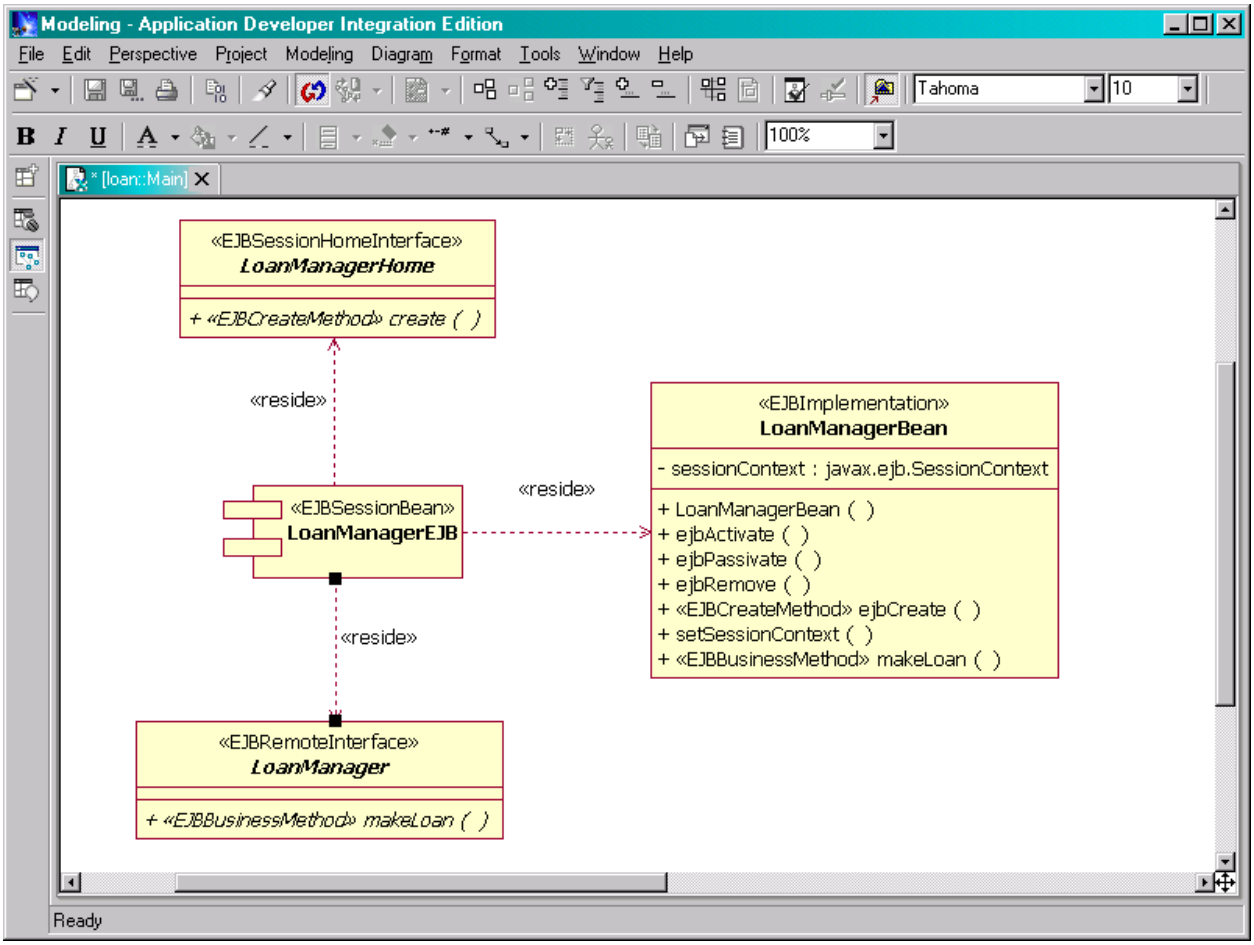
[8] Should you find it necessary to merge changes made in both environments, the best way is to start on the side with the *most* changes (and hence requiring fewer things to merge). Cut and paste source code and deployment descriptor elements until the changes are merged. Then update the project "metadata". When working in XDE, use "Reverse Engineer" and "Import EJB/Web Deployment Descriptor". When working in Application Developer, use "Refresh from Local".

