BEA WebLogic Integration Kit for IBM VisualAge for Java

User Guide

BEA WebLogic Integration Kit for IBM VisualAge for Java,
Version 3.5
Document Edition 1.1
February 2001

Copyright

Restricted Rights Legend

Trademarks or Service Marks

BEA WebLogic Integration Kit for IBM VisualAge for Java User Guide

Document Edition	Part Number	Date	Software Version
1.1	N/A	February 2001	3.5

Contents

CHAPTER

1 User Guide

This guide discusses the following topics:

- Overview of The Integration Kit
- Example: Compiling the Examples
- Example: Setting Server Properties
- Example: Using Command-Line Arguments
- Example: Cloudscape Database

Overview of The Integration Kit

IBM VisualAge for Java, Version 3.5 (Professional or Enterprise Edition), is an integrated, visual environment that supports the complete cycle of Java program development.

BEA WebLogic Server is an award-winning Java application server for developing, deploying, and managing Web applications. It simplifies the development of portable and scalable applications, and it provides interoperability with other applications and systems. BEA WebLogic Server also offers the most complete implementation of the Java 2 Enterprise Edition standard.

BEA WebLogic Integration Kit for IBM VisualAge for Java Version 3.5 (The Integration Kit) is a Java application that helps you develop and debug your WebLogic Server applications from within IBM VisualAge for Java. All activity in IBM VisualAge for Java is organized around a workspace that contains the Java programs that you are developing. The workspace also contains all the packages, classes, and interfaces that are found in the standard Java class libraries, and other libraries that

your classes may need. The Integration Kit adds the classes required to run WebLogic Server to your workspace. As you develop and debug your applications you will add classes and projects to your workspace.

This document begins by discussing the examples included in the installation. It then describes the most common situations in which the tools provided by The Integration Kit are used.

Examples

If you have installed the examples that are in the installation, The Integration Kit install creates a VisualAge project, `WebLogic Examples`, which contains example code illustrating how to use many of the capabilities of WebLogic Server. In addition to the Java code in the VisualAge project, some of the examples also require configuring server settings in order to run properly, and some offer command-line arguments which you can modify.

For instructions on how to run each example, see the BEA WebLogic Server Examples documentation. These documents describe how to build and run the examples from the command line. Most of the instructions also apply to running the examples within VisualAge, but there are a few differences to keep in mind:

Example: Compiling the Examples

Because code is automatically compiled in the IBM VisualAge for Java workspace, you can ignore the instructions for compiling the examples that are provided in BEA WebLogic Server documentation.

Example: Setting Server Properties

For most examples, you are required to configure various properties in the `weblogic.properties` file. The server will read its properties from the `weblogic.properties` file in the *`WebLogic`* home directory (that is, the directory in which WebLogic Server is installed). If the server is running in IBM VisualAge for Java when you change properties, you must stop the server and restart it after setting the example properties.

Example: Using Command-Line Arguments

1. Select an executable class for the example.
2. Choose Select → Properties from the menu in the workbench window.
3. Choose the Program tab in the properties window, and enter arguments for it in the Command Line Arguments text field.
4. Press OK.

Example: Cloudscape Database

When running IBM VisualAge for Java Version 3.5 with BEA WebLogic Server Version 5.1 and Cloudscape, keep in mind the following limitation. Cloudscape database usage is unsupported because of problems that occur when Cloudscape is run within the VisualAge environment. This issue is currently being investigated by IBM Support (PMR 15142,519,000).

Most of the examples shipped with BEA WebLogic Server use the `demoPool` database connection pool, which is configured to use a preconfigured Cloudscape database that is installed with WebLogic Server. In order to run these examples in IBM VisualAge

for Java, you must create the example tables in another database, such as Oracle, and change the `demoPool` configuration in `weblogic.properties` such that this second database is used.

