

OpenEdge Architect Tutorial 2009

Copyright © 2008 Progress Software Corporation

Progress[®] software products are copyrighted and all rights are reserved by Progress Software Corporation. This manual is also copyrighted and all rights are reserved. This manual may not, in whole or in part, be copied, photocopied, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Progress Software Corporation.

The information in this manual is subject to change without notice and Progress Software Corporation assumes no responsibility for any errors that may appear in this document.

The references in this manual to specific platforms supported are subject to change.

Progress[®], OpenEdge[®], Progress[®] OpenEdge[®] Architect and *Progress[®] OpenEdge[®] RDBMS* are registered trademarks of Progress Software Corporation.

All company and product names are the trademarks or registered trademarks of their respective companies.

Printed in the USA

December 2008
Version 2.0

Progress Software Corporation is concerned about the environment. To reduce waste and complete the recycling circle, we printed this manual and cover on stock that is recyclable. PSC has made every effort to look at environmental implications when deciding on this package.

Author: James Willis

Tutorial Overview

Overview

In this tutorial you will be introduced to the OpenEdge Architect development environment and how its features and tools make it easy to develop OpenEdge applications.

When you complete this workshop, you should be able to:

- Group and organize application files using Architect's project-based environment.
 - Use Workspace Preferences and Project Properties to configure a development environment.
 - Use Views and Perspectives to navigate and maintain resources.
 - Use the DB Navigator tool to view and modify database schema.
 - Use the OpenEdge Editor and its features to write application code.
 - Use Architect's Debugger to debug applications.
 - Use the Tools for Business Logic to create ProDataSet and Temp-table models and auto-generate ABL code from them.
 - Use the Visual Designer to create OpenEdge GUI for .NET application screens.
-

Lab 1 – Setting up the Environment

Overview

OpenEdge Architect uses workspaces and projects to store and group application files. Together, these features provide a robust way to organize and maintain source code and other files associated with an application. Workspace Preferences and Project Properties are used to configure and manage the working environment. The goal of this lab is to set up a working environment for the AutoEdge application by creating a workspace and a project, importing files, setting the Propath, and connecting databases.

The AutoEdge application is an example implementation of the OpenEdge Reference Architecture (OERA). The application revolves around a fictitious automobile dealer company and two dealerships. Information from each dealership is shared with the parent company that handles consumer facing functions.

For more information about the AutoEdge application and the OERA, please visit the Progress Software Developers Network at www.psdn.com.

Prerequisites

The OpenEdge Architect product and the AutoEdge application source code and database files should reside in the following directories before starting this lab:

1. OpenEdge 10.2A (Enterprise Architect, OpenEdge Ultra .NET Controls) installed in the **C:\Progress\OpenEdge** directory
2. AutoEdge application located in the **C:\OpenEdgeTutorial2009** directory with the following sub-directories:
 - **AutoEdge** - This directory contains the source files for the AutoEdge application.
 - **AutoEdgeDB** - This directory contains the different AutoEdge databases in their own subdirectories (autoedge, oerpcore, temp-db). This directory also contains the two scripts for starting and stopping the databases (**startdbs.bat** and **stopdbs.bat**).
 - **Tmp** – This directory is used to store temporary files, such as log files, for the environment. If this directory does not already exist, please create it.

