



OpenEdge Architect™

Working with Custom Templates

Matt Baker, October 2007

Edition 1.1 10/04/07

Contents

1.	Introduction.....	4
2.	Technical Overview	4
3.	The Customization Editor	5
4.	Creating New JET templates	5
4.1.	Installing the JDK and EMF.....	5
4.2.	Setting up your project environment.....	6
4.3.	Creating a new JET template	11
5.	Using custom templates	14
5.1.	Configuring OpenEdge Architect template overrides	14
5.2.	Testing the template override.....	15
6.	Sharing your custom templates.....	17
6.1.	Copying compiled templates to a shared location	18
6.2.	Storing your templates with the customization plugin.....	19
7.	Technical Details	19
7.1.	Template Classpath Settings	19
7.2.	Customization plugin version	19
7.3.	Customization preferences descriptions	20
7.4.	Source code for customization plugin	20

Certain portions of this document contain information about Progress Software Corporation's plans for future product development and overall business strategies. Such information is proprietary and confidential to Progress Software Corporation and may be used by you solely in accordance with the terms and conditions specified in your PSDN Subscription End User License Agreement. Progress Software Corporation reserves the right, in its sole discretion, to modify or abandon without notice any of the plans described herein pertaining to future development and/or business development strategies.

1. Introduction

Out of the box, OpenEdge Architect provides a set of wizards that help the user create new ABL files or insert new code structures into existing files. Starting with version 10.1B, OpenEdge Architect includes a Customization Editor, which enables developers to use their own custom templates. A template can work either with a custom wizard (created automatically when the template is installed), or with a standard wizard by overriding the pre-installed standard template.

This paper presents a technical overview, including configuration and examples, of working with templates in OpenEdge Architect. The instructions in this document will

- Walk you through the installation and configuration of the required tools

- Creating a customized template

- Using the custom template in your own development environment

- Sharing the custom template with others

Note: The discussion in this document does *not* pertain to the OpenEdge Architect Tools for Business Logic module, which provides its own mechanism for customizing templates.

2. Technical Overview

Each OpenEdge Architect wizard uses one or more templates to generate its code. For example, the New Class wizard uses the class template. It also uses the constructor and destructor templates if the user chooses the options to generate the constructor or destructor. These templates are based on Java Emitter Template (JET) technology. The Eclipse plug-ins needed to work with JET technology are available from the Eclipse project as a separate download. The templates are similar to Java Server Pages, but are flexible enough to output any sort of code, whether Java, OpenEdge ABL, or any other language.

All of the wizards that ship with OpenEdge Architect 10.1B use JET templates. These JET templates are compiled into Java class files that reside in the plug-in code that is shipped with the product. This document covers the creation and use of JET templates with OpenEdge Architect.

3. The Customization Editor

OpenEdge Architect version 10.1B ships with a tool called the Customization Editor. The Customization Editor enables users to customize certain aspects of the OpenEdge Architect environment. Specifically, it lets users define:

- New menu and toolbar options for the OpenEdge Editor perspective.
- New code-generation wizards based on user-supplied custom templates.
- Overrides of standard templates to achieve customized output from standard wizards.

Figure 1 shows the Customization Editor with the details needed for overriding the New Method template.

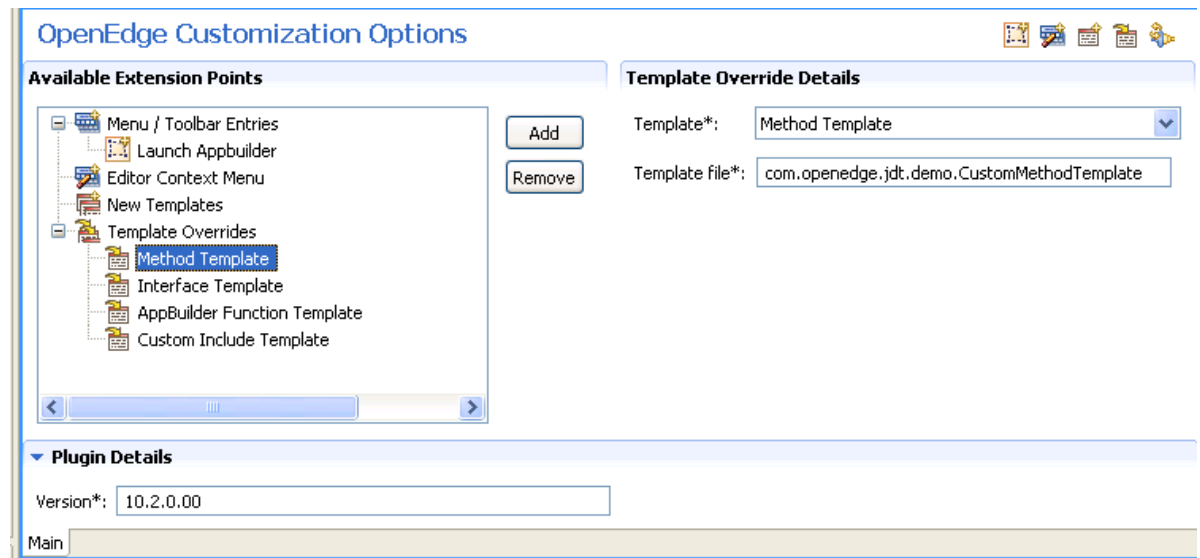


Figure 1. Customization Editor

The Customization Editor works by modifying the configuration files for the extensibility plug-in, **com.openedge.pdt.extensibility.user**. Initially, this plug-in is empty and does not contribute anything to the OpenEdge Architect environment. The editor modifies the extension points needed to create new toolbar and menu or template entries in specific locations. After changes are made, the configuration file for the extensibility plug-in is rewritten and the extensibility plug-in is restarted.

4. Creating New JET templates

OpenEdge Architect uses Java Emitter Templates (JET) to generate code. A JET file is a set of text and code that is used to generate an actual template. Each JET template is compiled into Java code, which is then compiled into Java byte code. This process is mostly transparent and the only thing that needs to be edited directly is the JET file itself. Follow the steps below to create your own custom template.

4.1. Installing the JDK and EMF

OpenEdge Architect does not ship with the Eclipse plug-ins needed to work with JET files. In order to create your own custom templates, you will need to download and install some additional Eclipse

plug-ins first. You can either download the required files directly from the Eclipse Web site, or use the update manager that is built into the Eclipse platform.

