Getting Started with Progress Developer Studio for OpenEdge

When you start Progress Developer Studio for OpenEdge for the first time, you need to specify a workspace location.

A workspace location is a folder in your file system where all of your development projects files reside. A workspace is used by a single developer. A best practice is to physically locate your projects under the workspace folder.

Here we specify the location for our new workspace. In addition, we specify that we want this location to be the default workspace. If you later want to share the projects and files of your workspace with another developer, you can export them. Once we have specified the workspace location, Developer Studio opens a Welcome page. We click the Workbench icon to open our new workspace.

Here is our new workspace that is ready for use. By default, a workspace is configured with a set of preferences for viewing and working with projects. In many cases, these default settings are sufficient. However, you may want to customize these settings. For example, your development team may have decided to standardize on lower case ABL keywords.
To set this preference, we open the workspace preference area and drill down to the Progress OpenEdge Editor settings.

Here we see a number of settings. To specify that we want to use lower case, we select the **Lower** radio button, and then we select **Case keywords** and **Apply keywords casing on save**. If you are new to ABL, there are a number of other settings you will find useful. For example, by selecting **expand keywords**, the editor will automatically complete keywords as you type.

Another useful group of settings for a new developer is to enable all of the syntax checking features. You do so by navigating to the Build settings page for OpenEdge Editor.

Here we select **compile on save if required**, all of the syntax checking settings, and we increase the number of errors that will be displayed when the application compiles. These settings will check syntax as you type and will display more error information to help you correct your code.

Another useful setting is to display line numbers. Displaying line numbers makes it easier to communicate with other developers and will also help you debug your code. For this setting, you navigate to the general **Text Editors** area and select **show line numbers**.

We save our preferences for our workspace.

Next, let's see how easy it is to import a project into a workspace. You typically do this when another developer has provided you with a project that you will work on. To import a project, you select **Import** and then navigate to the General area where you select **Existing Projects into Workspace**.

The best way to import a project is to import a zip file. To do this, we select an archive file by browsing to the zip file, and then select the **Server** project that we want to import into our workspace.

After we have completed the import, the new Server project appears in the Project Explorer pane. Notice that it has two sub-folders – **bin** which contains compiled code and **src** which contains the source code. A development team should agree upon naming conventions for these top-level folders for projects.

In addition, the Console view displays a message that an ABL Virtual Machine or AVM has started for this project. An AVM is the compile and runtime environment for ABL code. By default, every OpenEdge project in your workspace uses its own AVM.

Whenever you import a project into your workspace, a best practice is to recompile your application. You do so by cleaning your workspace which compiles all code in your workspace.

In our case, notice that a red x appears in the **src** folder for the project where you can identify which particular file has an error.

The Problems view provides you with compiler warnings and errors in your code.

When you hover over an error, you can see more information about the error.

In addition, if you double-click the error, the file where the error occurred is opened in OpenEdge Editor. Notice that the red x is used to show the line where the error occurred.

All ABL statements must end with a period. A period is missing from this code, so we correct the error.

When we save the source file, it now automatically compiles without error.

Next, let's create an OpenEdge project from scratch. We open the New OpenEdge Project wizard and name the project **Client**.

We want to use **src** and **bin** for the top-level project folder names so we make the necessary changes.

We complete creation of our Client project using the default values for everything else.

The newly-created project appears in the Project Explorer view. Notice that it's top-level folders are **bin** and **src**. You can create sub-folders under **src** to organize your application code.

Here we create the **GUI** and **Adapter** folders. You do not create sub-folders under the **bin** folder as they are created automatically when you compile your code.

[CustomerBEServiceAdapter]
To add existing source files to a project, you import them by dragging and dropping files from your file system into the appropriate project folders. Here, we add an ABL class file to the **Adapter** folder. A best practice is to always copy files into your projects, rather than link to them.

[GetCustomerInfo.cls, GetCustomerInfo.resx, assemblies.xml]
Next we add an ABL Form class file along with its resources to the GUI folder and and we add the Telerik library assemblies to the project.

[RunClient]
And finally, we add an ABL procedure file to the top-level **src** folder.

After adding these files to the **Client** project, we see errors in the **Problems** view. The errors you see when you copy files into your projects could be related to the order in which the files were copied.

A best practice is to clean your workspace each time you copy a set of files into your workspace.

[RunClient.p]
Now let's look at using OpenEdge Editor. First, let's open a procedure file.When you double-click a file containing ABL code, it opens in a new tab in the OpenEdge Editor view.

Notice that the Outline view displays the variables and using declarations defined in the procedure file.

When you click on a definition in the outline view, it takes you to that location in the editor.

[CustomerBEServiceAdapter]
Now let's open an ABL class file in the editor.

In this case, the outline view shows the data members and methods of a class.

We now have multiple files open in our workspace. If you click the "link with Editor" icon in Project Explorer, the file you are currently viewing in the editor is highlighted in Project Explorer so you can see its context within the folder structure of a project.

[GetCustomerInfo.cls]
Form class files, for example, a class representing a Telerik for WinForms UI, can be opened in either OpenEdge Editor or Visual Designer. Visual Designer is used to add controls to a form. OpenEdge Editor is used to modify ABL code in the form.

We open the form in OpenEdge Editor. Then we open the file in Visual Designer.

Here we see that we have the same file open in both views.

[GetCustomerInfo.cls]
In OpenEdge Editor, you can expand and hide parts of your source code to make the context of your code easier to understand.

Creating an ABL procedure or class file is easy with the Wizards provided by Developer Studio.

[RunClient.p]
[def var iCustNum as int init 500.]
OpenEdge Editor uses colors to represent the components of an ABL statement. As we type ABL code, the editor will automatically complete keywords because we configured this setting for the workspace. Here we type a simple ABL statement to define an integer variable. Notice that the ABL keywords are in a different color than the name of the variable. In addition, known ABL types such as integer appear in yet another color, and constants appear in blue.

[def var cCustName as string.]
If the ABL statement we type has a syntax error, we receive immediate feedback because we enabled all of the syntax checking features for our workspace.

If we save our file, a compilation error appears in the Problems view.

After we correct the error and save the file, the Problems view is clear.

We execute the code in the current procedure file by clicking the **Run** icon.

The form appears, but when we click the button to retrieve the customer name, a runtime error occurs. This is because the AVM for the client project cannot find the **ServiceInterface** folder which is located in the Server project.

To fix this, we must modify the project properties for the **Client** project, so that the source and compiled code of the **Server** project is available to the client code. We open the properties for the Client project.

Here we navigate to the Progress OpenEdge PROPATH area and add the **src** and **bin Server** folders to the OpenEdge PROPATH for the Client project's properties.

After this change is made for the Client project, we can successfully run the application in Developer Studio.

[GetCustomerName]
You may also find that you want to comment or uncomment your code. You can comment out a line or section of code by selecting the code, and then selecting that you want to comment the source. Notice that commented code is displayed in green.

To close the OpenEdge Editor or Visual Designer view for a file, you simply close the tab.
You have now been introduced to Progress Developer Studio for OpenEdge. You see how to set up your ABL development environment, import and create projects and application files, and use the OpenEdge Editor perspective to work with ABL application code.

To learn about developing OpenEdge ABL applications, take the course, Developing a Progress OpenEdge ABL Application.