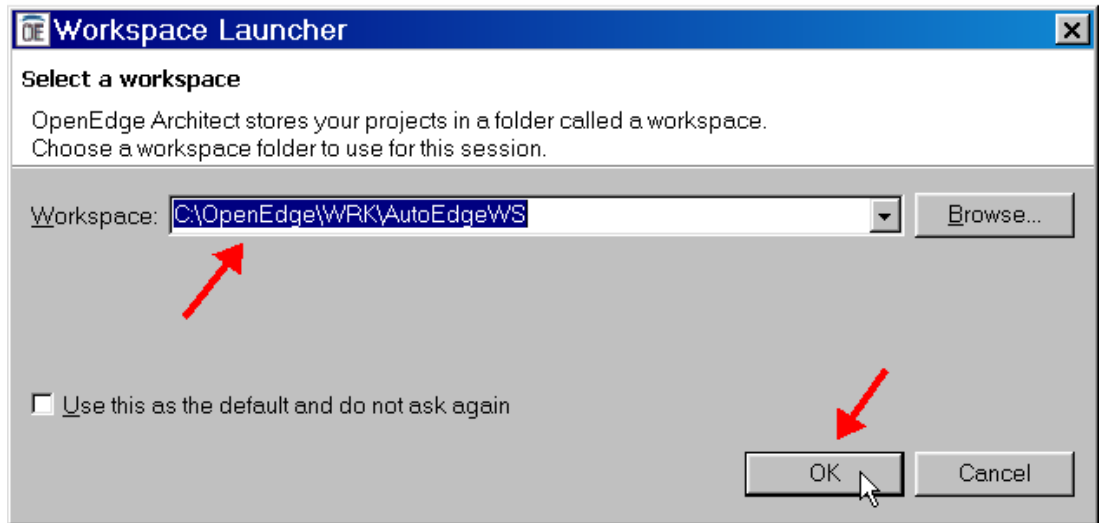
continued on next page


Lab 1 – Setting up the Environment, continued

Starting Architect and creating your first Workspace

This section shows how to start OpenEdge Architect.

1. Start Architect by selecting **Start→Programs→OpenEdge→ OpenEdge Architect**. The Workspace Launcher dialog appears.
2. Set the Workspace to **C:\OpenEdge\WRK\AutoEdgeWS**. This will create a new physical directory at that location (if it does not already exist) and set up the basic Workspace environment.



 **Tip:** As shown above, there is an option to set the currently configured workspace as the default workspace. Setting this option will stop Architect from prompting for a workspace every time you open the tool and automatically open the given workspace. You can switch to other workspaces at any time by selecting **File→Switch Workspace**. It's common for developers to work with multiple workspaces, so many leave this option unchecked since they find it convenient to be prompted for the workspace to open when starting Architect. This option, along with many other workspace options, can be modified using Workspace preferences, which will be covered later in the lab.

continued on next page

Lab 1 – Setting up the Environment, continued

Starting Architect and creating your first Workspace, continued



- Note:** The following are some important notes regarding workspaces:
- Multiple projects, even of different types, can co-exist within a workspace.
 - All files created during the application development process using Architect are contained in a workspace.
 - Architect monitors changes made to any resource files and keeps information about those files. This includes code errors, file reference dependencies, and file locations. This information is kept in the Workspace's state data files.
 - Changes made to source code files outside of Architect's environment will not be reflected in Architect until a refresh is made on the workspace. This is usually accomplished by selecting the File→Refresh menu option.
 - You may have multiple workspace areas set up for your development environment.
 - Workspaces can be physically located anywhere on your system or on a network drive. However, workspaces can only be used by one developer at a time. Because of this, it makes sense to keep your workspaces local to your computer. However, a workspace's projects and resources can be shared either through exporting/importing or by using Source Code Management plug-ins.



Note: What if you want to move a Workspace to a different physical directory? This can be done, but you will need to make a few changes. Use Windows Explorer to either move or make a copy of the workspace. Start Architect and enter the directory location for the new Workspace in the Workspace Launcher dialog. Once you start Architect, you will need to change any properties that were pointing to the old physical directory such as the PROPATH. The changes can be minimized by using Configuration Variables (see on-line help for information on Configuration Variables). One area that will need to be changed is the working directory in the project settings. To change the working directory, go to the OpenEdge project properties and select the Browse button.