Option 1: Download and install the required files manually.

- 1) Download the Java Development Toolkit (JDT) from <http://www.eclipse.org/jdt/>. Make sure you download the JDT version that matches the Eclipse platform that OpenEdge Architect is based on. For OpenEdge Architect 10.1B, this is JDT version 3.2.0.
- 2) Download the Eclipse Modeling Framework (EMF) tools from <http://www.eclipse.org/emf>. The EMF provides the JET functionality. Make sure you download the EMF-SDO-SDK for the Eclipse platform that OpenEdge Architect is based on. For OpenEdge Architect 10.1B, this is EMF-SDO-SDK version 2.2.0.
- 3) After downloading the correct zip files, extract the files to the **oeide** subdirectory of the OpenEdge installation directory.
- 4) Restart OpenEdge Architect.

Option 2: Use the Update Manager to download the files.

Start the Feature Update wizard by selecting **Help>Software Updates>Find and Install** from the OpenEdge Architect menu bar.

Select Search for new features to install and click Next.

- 1) Select Eclipse Modeling Framework (EMF) Updates and The Eclipse Project Updates, and click Finish.
- 2) Turn off the check box option titled “Show the latest version of a feature only”.
- 3) From the list of updates, select Eclipse Java Development Tools and EMF SDK that matches the Eclipse platform that OpenEdge Architect is based on. For OpenEdge Architect 10.1B, this is JDT version 3.2.0 and EMF-SDO-SDK version 2.2.0. Click Next.
- 4) The next page displays the license agreements. If you agree with the license, click the “I accept” radio button and then click Next.
- 5) The next page shows a list of features to be installed and lets you specify the installation directory for each one individually. The default destination is the OpenEdge Architect installation directory. To change this, select a feature and click Change Location. You should specify one or more directories on your local hard drive.
- 6) Click Finish to install the plug-ins. After the installation completes, you are prompted to restart OpenEdge Architect.
- 7) Restart OpenEdge Architect.

4.2. Setting up your project environment

Now that you have the required plug-ins, you need to configure a project to hold your JET templates.

Task 1. Create a new OpenEdge project.

Select **File>New>OpenEdge Project** from the main menu. You can use any project name. For the example described in this document, use **com.openedge.jet.demo**, as shown in Figure 2.

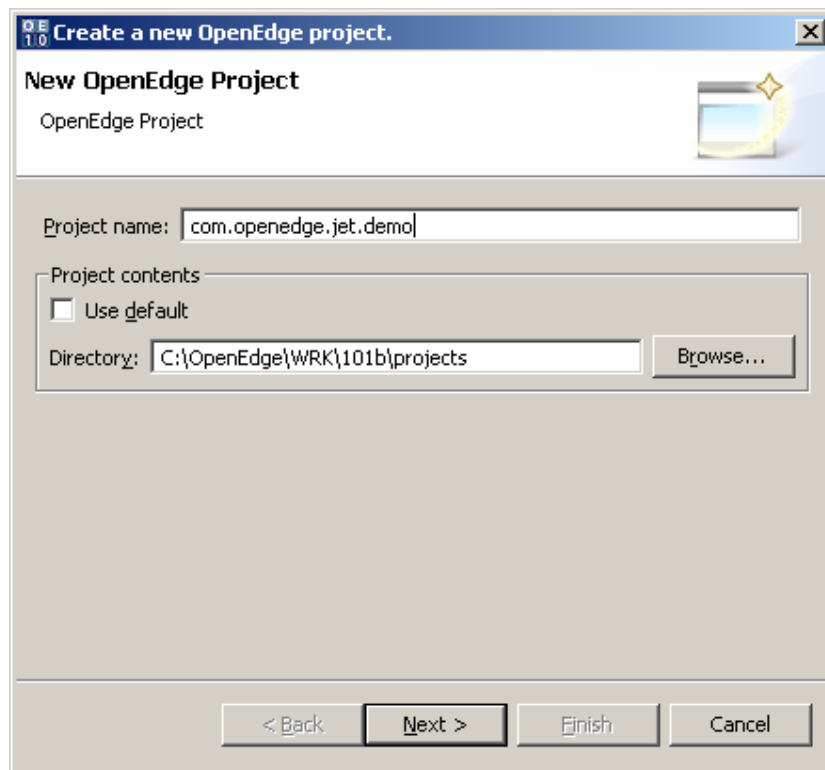
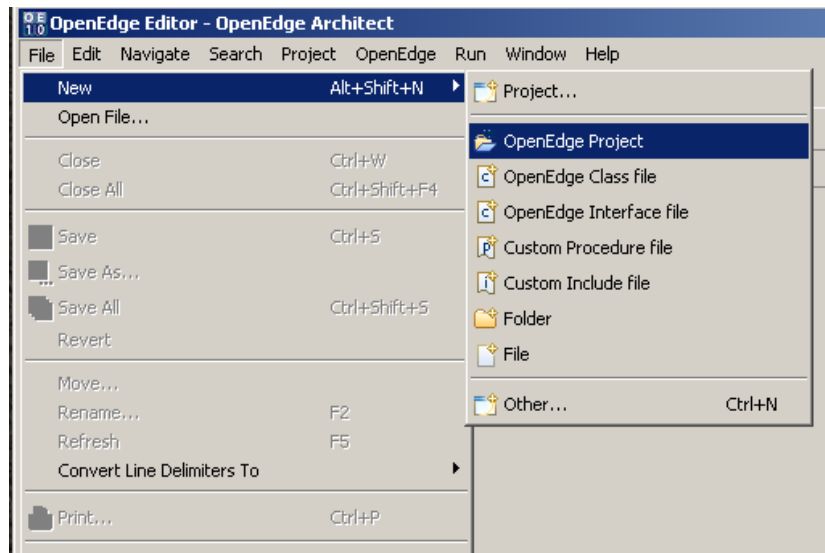


Figure 2. Creating a new OpenEdge project

Task 2. Configure the project to enable JET.

- 1) Select File>New>Other>OpenEdge->JET Templates>Enable JET for OpenEdge Projects from the OpenEdge Architect menu bar. The wizard selection menu appears (Figure 3).