WebLogic Server provides a Data-Definition Language (DDL) file for the examples database, and a Schema tool that you can use to create a database from a DDL file. For additional instructions, see the `examples.utils` documentation for BEA WebLogic Server.

CHAPTER

2 Usage Scenarios

This section describes the most common scenarios in which The Integration Kit tools are used.

- Starting and Stopping WebLogic Server from Inside IBM VisualAge for Java
- Generating and Deploying WebLogic Deployable EJB Jars
- Creating Java Client Applications
- Debugging Server-Side and Client-Side Code
- Developing Servlet Applications
- Developing Applet Applications
- Installing WebLogic Service Packs
- Uninstalling Service Packs
- Using the Export Utility
- Using the Import Utility
- Known Issues with the Export and Import Utility
- Known Issues with the Integration Kit

Starting and Stopping WebLogic Server from Inside IBM VisualAge for Java

Starting the Server

1. Select and double-click the WebLogic Server project in the All Projects pane of the Projects tab. A separate window, containing only the WebLogic Server package, is displayed on the screen.
2. In the WebLogic Server window, click the `Run` button, or right-click the `WebLogic Server`. Select `Run` → `Main`.

Stopping the Server

While the server is running, a dialog box is displayed. You can use this dialog box to shut down the server.

Generating and Deploying WebLogic Deployable EJB Jars

1. Create a new IBM VisualAge for Java project. Then either develop or import the EJB's into the project.
2. Import any XML deployment descriptors required for the EJB into the package or project using the IBM VisualAge for Java Import Utility. If you are generating a single jar file with multiple beans, then the deployment descriptors must exist at the project level. Verify that the XML files are directly under the Project name by viewing the resources tab window of the Visual Age for Java Workbench.

3. If you are generating an EJB by selecting a package, the xml files must be at the package level. The XML file must exist in a folder structure similar to that of the package name. The folder structure is placed under the Project name in the resources tab window of the Visual Age for Java Workbench.

4. Select the project or an EJB package inside the project. Then select Selected → Tools → WebLogic Server Tools → Generate EJB jar.

   Note: To use the `Generate EJB jar` tool, users of the Enterprise Edition of IBM VisualAge for Java must first create an open edition of the EJB package. To do this, select the package, and then select Selected → Manage → Create Open Edition.

   Note: If you are building this package for the first time, you are prompted for the name of an output jar file. For more information, see the *Tutorial*.

5. If you need to change this name later, use the `Configure EJB package` tool as follows:

   a. Select the project or an EJB package inside the project.

   b. Select Tools → WebLogic Server Tools → Configure EJB package.

6. Add the generated jar file to the deploy statement in the `weblogic.properties` file. For example, to deploy an EJB called `ejb_basic_statelesssession.jar` in the `weblogic.properties` file, add the following line to the `weblogic.properties` file:

   ```
   weblogic.ejb.deploy=C:/weblogic/myserver/ejb_basic_statelessSession.jar
   ```

   Here `c:/weblogic` refers to the directory in which BEA WebLogic Server is installed

7. Make any other necessary configuration changes in the `weblogic.properties` file. For example, to successfully deploy an EJB that makes use of a database, you must add statements to the `weblogic.properties` file. For example, if the EJB uses an Oracle database, then the lines shown in the following listing must be added to the `weblogic.properties` file.

Listing 2-1 Declaring Usage of Oracle Database in `weblogic.properties` File

```
weblogic.jdbc.connectionPool.demoPool=\
url=jdbc:weblogic:oracle,\
```

```
driver=weblogic.jdbc.oci.Driver,\
loginDelaySecs=1,\
initialCapacity=4,\
maxCapacity=10,\
capacityIncrement=2,\
allowShrinking=true,\
shrinkPeriodMins=15,\
refreshMinutes=10,\
testTable=dual,\
props=user=SCOTT;password=tiger;server=demo

weblogic.jdbc.TXDataSource.weblogic.jdbc.jts.demoPool=demoPool

weblogic.allow.reserve.weblogic.jdbc.connectionPool.demoPool=ever
yone
```

Note: For instructions on how to declare a database, see the BEA WebLogic Server documentation.