<table>
<tr><td colspan="2">(LoanAppEJB) EJB Code Model</td></tr>
<tr><td>com.megabank.loan.LoanEJB</td><td>A CMP 2.0 EJB that represents a “loan”, with local interfaces. It has 3 CMP fields and one primary key field:
<table>
<tr><td>float amount</td><td>Original loan amount</td></tr>
<tr><td>float balance</td><td>Remaining balance</td></tr>
<tr><td>float interestRate</td><td>Interest rate: e.g., 3% = .03</td></tr>
<tr><td>java.lang.Integer id</td><td>Primary key field</td></tr>
</table>
The get/set methods for the CMP fields (but not the PK field) are exposed on the local interface.</td></tr>
<tr><td>com.megabank.loan.LoanManagerEJB</td><td>A Stateless Session EJB for managing loans, with remote interfaces. It has one “business” method on the remote interface:
<code>int makeLoan (float salary, float amount, float interestRate, int termMonths);</code>
The returned “int” is the id of the new Loan (or zero on failure)</td></tr>
<tr><td>com.megabank.loan.ServiceLocator</td><td>Adapted from the Sun Core J2EE Patterns, which are available as a reusable asset for XDE on the Rational Developer Network (http://www.rational.net), ServiceLocator provides a convenient interface for EJB clients. It hides most of the EJB creation details. It is used by “Controller” (below).</td></tr>
<tr><td colspan="2">(LoanAppWeb) Virtual Directory Model</td></tr>
<tr><td>loan/index.html</td><td>The “home” page for the web interface. It has a form for specifying the applicant’s annual salary, the loan’s amount, rate, and term (in years) for a loan. The form is submitted to the servlet “Controller” which creates a new loan.</td></tr>
<tr><td>loan/result.jsp</td><td>Displays the result of the loan-creation process. The Controller servlet forwards the results to this page for displaying to the user.</td></tr>
<tr><td colspan="2">(LoanAppWeb) Web Code Model</td></tr>
<tr><td>com.megabank.loan.servlet.Controller</td><td>Processes the form submission from “index.html” and forwards the results to “result.jsp”.</td></tr>
<tr><td colspan="2">(JavaUtils) Utils Code Model</td></tr>
<tr><td>com.megabank.utils.USDollar</td><td>Does correct calculations in US Dollars while avoiding potential rounding errors when using “float” values for $$$$.</td></tr>
</table>