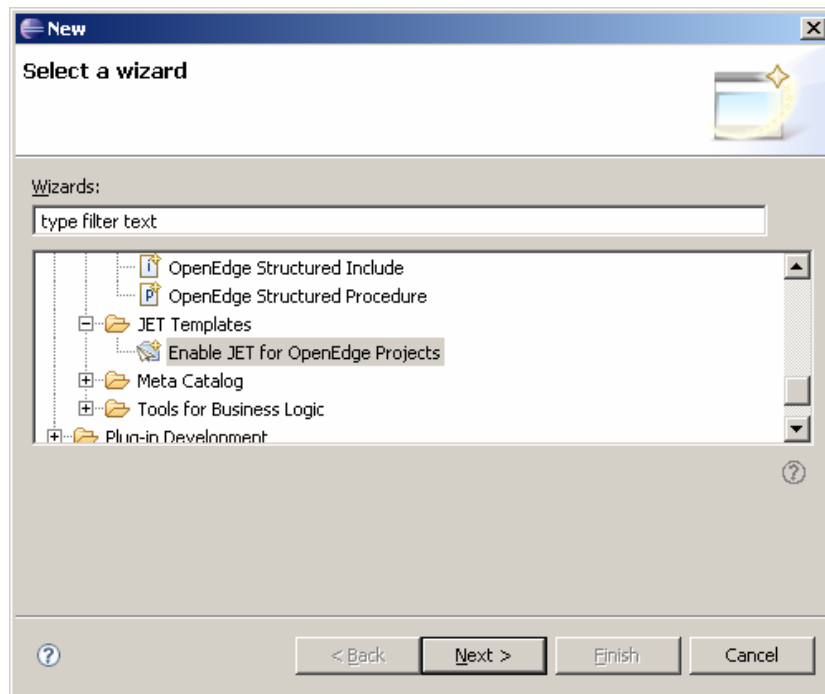


Figure 3. Wizard selection menu

- 2) Click **Next**. You see a list of the OpenEdge projects in your workspace (Figure 4).

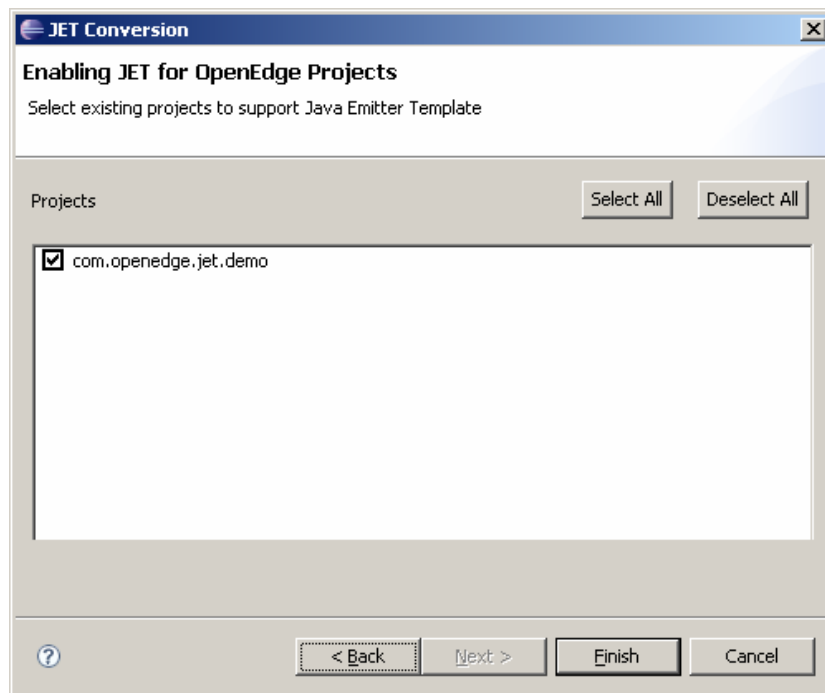


Figure 4. Project list

- 3) Select the project that you just created and click **Finish**.

Your project is now configured to create and build JET templates. The configuration process modifies the project as follows:

The project contains three new directories: **bin**, **src**, and **templates**.

The **.project** file is modified to add the JET builder and JET nature to the project.

A file named **.jetproperties** is created. This file contains the preferences that indicate where JET templates are stored and where the compiled results of the templates will be created.

Figure 5 shows the newly added resources.

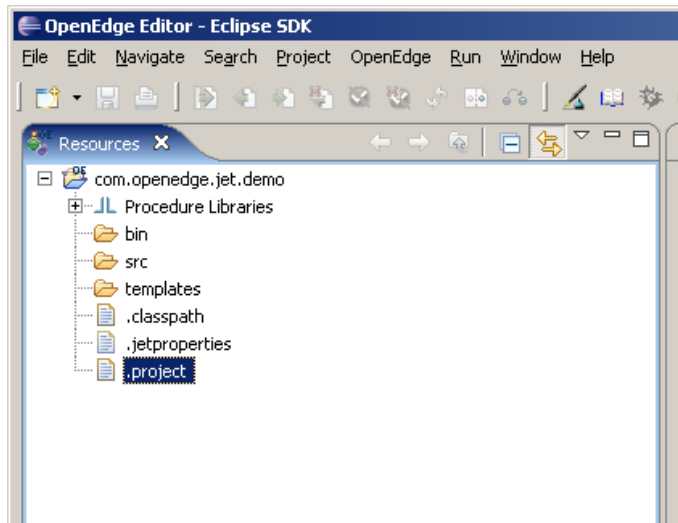


Figure 5. JET-enabled project

When you view the project properties, you will notice a number of new property pages in addition to the OpenEdge Architect property pages.

Task 3. Update the Java build path.

In order for new templates to compile, the Java build path must include certain OpenEdge Architect libraries, found in the **oe_common_services.jar** file.

- 1) Select **Project>Properties** from the main menu.
- 2) Select the **Java Build Path** property page and then select the **Libraries** tab (Figure 6).