8. Start WebLogic Server from inside IBM VisualAge for Java IDE.

Considerations

- If you configure server properties in the `weblogic.properties` file while the server is running in IBM VisualAge for Java you must stop the server and restart it.
- To successfully deploy and test an EJB (which generally communicates with a database) you must complete some setup tasks for the database driver and tables, and the pool and DataSource for your EJB. For example, when using Oracle, and the `demoPool` used in the examples, you must complete the following procedure:

  a. Create your database and tables, and populate the tables with data (For instructions, see *Oracle Databases* and the BEA WebLogic Server documentation about JDBC drivers.)

  b. Add the code, as shown in Listing 2-1, to the `weblogic.properties` file.

  c. Append the following address to the server's class path: *`WebLogic`*`\classes`. In this context, *`WebLogic`* refers to the directory in which BEA WebLogic Server is installed.

Creating Java Client Applications

Before you can create a Java client application, you must deploy an EJB jar in BEA WebLogic Server. Then complete the following procedure:

1. Create a new IBM VisualAge for Java project (or use the EJB project) and either develop or import into the project the Java code required by your EJB client.
2. Set the project's classpath.

   Note: In VisualAge for Java, you can set the classpath at the workspace level or at the class level. If you specify a workspace-level classpath, every class in the workspace will reference these classes when it is compiled or run. This classpath is also needed to find resource files used by your classes. During the installation process for The Integration Kit, you added the `WebLogic Server Classes`, `WebLogic Support Libraries`, and `WebLogic Java Enterprise Libraries` to the workspace-level classpath because most of these classes are required by the examples delivered with The Integration Kit.
3. Select Window → Options.
   a. In the Options dialog, select `Resources`.
   b. Click `Edit` next to the Workspace classpath field.

Note: In addition, you may also set classpath at the class level. These class-level classes are prepended to the workspace-level classpath for the given class. To set the classpath for a class:

   a. Right-click the class and select `Properties` from the class's pop-up menu.
   b. Click the Classpath tab in the Properties dialog box.
4. Click `Edit` next to the Project path field.
5. Start WebLogic Server from inside IBM VisualAge for Java IDE. For instructions, see "Starting and Stopping WebLogic Server from Inside IBM VisualAge for Java" earlier in this document.

   Note: Before starting the server you must make the necessary configuration changes in the `weblogic.properties` file.

6. Run the client in IBM VisualAge for Java.

   Note: If your client's program requires command-line arguments, perform the following procedure:

   a. Select the main class for the example.

   b. Click on Selected → Properties in the workspace window.

   c. Choose the Program tab in the properties window, and enter the arguments in the Command Line Arguments text field.

   d. Press OK.

7. When you have finished developing your WebLogic application in IBM VisualAge for Java, export it to the filesystem so it can be used in a production system. Because EJB jar files are exported as part of the development process, they do not need to be exported. Other code (such as Applet code) is compiled only within the VisualAge workbench during development. You must export them to the filesystem.

   The BEA WebLogic Server development environment contains three directories in which classes for BEA WebLogic Server applications reside:

   - *WebLogic*/myserver/clientclasses -- contains classes required by client applications.
   - *WebLogic*/myserver/serverclasses -- contains classes required by server-side objects.
   - *WebLogic*/myserver/servletclasses -- contains servlet classes.

   In this context *WebLogic* is the directory in which WebLogic Server is installed.

   Classes and resources from your WebLogic Server project in VisualAge should be exported to the appropriate directories. Classes that are shared between the client and server should be exported to both /clientclasses and /serverclasses.

   To export a class or package, complete the following procedure:

   a. Select the class or package.

   b. Click Selected → Export.

   c. Select Directory as the Export destination and click Next.

   d. Enter the appropriate directory in the Directory field.

e. Make sure that `Class` is checked in the `What do you want to export?` section.

f. Select Finish.