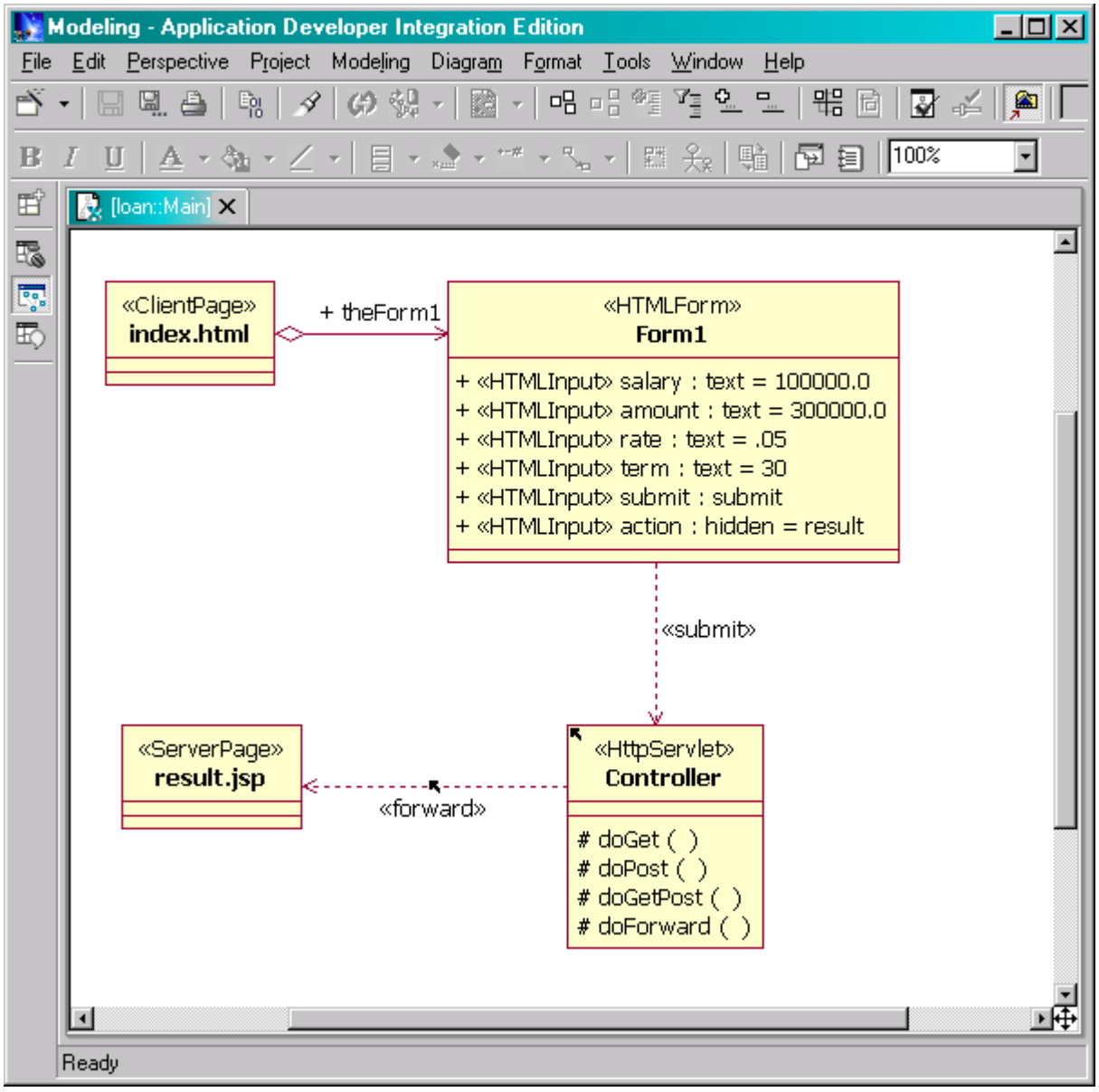
The “LoanEJB” looks like this:



The “LoanManagerEJB” looks like this:



The Web tier looks like this:



Save Changes in XDE

A few steps are required to save *all* changes made in XDE, so that they are visible in Application Developer v5. The models and source files must be synchronized, as needed, and saved. Also, the Web and EJB deployment descriptors need to be exported from XDE to appropriate locations.

An Important Note about Deployment Descriptors

At this point it is important to understand the role of the deployment descriptors in the process of synchronizing XDE and Application Developer v5, as well as the limitations of descriptor handling in XDE. Both tools use the descriptors to associate "metadata" with the Java source code they are importing. The descriptors indicate which sources correspond to EJB's, Servlets, *etc*., along with the properties of these objects. Also, the descriptors are used to update each tool with the latest changes in the other tool.

There are a few limitations in XDE's handling of descriptors that you should keep in mind.

1. In a few rare scenarios, XDE can lose security role information that was defined in Application Developer. The information is deleted by XDE during a round trip from Application Developer to XDE and back again. The full details are documented in the XDE release notes.

2. When XDE imports and exports web deployment descriptors (web.xml), it only handles information for Servlets in the Java Code Model (the one I named "Web Code Model"). JSP information and relationships between the Servlets, JSP's, and HTML pages (*e.g.,* forwards, includes, and submits) are not currently handled. You may decide that it is better not to export the web.xml from XDE, but to make the appropriate changes manually in Application Developer.[9]

Therefore, use caution when working with deployment descriptors. As a precaution, make a back-up copy of each descriptor before importing a version of it from the other tool, until you determine the approach that works best for you. Compare the updated and saved copies and make sure that the current working copy is complete and correct.

Save All Changes in XDE

Here are the steps required to save information from XDE in preparation for importing into Application Developer.