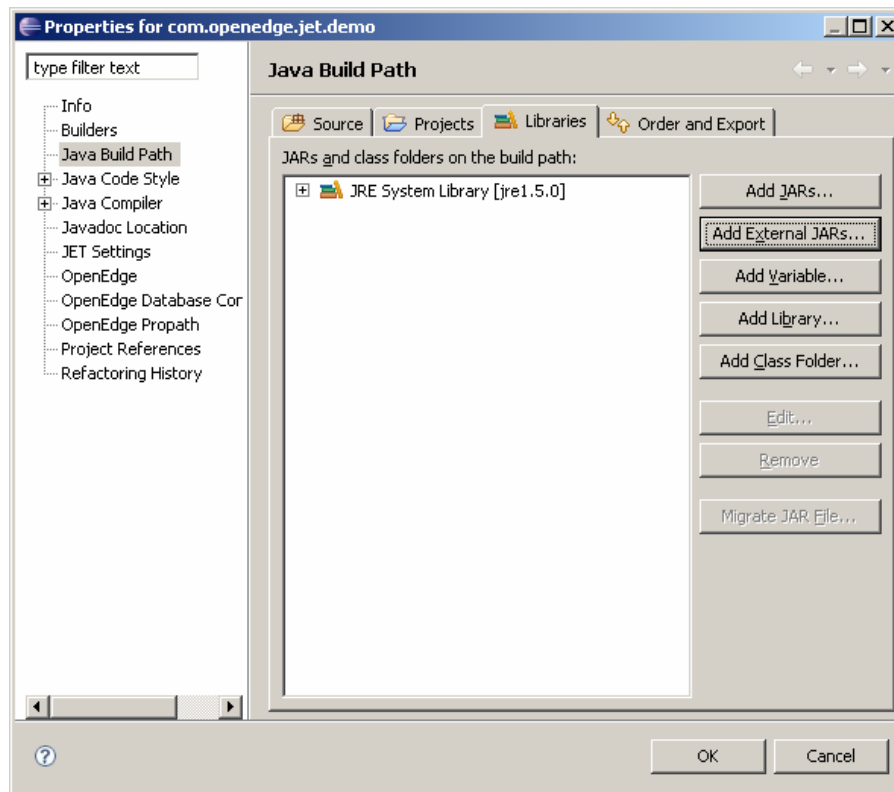


Figure 6. Java Build Path libraries

- 3) Click **Add External JARs** and select **oe_common_services.jar** in the **com.openedge.pdt.core** plug-in directory (Figure 7). You can find the OpenEdge Architect plug-ins in the following directory:

OpenEdge-install-dir\oeide\architect\eclipse\plugins\com.openedge.pdt.core_version

Click **Open**.

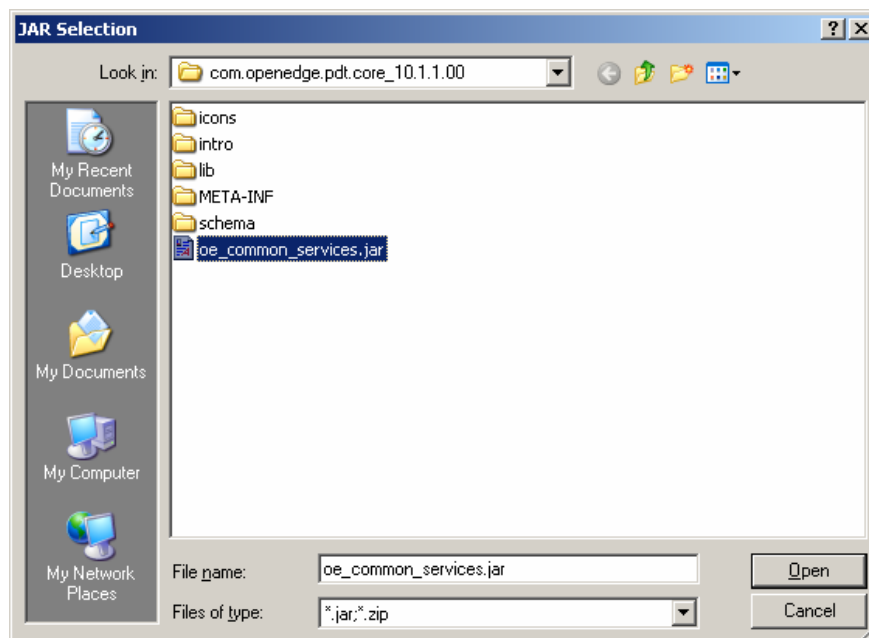


Figure 7. Selecting the oe_common_services.jar file

4) Do not close the Properties dialog yet.

Task 4. Set up JET builder properties.

Before closing the property pages for the project, select **JET Settings** in the tree view. On the Jet Settings property page (Figure 8), enter **src** in the **Source Container** field, and make sure the **Template Containers** field specifies **templates**. Click **OK**.

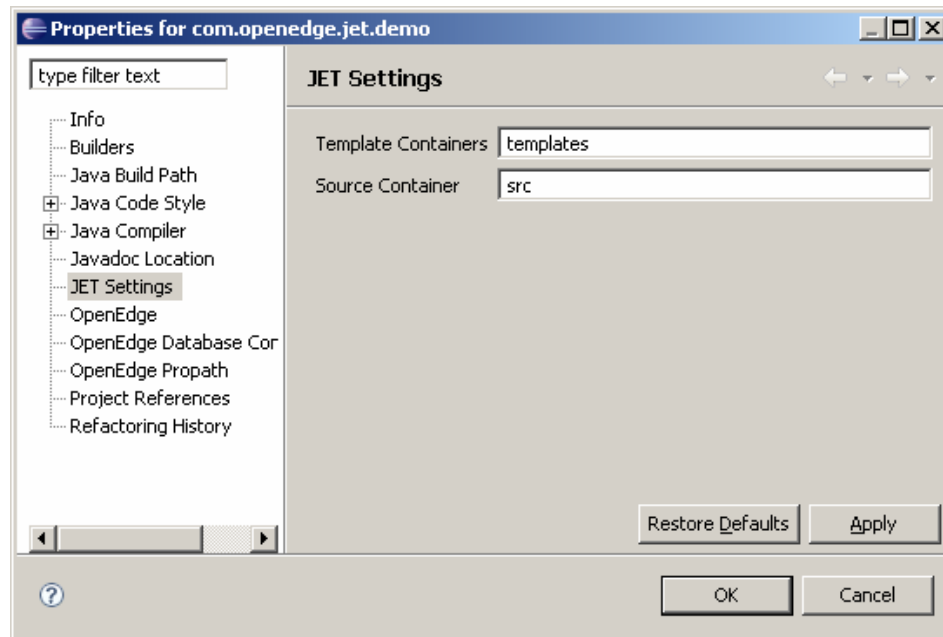


Figure 8. JET settings

4.3. Creating a new JET template

The easiest way to create a new JET template is to copy one from an existing template, and then make the appropriate changes to the copy. In the following steps you'll create a copy of an existing template, modify it to remove the comments, and lowercase all the generated code.