continued on next page

Lab 1 – Setting up the Environment, continued

Starting Architect and creating your first Workspace, continued

3. Click **OK**. An Architect splash screen will display the loading status.

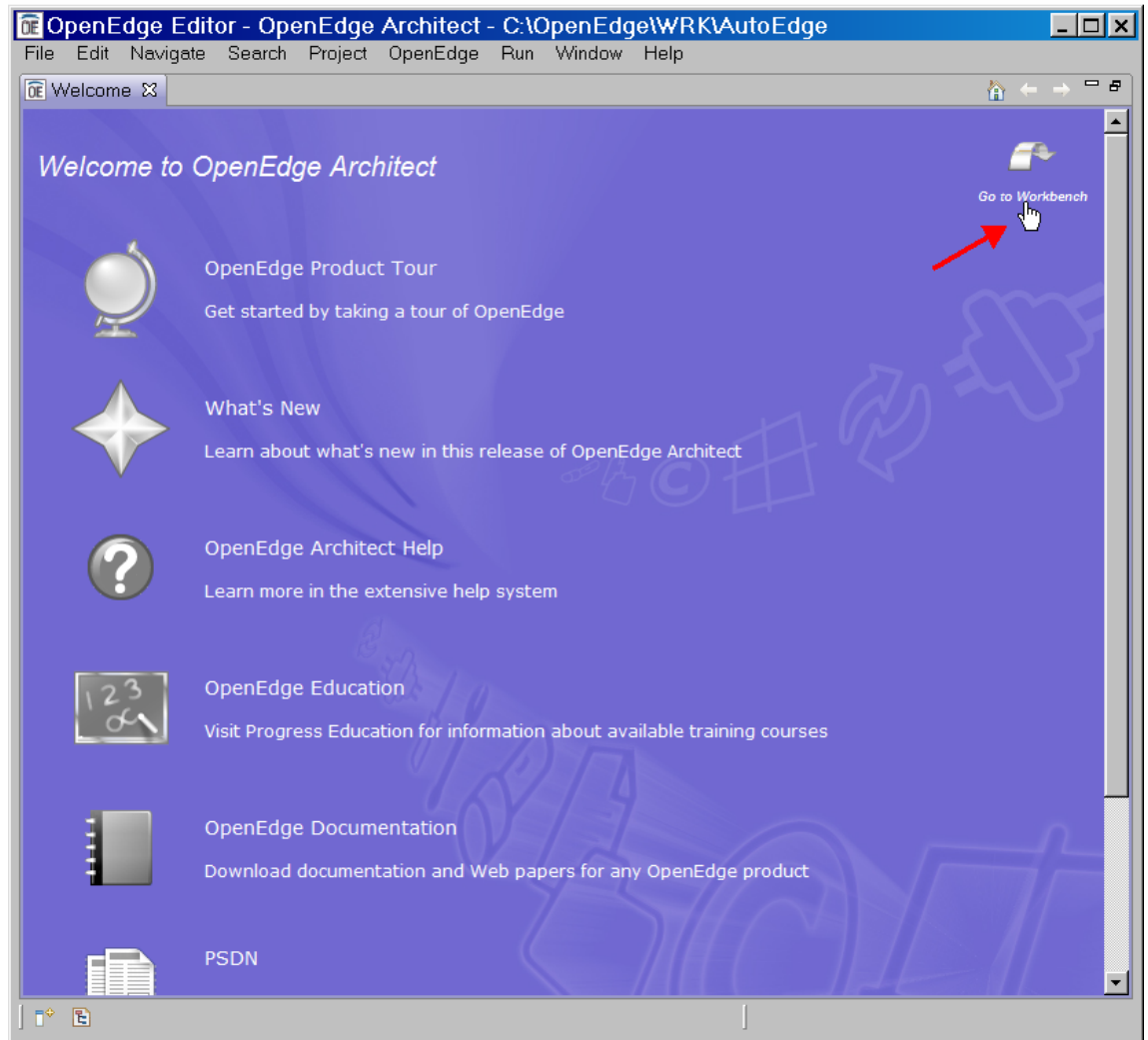


continued on next page

Lab 1 – Setting up the Environment, continued

Starting Architect and creating your first Workspace, continued

4. A Welcome to OpenEdge Architect screen appears with links to common tutorial and help resources for the Architect product. Select the **Go to Workbench** icon in the top right corner of the screen.