1. Select the Virtual Directory Model in the Model Explorer, right click, and invoke "Generate Code".
2. If you have automatic code-model synchronization ("autosync") turned off, generate code for the other models.
    a. Select each Model (except the Virtual Directory Model) in the Model Explorer.
    b. Right click and invoke "Generate Code".
3. Use "File > Save All" to save all model and project changes.
4. Export the EJB deployment descriptor (Read the notes above!):
    a. Select the EJB Code Model in the Model Explorer.
    b. Right click and invoke "More Java Actions > Export EJB Deployment Descriptor".
    c. In the prompt, browse to the "…/LoanEJBModel/ejbModule/META-INF" directory.
    d. Click "OK".
5. Export the Web deployment descriptor (Read the notes above!):
    a. Select the Web Code Model or the Virtual Directory Model in the Model Explorer.
    b. Right click and invoke "More Java Actions > Export Web Deployment Descriptor".
    c. In the prompt, browse to the "…/LoanWebModel/Web Content/WEB-INF" directory.
    d. Click "OK".

Check-in Model Changes to UCM

Now is a good time to check in your model changes. Recall that I recommend managing all other files from within Application Developer.

1. In XDE, select the four models in the Navigator.
2. Right click and invoke "ClearCase > Check in".
3. Check "Keep checked out" if you wish.
4. Add a comment if you wish.
5. Click "OK".
6. If you get prompts stating that the model files have changed on disk and asking if you want to reload them into XDE, answer "OK".

Updating Application Developer v5

I must use the "Refresh" command to tell Application Developer to look for the new and updated files in the projects. To avoid annoying compilation errors from missing dependencies, I import the projects in the order of those dependencies. In my example, I import them in this order:

- JavaUtils
- LoanAppEJB (needs JavaUtils)
- LoanAppWeb (needs JavaUtils and LoanAppEJB)

[9] One way to do this is to export the file from XDE to a temporary location, edit that file, then cut and paste changes into the descriptor editor in Application Developer.

Update JavaUtils Information

1. Make sure you followed the instructions in the previous section for saving all changes in XDE.
2. In Application Developer v5, select the "JavaUtils" top-level folder in the Navigator or Package Explorer.
3. Right click and invoke "Refresh".
4. If you are using ClearCase, you will be asked whether or not you want to add all the new files to ClearCase. Answer "OK". You will then need to select an "activity".
5. Check the task list for any errors (*e.g.,* compiler errors) and correct them as needed.

Import LoanAppEJB Information

1. In Application Developer v5, select the "LoanAppEJB" top-level folder in the Navigator or Package Explorer.
2. Right click and invoke "Refresh".
3. If you are using ClearCase, you will be asked whether or not you want to add all the new files to ClearCase. Answer "OK". You will then need to select an "activity".
4. Check the task list for any errors (*e.g.,* compiler errors) and correct them as needed.

Note that the deployment descriptor "ejb-jar.xml" is also processed during these steps.

Import LoanAppWeb Information

1. In Application Developer v5, select the "LoanAppWeb" top-level folder in the Navigator or Package Explorer.
2. Right click and invoke "Refresh".
3. If you are using ClearCase, you will be asked whether or not you want to add all the new files to ClearCase. Answer "OK". You will then need to select an "activity".
4. Check the task list for any errors (*e.g.,* compiler errors) and correct them as needed.

Note that the deployment descriptor "web.xml" is also processed during these steps.

Check-in Changes to UCM (Optional)

Now is a good time to save the changes to the source and descriptor files. If you are using ClearCase, perform the following steps.

1. In Application Developer, in the Navigator, multi-select elements that you want to check in.
2. Right click and invoke "Team > Check in".[10]
3. Check "Keep checked out" if you wish.
4. Add a comment if you wish.
5. Click "OK".
6. If you get prompts stating that the files have changed on disk and asking if you want to reload them into Application Developer, answer "OK". (This should only happen if the file is open in an editor.)

Develop Software in Application Developer v5

Once Application Developer is updated with the changes to all the projects, work can begin in Application Developer.

For my example, I add a second CMP 2.0 Entity bean, a CMR relationship between the two Entity beans, a second servlet and another HTML page. As before, I summarize the changes here. Add the newly created files to UCM, if prompted to do so.

[10] Unfortunately, this menu option is disabled for some multiple-item selections; selecting all files or all folders seems to help. The buttons in the ClearCase toolbar seem to be less restrictive.