Task 1. Copy the template files.

You can find the templates used by the wizards that ship with OpenEdge Architect in the following directory:

```
OpenEdge-install-dir\oeide\architect\eclipse\plugins\  
com.openedge.pdt.text_version\templates
```

Where **OpenEdge-install-dir** is where OpenEdge Architect is installed, and **version** is the current version of the OpenEdge Architect plug-in (**10.1.1.02** for OpenEdge 10.1B02). For example:

```
C:\progress\openedge101b\oeide\architect\eclipse\plugins\  
com.openedge.pdt.text_10.1.1.02\templates\
```

From this directory, copy two files to the **templates** subdirectory of your Eclipse project directory:

The appropriate source template — For this example, use **method.ijet**. This file contains the actual template code used by OpenEdge Architect to generate a new method.

generator.skeleton — The skeleton file is used as a template by the JET compiler to create the basic structure of the new template.

The Eclipse platform automatically compiles each file as it is created or modified, using two custom builders, the JET compiler and the JDT builder. The JET compiler creates a **.java** file, and the JDT builder creates a **.class** file. In this example, the automatic compilation procedure results in the directory structure shown in Figure 9.

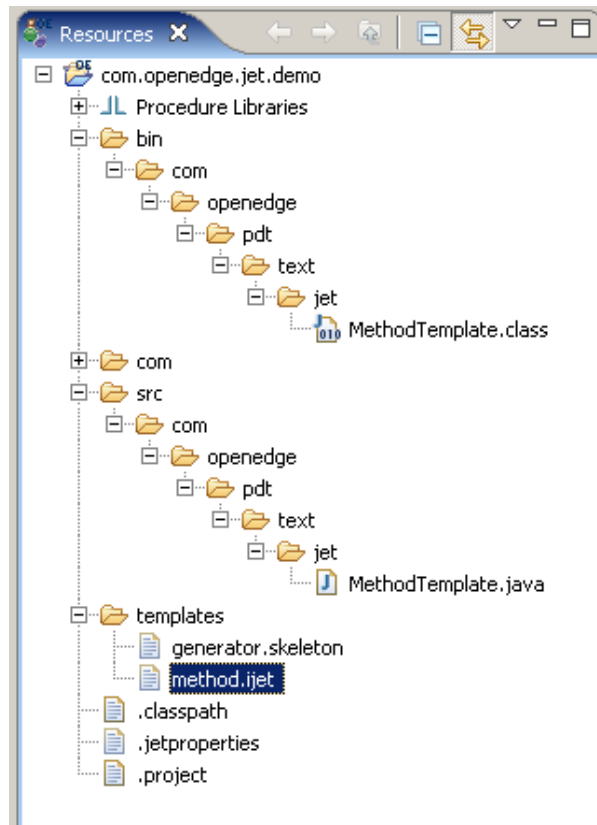


Figure 9. Project structure following template creation

Task 2. Modify the JET template.

Task 3.

- 1) Open the **method.ijet** file. Several annotations at the top of the JET template denote where the compiled version of the template will be created.
- 2) Edit these annotations to make the class name and package structure match the project name:
 - Change the class name from **MethodTemplate** to **CustomMethodTemplate**.
 - Change the package name from **com.openedge.pdt.text.jet** to **com.openedge.jet.demo**.
- 3) Save your changes. This causes the JET builder and the Java builder to run and create a new directory structure within the project. You can now manually delete the two **pdt** folders and their subfolders. Don't worry about deleting the wrong folder. If you do, simply select **Project>Clean** from the main menu and the JET builder will recreate the directories and Java files.