Debugging Server-Side and Client-Side Code

1. Create an IBM VisualAge for Java project.
2. To use the IBM VisualAge for Java debugger, any server-side classes must be included in an IBM VisualAge for Java project. You can either create the project in IBM VisualAge for Java or, if the code already exists, you can import it into IBM VisualAge for Java. Client-side and server-side code may be included in the same project.
3. If the object you have developed is a server-side object, add the project to the `Project Path` field of the `Classpath` tab for the `Server` class.
   a. Expand the WebLogic Server project, then the `weblogic.integration.visualage.server` package.
   b. Right-click the `Server` class and select `Properties`.
   c. On the Class Path tab, click the Edit for the Project Path field.
   d. Select the check box for your project, and click OK.
4. Edit `weblogic.properties` as required. When running WebLogic Server from within IBM VisualAge for Java, classes are loaded from the VisualAge workspace, but the `weblogic.properties` file is read from the WebLogic home directory.
5. Within the `WebLogic Server` window, click on `Run` button, or right-click on the `WebLogic Server` and select `Run main...` in the `Run` sub-menu.

You can set breakpoints in server-side code and then trace the code when it is invoked from a client. You may run the client from within or outside IBM VisualAge for Java, whichever is more convenient.

Note: External client calls may time out if the server is stopped at a breakpoint for too long.

In addition, the IBM VisualAge for Java class loader allows you to modify and continue debugging server-side code without restarting WebLogic Server, as long as you change only the content of the object methods. If you make any changes to an object's interface, you must then restart the server.

Note: While debugging an EJB, you may change the content of the EJB methods without rebuilding the jar file. Changes to the EJB's interfaces (remote interface or home interface) or the addition of a new EJB jar file require the use of the `Generate EJB jar` tool and restarting the server.

While it is possible to debug and modify an EJB without rebuilding the jar file, you must rebuild the jar before attempting to deploy the EJB in a server running outside IBM VisualAge for Java.

Developing Servlet Applications

To run and debug servlet projects in IBM VisualAge for Java, you must add a path for the directory in which IBM VisualAge for Java stores the servlet classes to the servlet classpath property in the `weblogic.properties` file:

```
weblogic.httpd.servlet.classpath=\
VisualAge/ide/project_resources/projectName
```

In this line, *`VisualAge`* and *`projectName`* are defined as follows:

- *`VisualAge`* is the directory in which IBM VisualAge for Java is installed
- *`projectName`* is the name of your servlet project

For example, to debug the servlets in the `WebLogic Examples` project, you would set the servlet classpath as follows:

```
weblogic.httpd.servlet.classpath=\
C:/Program Files/IBM/VisualAge for\
Java/ide/project_resources/Weblogic Examples
```

Note: To implement the previous example, IBM VisualAge for Java must be installed in the default location. You can include more than one project in the servlet classpath and of course, you may also include directories that contain packages that are not included in the IBM VisualAge for Java workspace. However, you will not be able to debug any servlets that are not in the IBM VisualAge for Java workspace.

Developing Applet Applications

You can run applets with WebLogic Server in IBM VisualAge for Java in either of two ways:

- You can run the applet with the IBM VisualAge for Java Applet Viewer. This method allows you to debug the applet.
- You can run the applet in a browser. You cannot debug the applet code in this case, because the applet is running in the browser virtual machine, rather than in the IBM VisualAge for Java virtual machine. However, using this method allows you to test the applet in its HTML context.

Running the Applets with the VisualAge Applet Viewer

1. Select an applet class.
2. If the applet requires parameters that are normally specified on an HTML page, click Selected → Properties.
3. Add the required parameters to the Parameter field on the Applet tab. Click OK.
4. Click Selected → Run → In Applet Viewer.