<table>
<tr><td colspan="2">(LoanAppEJB) EJB Code Model</td></tr>
<tr><td>com.megabank.loan.CustomerEJB</td><td>A CMP 2.0 EJB that represents a “customer” for a loan, with local interfaces. It has 2 CMP fields, one of which is the primary key field:
<table>
<tr><td>String name (PK field)</td><td>Customer’s name</td></tr>
<tr><td>float salary</td><td>Annual salary</td></tr>
</table>
The get/set methods for the CMP fields are exposed on the local interface.</td></tr>
<tr><td colspan="2">(LoanAppWeb) Virtual Directory Model</td></tr>
<tr><td>loan/login.html</td><td>The “login” page for the web interface. It has a form for specifying the user name and password. The form is submitted to the servlet “Authenticator” which authenticates the credentials.</td></tr>
<tr><td colspan="2">(LoanAppWeb) Web Code Model</td></tr>
<tr><td>com.megabank.loan.servlet.Authenticator</td><td>Processes the form submission from “login.html” and forwards the results to “index.html”.</td></tr>
</table>

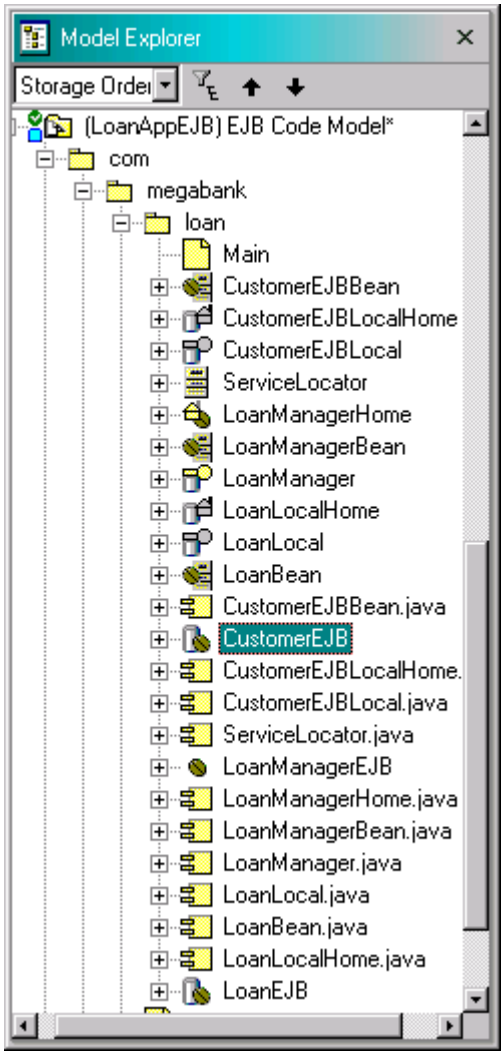
Save Changes in Application Developer

Save all modifications in Application Developer when you are ready to work in XDE again. For example, invoke “File > Save All”. Saving your changes in UCM is also a good idea at this point.

Update XDE with Changes

The process of updating XDE with the Application Developer changes is very similar to the reverse process described above.

1. If using ClearCase:
    a. Multi-select all elements in the Navigator. (*e.g.,* select the first element, then use shift + click on the last element.)
    b. Right click and invoke “ClearCase > Refresh Status…”
2. Multi-select the top-level folders for “JavaUtils”, “LoanAppEJB”, and “LoanAppWeb” in the Navigator.
3. Right click and invoke “Refresh from Local”.
4. You may get asked to add new directories or files to ClearCase. These may be compiler output directories that don’t need to be managed. Look at the paths and decide whether or not you want to add the elements to ClearCase.
5. Switch to the Model Explorer.
6. For each project where you added new files, all but “JavaUtils” in this case:
    a. Select the model in the Model Explorer.
    b. Right click and invoke “More Java Actions > Add/Remove Modeled Files…”
    c. Click the “Add Recursively” button.
    d. Click “OK”.
7. Import the EJB and Web deployment descriptors into the corresponding XDE models. In my example:
    a. For “LoanAppEJB”, select the model in the Model Explorer, right click and invoke “More Java Actions > Import EJB Deployment Descriptor.” Browse to the descriptor in “…/LoanAppEJB/ejbModule/META-INF/ejb-jar.xml”.
        i. Browse to the “com.megabank.loan” package and make sure the new “CustomerEJB” class and interfaces have the same kinds of icons that the older beans have. Something like this:

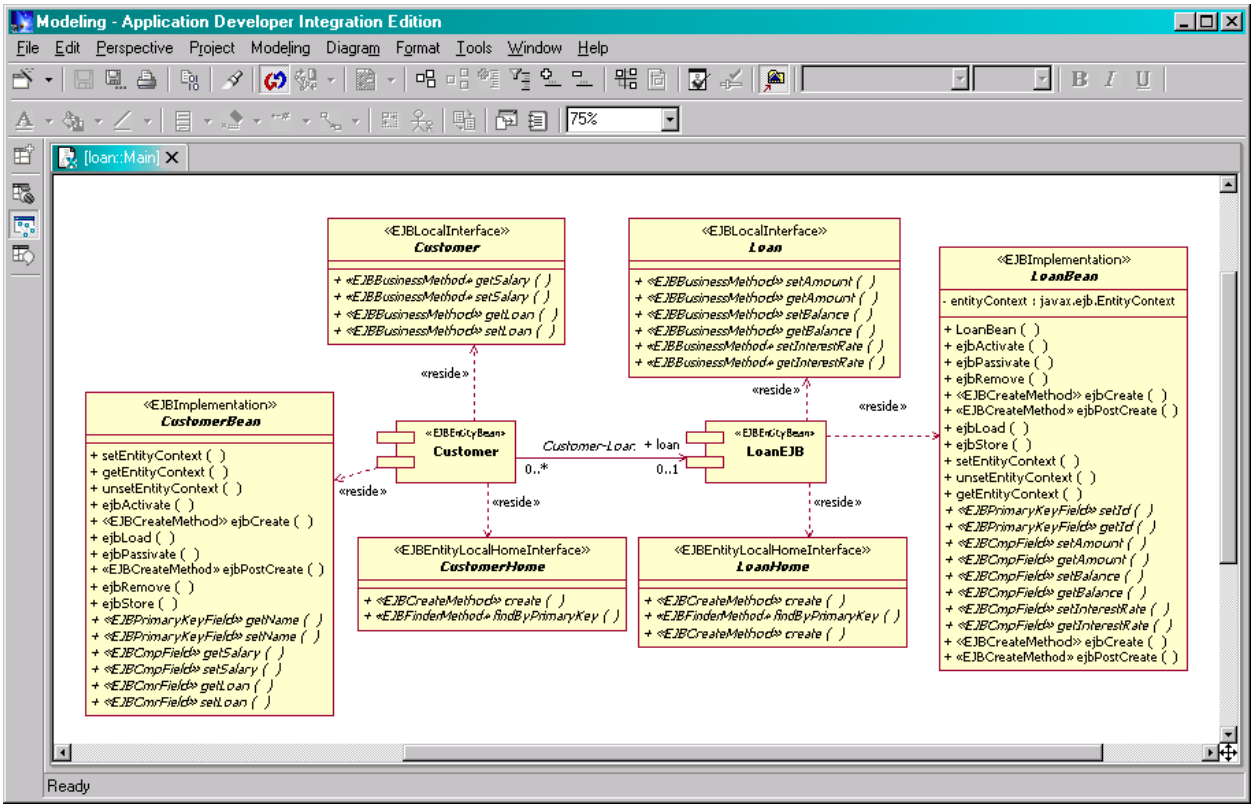


b. For "LoanAppWeb", select the "Web Code Model" in the Model Explorer, right click and invoke "More Java Actions > Import Web Deployment Descriptor." Browse to the descriptor in "…/LoanAppWeb/Web Content/WEB-INF/web.xml"[11]. The "Authenticator" servlet should have the same "http" icon that the original "Controller" servlet has:

8. In the "Virtual Directory Model", invoke the "Reverse Engineer" command to update the model with imported source code changes.

After importing these models and updating diagrams, I have the following for "LoanAppEJB", where the new "CustomerEJB" is shown with the older "LoanEJB", for example:

[11] If you get errors written to the output window about "null" values and there are "<servlet>" tags in the "web.xml" for JSP pages, try temporarily commenting out those tags. This should allow the Code Model to get properly updated. Note that the string "<!-- … -->" is a comment in XML (and HTML). Anything can appear between the delimiters.



Editing of the models and the source code can now begin in XDE. The processes described above can be repeated when needed to keep the two tools synchronized.