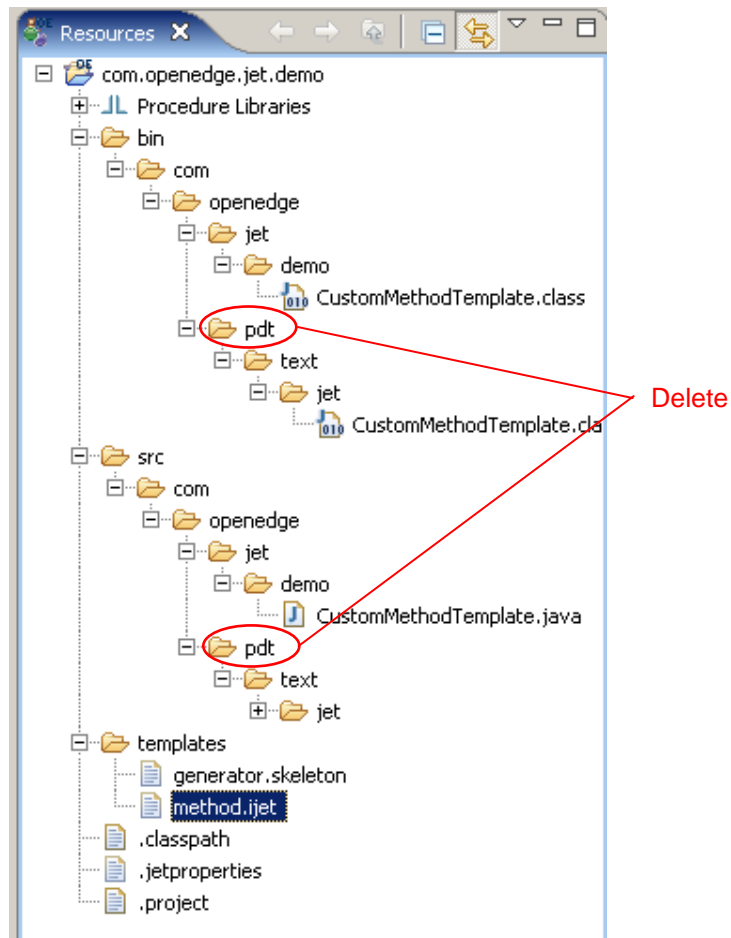


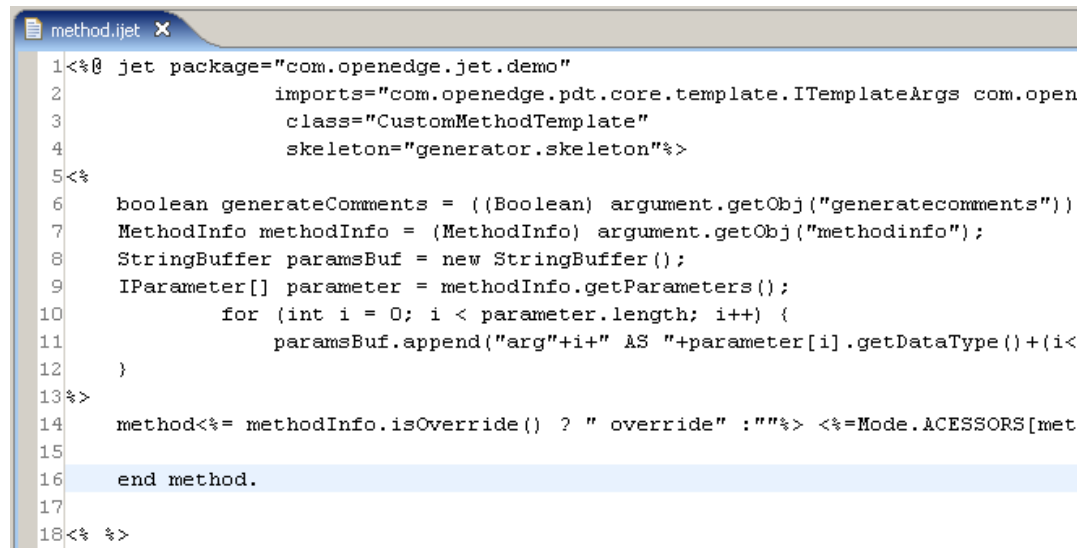
Figure 10. Unneeded folders to be removed

- 4) Now make a few more changes to the template:

Remove the entire comments section.

Change the METHOD, OVERRIDE, FINAL, AS, and END METHOD keywords to lower case.

- 5) Save your changes. The resulting file should look something like this the one shown in Figure 11.



```
1<%@ jet package="com.openedge.jet.demo"
2      imports="com.openedge.pdt.core.template.ITemplateArgs com.open
3      class="CustomMethodTemplate"
4      skeleton="generator.skeleton"%>
5<%
6    boolean generateComments = ((Boolean) argument.getObj("generatecomments"))
7    MethodInfo methodInfo = (MethodInfo) argument.getObj("methodinfo");
8    StringBuffer paramsBuf = new StringBuffer();
9    IParameter[] parameter = methodInfo.getParameters();
10   for (int i = 0; i < parameter.length; i++) {
11       paramsBuf.append("arg"+i+" AS "+parameter[i].getDataType()+"(i<
12   }
13%>
14   method<%= methodInfo.isOverride() ? " override" : ""%> <%=Mode.ACCESSORS[met
15
16   end method.
17
18<% %>
```

Figure 11. Edited method.ijet file

5. Using custom templates

After creating a custom template, you can use it in your development environment with the OpenEdge Architect Customization Editor. The instructions in this section explain how to override the standard method template (used by the Add Method wizard) with the custom method template that you created, and how to test your changes.

By a similar procedure, you can also associate a custom template with a new wizard that you define. See the Customization Editor online help for detailed instructions on these and other procedures.

5.1. Configuring OpenEdge Architect template overrides

You can use a custom template in place of one of the standard templates installed with OpenEdge Architect. In this example; you configure OpenEdge Architect to use your custom template when creating a new method in an existing class.

- 1) Start the Customization Editor by selecting **OpenEdge>Tools>Customization Editor** from the main menu.
- 2) Select **Template Overrides** in the tree view and click **Add** to create a new override entry.
- 3) Select the **Method Template** entry from the combo box on the template override details page.

- 4) Enter `com.openedge.jet.demo.CustomMethodTemplate` in the **Template File** field (Figure 12).

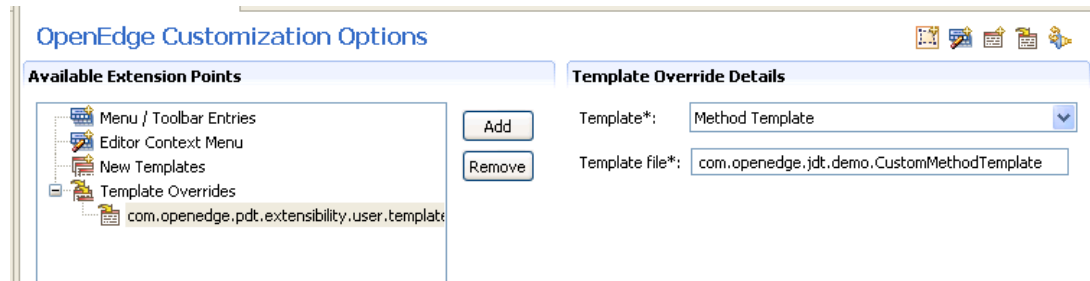



Figure 12. Overriding the Method template

- 5) Save your changes by selecting **File>Save** from the main workbench menu.
- 6) To make your changes immediately available, refresh your Workspace by clicking  , the rightmost button on the Customization Editor toolbar. Otherwise, the changes don't take effect until you restart OpenEdge Architect. A dialog will be displayed a couple of times prompting you to reset your current perspective. This happens as the customization plug-in is uninstalled and then reinstalled. Choose yes to both prompts.

5.2. Testing the template override

The final step is to use the Add Method wizard so that you can verify that your changes are in effect. You need to be editing a class file to use this wizard, so begin by creating a new class.

Task 1. Create a new class file.

- 1) Select **File>New>ABL Class**.
- 2) Name the new class **MyClass** (Figure 13).