Running the Applets in a Browser

1. Export the applet classes to the *weblogic/myserver/*`serverclasses` directory of the WebLogic Server distribution by following the instructions for exporting code from IBM VisualAge for Java to a production WebLogic Server in the *BEA WebLogic Integration Kit for IBM VisualAge for Java Tutorial*.
2. Copy the HTML file that tests the applet to BEA WebLogic Server's document root directory. This directory is the one that is searched by WebLogic Server when it is looking for public HTML files. By default, this directory is the *weblogic/*`myserver/public_html` directory in your BEA WebLogic Server installation directory.
3. `WebLogic Server` window, click `Run`, or right-click `WebLogic Server` and select `Run main...` in the `Run` sub-menu.
4. In the Web browser, enter `http://localhost:7001/`*appletTest*`.html`, replacing *appletTest* with the name of the HTML file that tests the applet.

Installing WebLogic Service Packs

BEA occasionally distributes service packs as a safe, easy and convenient way to deliver resolutions to product limitations that users can incorporate into the current release of WebLogic Server.

Release 2.1 of The Integration Kit is already configured for Service Pack 8. You should do this procedure only if you would like to install a new service pack.

To install a service pack into BEA WebLogic Server, follow the instructions in the appropriate WebLogic Server documentation, such as the documentation for Service Pack 8 for WebLogic Server Version 5.1.

Service Pack Installation Procedure for Integration Kit

To install any service pack into your Integration Kit, complete the following procedure:

1. Extract the contents of service pack zip file (*weblogic510spX.zip*) and put them in a temporary directory. The *X* in the name of the zip file refers to the service pack number.
2. For the jars in the zip file — *weblogic510spX.jar*, *weblogic510spXboot.jar*, and *WebLogic_RDBMS.jar*, follow these steps:
   a. In the All Projects pane of the Projects tab, select the WebLogic Server project.
   b. From the File menu, select Import.
   c. Select Jar file as the import source. Select Next.
   d. Browse the Filename field and select the service pack jar that you extracted into a temporary directory earlier.
   e. In the area labeled What type of file do you want to import?, choose `.class` and `resource`.
   f. Select the `overwrite resource files` checkbox in the import dialog box.

   Note: If you do not select this checkbox, you will be prompted to select an equivalent checkbox for each class you overwrite.

   g. Select Finish. VisualAge asks whether you want to create editions of classes that are replaced by the service pack. Choose Yes To All.
   h. To keep track of the service packs that you have installed, you can version all of the classes that you have just installed with an appropriate name. To do so, complete the following procedure:
      - From the Selected menu, choose Managed, then Version.
      - Select One Name. In the corresponding field, enter a name that is representative of the service pack that you have just installed, for example *51spXv1*.
   i. Select OK.

Notes About Installation

- Once the service pack is installed, the imported project is referred to as an open edition. Versioning the project allows you to refer to differ editions of the same

project. To find out the different editions of the project, you can use the Repository Explorer to view them.

- Note the order in which you must import the jar files: first `weblogic510spX.jar`, then `weblogic510spXboot.jar`, and finally `WebLogic_RDBMS.jar`.
- IBM VisualAge for Java reports some new problems after importing the jars. You can disregard these problems. These problems appear because when you import the whole service pack jar, you are importing classes that are deliberately omitted from The Integration Kit on purpose, as they are not used in the IBM VisualAge for Java environment (for example - the user-interface related classes). We had decided this was acceptable because the alternative would be giving you a long list of classes and asking you to use the import detail dialog to deselect everything that should not be imported.

Uninstalling Service Packs

1. Select the WebLogic Server project. This project is the one into which you imported the service pack, so it is now the project you want to remove in the Workbench view.
2. Right click the mouse; a menu is displayed. Select `delete`.
3. Click OK to delete the project from the workspace.
4. Click Window → Repository Explorer.
5. Select the `Projects` tab.
6. In the Project Names window, select the `WebLogic Server` project (the project to be removed).
7. In the Editions window, select the project edition that you want to remove from the repository: Specify the name you assigned to the service pack (*`51spXv1`*).
8. Right click the mouse; a menu is displayed. Select Purge…

To replace the original edition of the service pack:

a. From the editions window, select the edition of the service pack you want to put in the workspace.

b. Right click the mouse; a menu is displayed. Select `Add to Workspace`.