Conclusions

This whitepaper describes the steps for using Rational XDE v2002 and Application Developer v5 together. The tools share project locations (but not workspace locations). CM tasks are handled in Application Developer v5, except for managing the XDE models, where XDE itself is used.

The bulk of the effort described here occurs during the setup process, after which relatively few steps are required to keep the two IDE's synchronized. You work in whichever IDE is best for a particular task and then switch to the other IDE, as desired.

Appendix A – Using Existing XDE Modeling Projects

If you are using XDE inside Application Developer v4 or Application Developer Integration Edition v4.1, in the "Modeling" category of the New Project wizard, there are "EJB Modeling Project", "Web Modeling Project", and "Enterprise Application Modeling Project" projects. These projects correspond to Application Developer's "EJB Project", "Web Project", and "Enterprise Application Project" projects, respectively, which are provided by both Application Developer v4 and v5. In fact, the "Modeling" projects *are* the Application Developer projects, with the addition of appropriate models.

However, in XDE, these projects are designed to support J2EE 1.2, since that is the latest standard supported by Application Developer v4. Despite this, if you are migrating existing modeling projects to J2EE 1.3, you can continue to use these projects, with the following modifications.

Clone the Workspace and Projects

You will still need separate workspaces for XDE and Application Developer v5.

1. Pick a new workspace location for Application Developer v5, as discussed previously.
2. Invoke Application Developer v5 using that workspace.

3. Import your existing projects into Application Developer v5, using one of the "File > Import" options. One possibility is to create new projects and import the source files, zip files, *etc*. Note that this is a good opportunity to do any major project *refactoring* that may be needed.