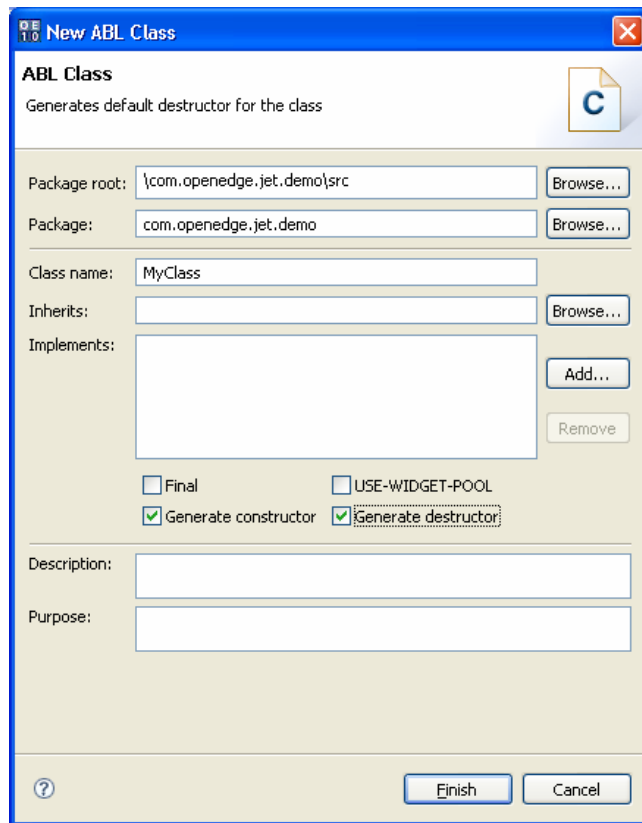


Figure 13. Creating a new OpenEdge class

Task 2. Add a new method to the class.

- 1) With focus on the Editor buffer for the new file, start the Add Method wizard by selecting **Source>Add Method** from the main menu (Figure 14).

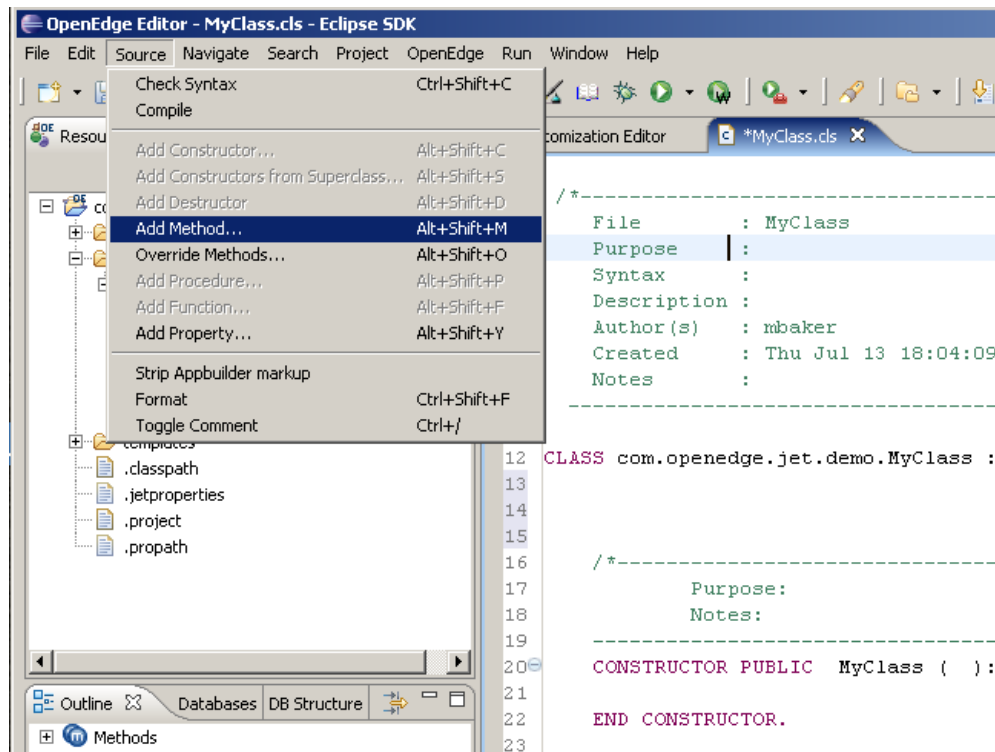


Figure 14. Starting the Add Method wizard

- 2) Specify a method name and a return type, and click **Generate** (Figure 15).

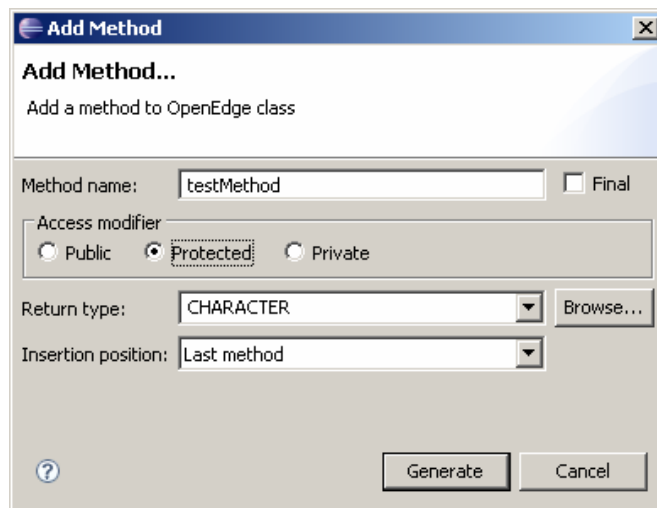


Figure 15. Add Method wizard

The new method wizard should now insert a method called “testMethod” into your code without the comment headers and with all of the keywords in lower case.

6. Sharing your custom templates

Templates can be shared with other people or with other workspaces. To share a template you will need to copy the compiled version of the template to an accessible location where it can be found. You will also need to configure OpenEdge Architect to point to the location of the copied template.

There are two ways this can be done; by either copying the compiled templates to a shared location, or by putting the compiled templates directly into the customization plugin.