The selected project is added to the workspace.

Using the Export Utility

This section explains how to export projects and packages from VisualAge for Java.

Exporting Projects from VisualAge for Java

1. Start VisualAge for Java.
2. Click WorkSpace → Tools → WebLogic Export Tools → Export VAJ. Completing this step may require a long time if you have a large number of projects in your repository.
3. Select a project (for example `WebLogic Examples`) from the tree view. Only the latest versioned project will be used.

   Note: Do not attempt to export core Visual Age for Java classes. The Export Utility is intended to export user developed projects and packages.

4. Enter an Export Directory (for example `c:\temp\ejbexport`). Ensure the path exists.

   Note: If you are using the Browse button, highlight the directory you want before clicking OK. Otherwise, your selection will not be implemented.

   Note: When selecting a Project, the export utility exports all the packages in the selected project from the repository into the workspace before the project can be exported to the file system. If there is new work in the Project workspace that has not been versioned, a warning will be issued and logged. This Project will then be skipped and the export utility will continue with other Projects or Packages. If the Project exists in the workspace and the Project where the package resides has been versioned,

the Project will be deleted from the workspace and placed back into the repository provided that the project is versioned.

5. Click the Export button.

   Note: When selecting a package, the export utility exports the package from the repository into the workspace before the package can be exported to the file system. If the same project to be exported resides in the workspace and the project has been modified or not versioned, then a warning will be issued and logged. A warning will also be issued and logged if there is new work that has not been versioned in the Project where the package resides. This package will then be skipped and the export utility will continue with other packages.

   Note: If the package exists in the workspace and the Project where the package resides has been versioned, the Project will be deleted from the workspace. Since there is a relationship between the Project and the package selected, the entire Project will be deleted from the workspace but still resides in the repository.

   Note: It is recommended that you version the project containing the desired package first, and then delete the project from the workspace before you begin the export.

6. An export Log file (`ExportLog.txt`) is created in the Export directory (for example `C:\temp\ejbexport`).

To view the files that were exported, go to the export directory (`c:\temp\ejbExport`). All resources, `.java` and `.class` files are exported. A list of exported files named `export.txt`. Do not edit or delete this file.

Exporting Packages from VisualAge for Java

1. Start VisualAge for Java.
2. Click WorkSpace → Tools → WebLogic Export Tools → Export VAJ. (This might take some time depending on the number of projects in your repository)
3. Expand the Project node (for example `WebLogic Examples`). All packages under WebLogic Examples are displayed.

4. Select the packages to export (for example `examples.ejb.basic.ContainerManaged`, `examples.ejb.basic.StatelessSession`, `examples.ejb.basic.StatefulSession`)

   Note: Do not attempt to export core Visual Age for Java classes. The Export Utility is intended to export user developed projects and packages.

5. To select multiple packages, hold down the `Ctrl` button while clicking with the mouse.

6. Enter an export directory (for example `C:\temp\packExport`). Ensure that the directory exists.

   Note: If you are using the Browse button, highlight the directory you want before clicking OK. Otherwise, your selection will not be implemented.

7. Click the Export button.

8. An export Log file (`ExportLog.txt`) is created in the Export directory (for example `C:\temp\ejbexport`).

Exporting projects or packages from Visual Age is a single operation. Repeating the export to the same directory will overwrite the `export.txt` file. The data will not be lost, but the list of data in the `export.txt` file will be overwritten.

Using the Import Utility

This section explains how to create a Visual Cafe project from either a VisualAge for Java project or a VisualAge for Java package.