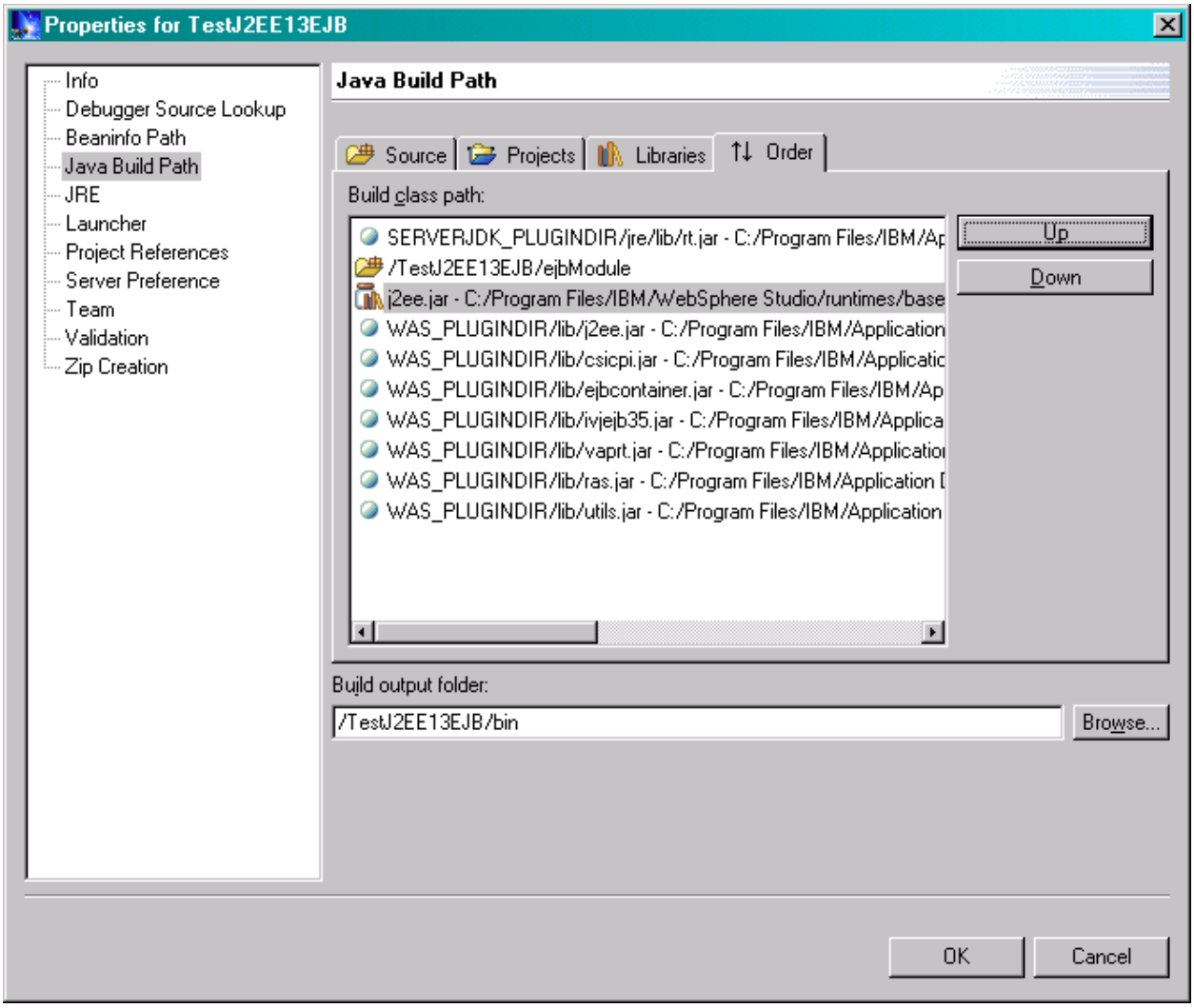
Set the XDE J2EE Preference to J2EE 1.3

1. Open all the models that need converting.
2. Invoke the "Window > Preferences" command.
3. In the Preferences dialog, select "Rational XDE > Java > EJB/Servlet Preferences".
4. Select the first model that needs changing in the "Settings for:" menu.
5. Select the "1.3 (EJB 2.0, Servlet 2.3)" option in the "Default J2EE Version" menu.
6. Repeat the previous two steps for *each* model that needs changing.
7. Click "OK".

Unfortunately, this J2EE preference must be reset every time one of these models is opened for the first time in a new XDE session. XDE assumes that these projects are only to be used for J2EE 1.2 development, even though XDE itself supports J2EE 1.3, so XDE corrects the "error".

Add the J2EE 1.3 Jar to the Build Path

1. Select each project in the Navigator.
2. Right click and invoke the "Properties" command.
3. Select the "Java Build Path" on the left.
4. Click the "Libraries" tab on the right.
5. Click the "Add External Jars" button.
6. Browse to the "j2ee.jar" that comes with Application Developer's built-in version of WebSphere Application Server v5. If you installed Application Developer v5 in the default location, it is here:
   C:\Program Files\IBM\WebSphere Studio\runtimes\base_v5\lib\j2ee.jar
7. Click the "Order" tab.
8. Select the "j2ee.jar" you just added.
9. Click the "Up" button until "j2ee.jar" is positioned above all the other J2EE-related jars. (If in doubt, it is okay to send it all the way to the top.) The dialog should look something like this:



10. Click “OK”.

Fortunately, this step does *not* have to be repeated.

Appendix B – Using Separate Project Locations

This whitepaper recommends sharing project locations between XDE and Application Developer v5. However, if you decide to keep those locations separate, you can still use the import feature in each IDE to keep the corresponding projects synchronized. This is only necessary if you work with source code and deployment descriptors in XDE.

Since both IDE’s use the same Eclipse framework, you use the same “File > Import” command to import updated sources from the *other* IDE. For example, suppose you have updated your models and source code in XDE and you want to update the corresponding Application Developer v5 projects with the changes. In Application Developer v5, invoke the “File > Import” command for each project and then Select the “File System” option.

There are small differences in the project structures, mostly changed default directory names, between XDE v2002 and Application Developer v5. This is especially true if you used XDE with Application Developer v4.X. When selecting what files to import, you will need to select the source and destination directories appropriately so that the files end up in the correct locations. Do some experimentation to figure out what is right for your projects.

When selecting what to import, choose only the source files, Java property files, and deployment descriptors.

Finally, when importing J2EE components into XDE, you must also import the deployment descriptors separately into the models. (This was discussed previously.) In the Model Explorer, select each EJB and Web Java Code

Model, then invoke “More Java Actions > Import … Deployment Descriptor”. Choose the same descriptor you just imported as a file, “ejb-jar.xml” for EJB projects and “web.xml” for Web projects.

Rational Software
Dual Headquarters:
Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Toll-free: (800) 728-1212
E-mail: info@rational.com
Web: www.rational.com
International Locations: www.rational.com/worldwide