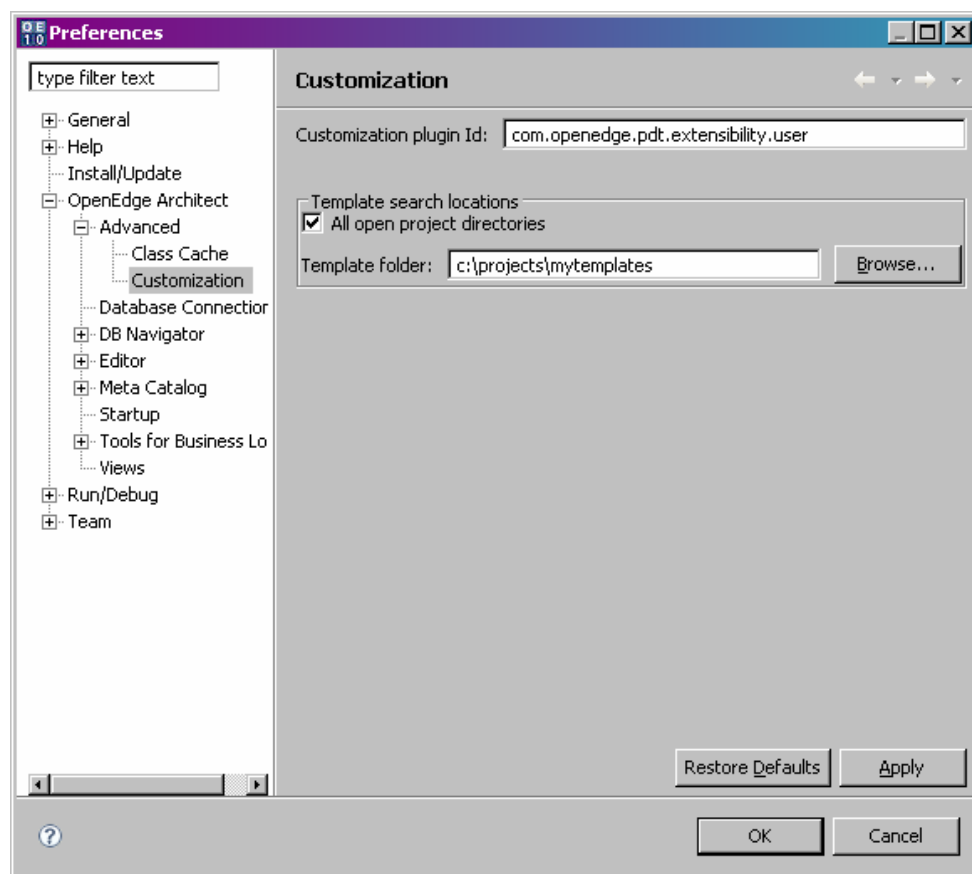
6.1. Copying compiled templates to a shared location

For each template, the Java compiler creates a `.class` file in the project's `bin` directory. To share a template with someone else or another workspace you will need to put the `.class` files that were created by the java compiler in the 'bin' directory of your project to a location where they can be found.

Step 1: Select Window->Preferences from the main menu.

Step 2: Open the customization preferences page by selecting OpenEdge Architect->Advanced->Customization from the preferences dialog.

Step 3: Set the template folder setting to read "c:\projects\mytemplates".



Step 4: Press the OK button to close the preferences dialog.

Step 5: Create a directory on your hard drive named "c:\projects\mytemplates"

Step 6: From Windows™ Explorer copy the contents of the "com.openedge.jet.demo/bin" directory to the "c:\projects\mytemplates" directory on your hard drive. Make sure to preserve the directory structure so that the package structure of the java class files is preserved.

6.2. Storing your templates with the customization plugin

The other way to share templates is directly with the customization plugin itself. The class path for finding compiled templates has the 'bin' directory for the customization plugin automatically added to it. To take advantage of this:

From Windows TM Explorer, copy the contents of `com.openedge.jet.demo/bin` directory to the `bin` subdirectory of your customization plugin directory. Make sure to preserve the directory structure so that the package structure of the java class files is preserved. This directory is usually at `<OpenEdge Installation>/oeide/architect/eclipse/plugins/com.openedge.pdt.extensibility.user_1.0/bin`. If this directory doesn't exist, then simply create it.

7. Technical Details

This section is here for anyone who wants to understand how templates work in OpenEdge Architect. If you intend to share your templates with others some of this information will probably be useful.

With the exception of Tools for Business Logic, all the templates used by OpenEdge Architect are handled the same way. All of the templates shipped with OpenEdge Architect are based in Java Emitter Templates (JET).

Each template used by OEA is assigned a unique identifier. The individual wizards use the unique to find the correct template and generate code. The resulting code is then inserted into a new or existing file depending on the wizard. The template lookup uses extension points defined by the `com.openedge.pdt.core` plugin to discover which templates are overridden with custom templates.

7.1. Template Classpath Settings

Because JET is based on java technology a class path (similar to `PROPATH`) is used to search for the location of template class and to load its dependencies. The two options in the preferences control how this classpath is generated. By default, only the classpath for the plugin in which a template resides is used. The two options in the preferences can be used to customize how the class path is calculated. Choosing the "All open project directories" option appends each open project's root directory as well as the "bin" directory of each project, if it exists, to the classpath. Specifying a directory for the "Template folder" option in the preferences appends that directory to the classpath as well.

By default, the `com.openedge.pdt.extensibility.user` plugin has dependencies on the core and text plug-ins from OEA. This means that any classes in these two plug-ins and their dependencies are available to use as part of your custom template. Since the source code for the customization plugin is shipped with OEA, it can be imported as a project into an Eclipse installation and modified directly.

Because Eclipse extension points are used to provide template customization, you can create your own Eclipse plugin and use the extension points defined in OEA to provide your own templates.

7.2. Customization plugin version

The customization plugin provides a version number field to uniquely version your plugin. If you intend to share your changes or preserve them across different installations or service packs, be sure to modify the version number to something higher than 1.0. This will create a copy of the

`com.openedge.pdt.extensibility.user` plugin with the new version number. Doing this will ensure that the installation of a service pack will not overwrite your changes.

7.3. Customization preferences descriptions

On the customization preferences page, there are two settings that can be used to specify where to search for templates. The first setting “All open project directories” appends the root directory and the “bin” directory of each project to the search path for the templates. The second setting “template folder” appends the specified folder to the search path for the templates.

The template folder settings specified above are appended to the class path for the `com.openedge.pdt.core` plugin. This means that only the classes that are available to the `com.openedge.pdt.core` plugin are available for use in your custom template.

7.4. Source code for customization plugin

The source code for the customization plugin project is shipped with OpenEdge Architect. You can find it at:

`OpenEdge-install-dir\oeide\architect\eclipse\plugins\com.openedge.pdt.extensibility.user_1.0\`

The `.project`, `plugin.xml`, and `.classpath` files are included so you can import the project into an existing workspace and then modify and compile the JET templates directly without having to copy them to an alternate directory. This plugin is initially empty, but may be modified to include additional items as necessary.