Creating a Visual Cafe Project from a VisualAge for Java Project

1. Click Tools → Import Visual Age Projects/Packages. A dialog box is displayed. Select the file `export.txt` from the export directory which will have a pathname such as `C:\temp\packExport`.

2. Click OK in the dialog box. A new window opens, showing the projects and packages in the export directory.
3. Select the packages you want to import into each associated project. By default, all projects are selected.

Note: Only the first row of each project is used for Template and Project Name Selection.

4. In the Project row, select a type of project template by clicking a `Project Template` cell, such as `Empty Enterprise Bean`. A scroll menu is displayed.

   Note: EJB, servlet, and applet templates will be mapped to their appropriate templates. All other templates will use the empty project template.

5. Specify a Target Directory (such as `c:\VisualCafeEE\projects`) in which to put the project files, either by selecting it or by typing it in the box.

Note: If you are using the `Browse` button, highlight the directory you want before clicking OK. Otherwise, your selection will not be implemented.

6. Click the `Project To Project` button.
7. A log file (`ImporLog.txt`) is created in the selected target directory (such as `C:\VisualCafeEE\Projects`).
8. A project named `WebLogic Examples.vep` is created. The project type is the one you selected earlier: `Enterprise Java Bean`. The files that have been added to the project are listed on the file tab.

   If you subsequently click the `Project to Project` button, a new project `WebLogic Examplesdup.vep` will be created.

   Close the Export Utility by clicking the close button.

9. Check the project type by completing the following procedure:

   a. Select the Project.

   b. Select Project → Options from the menu. Make sure that the project selected is an EJB Application.

   Select Project → Configure Deployment Descriptor. You should be able to see the declarations being made for the EJB.

Creating a Visual Cafe Project from a VisualAge for Java Package

1. Select Tools → Import Visual Age Projects/Packages. A dialog box is displayed. Select the file `export.txt` from the export directory such as `C:\temp\packExport`.
2. Click OK. A new window opens, showing the projects and packages in that directory.
3. De-select any packages you do not want to import into a separate Visual Café Project.
4. Select a type of template to use for each package, such as `Empty Enterprise Bean` Template.

   Note: EJB, servlet, and applet templates will be mapped to their appropriate templates. All other templates will use the empty project template.

5. Select a target directory in which the project files will be placed, such as `C:\VisualCafeEE\projects`.

Note: If you are using the `Browse` button, highlight the directory you want before clicking OK. Otherwise, your selection will not be implemented.

6. Click `Package To Project` button.
7. A log file (`ImporLog.txt`) is created in the selected target directory (such as `C:\VisualCafeEE\Projects`)
8. Check the project type by completing the following procedure:

   a. Select the Project.

   b. Select Project → Options from the menu. Make sure that the project selected is an EJB Application.

   c. Select Project → Configure Deployment Descriptor. You should be able to see the declarations being made for the EJB.

Known Issues with the Export and Import Utility

1. During the import process, projects are created in VisualCafe. Free memory is used up as projects are opened in VisualCafe by the Import utility. Therefore, there is a limit to the number of projects that can be created and opened at a time. This in turn limits the number of projects/packages that can be imported at a time. An informal test revealed that with approximately 128 MB of free memory, approximately 28 new projects could be imported in one go.
2. When a jar file existing under the resources tab in a Project or Package is exported, the VisualAge for Java Export Utility tends to freeze. A fix to this is available from IBM. Please install *`VisualAge for Java Patch 2`* patch that is available for VisualAge 3.5 from the IBM website.

Known Issues with the Integration Kit

1. When installing the Integration Kit, three Weblogic features are added. After they have been added, their entries still appear in the "Add Features" list (see BEA WebLogic Integration Kit for IBM VisualAge for Java Installation Guide). A similar condition occurs if you select "Delete Features" where only one of the features you installed appears. Installing the `Visual Age for Java Patch 2` patch will resolve this problem.