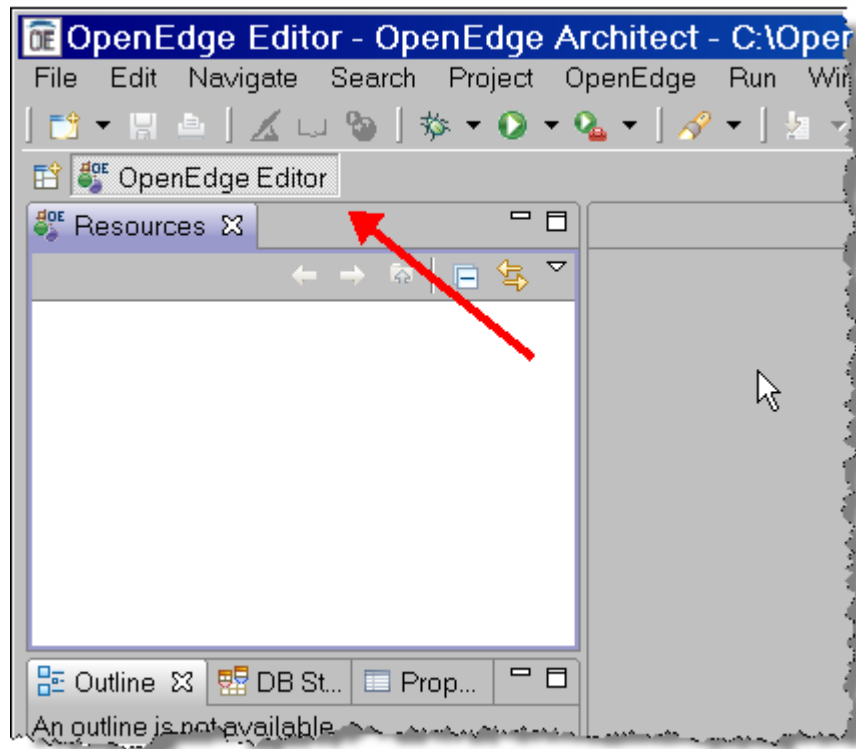


continued on next page

Lab 1 – Setting up the Environment, continued

Starting Architect and creating your first Workspace, continued

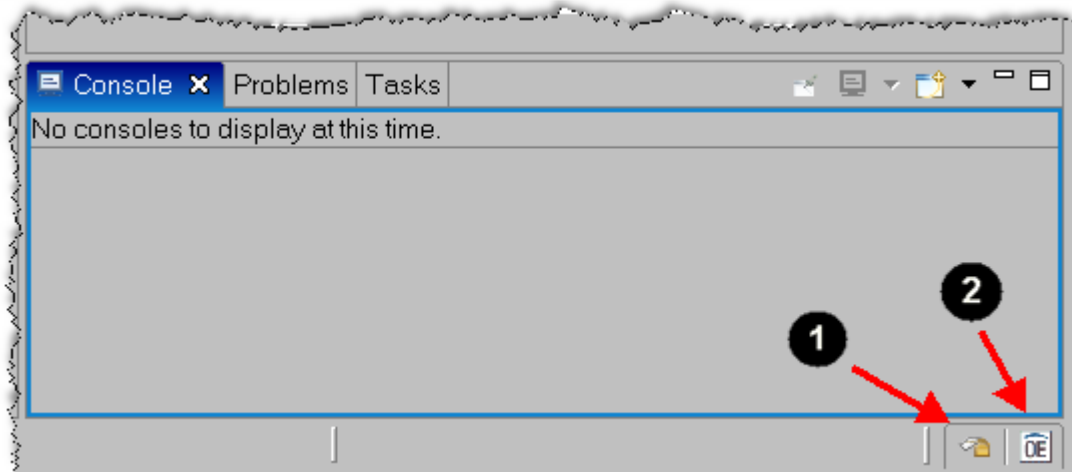
The Workbench is displayed showing the **OpenEdge Editor** perspective. The workspace will have been created and you are ready to create your first project.




continued on next page

Lab 1 – Setting up the Environment, continued

Starting Architect and creating your first Workspace, continued



 **Tip:** The Welcome screen is hidden and can be easily recalled by selecting the Return to Welcome icon (❶) in the lower right corner of the screen. Also shown above is the OpenEdge Architect Help icon (❷) which opens the OpenEdge Architect Help screen. Both of these icons can be removed by right-clicking on the icon and selecting Close. Another way to open the Welcome screen or the OpenEdge Architect Help screen is to select their respective options from Workbench's Help menu.

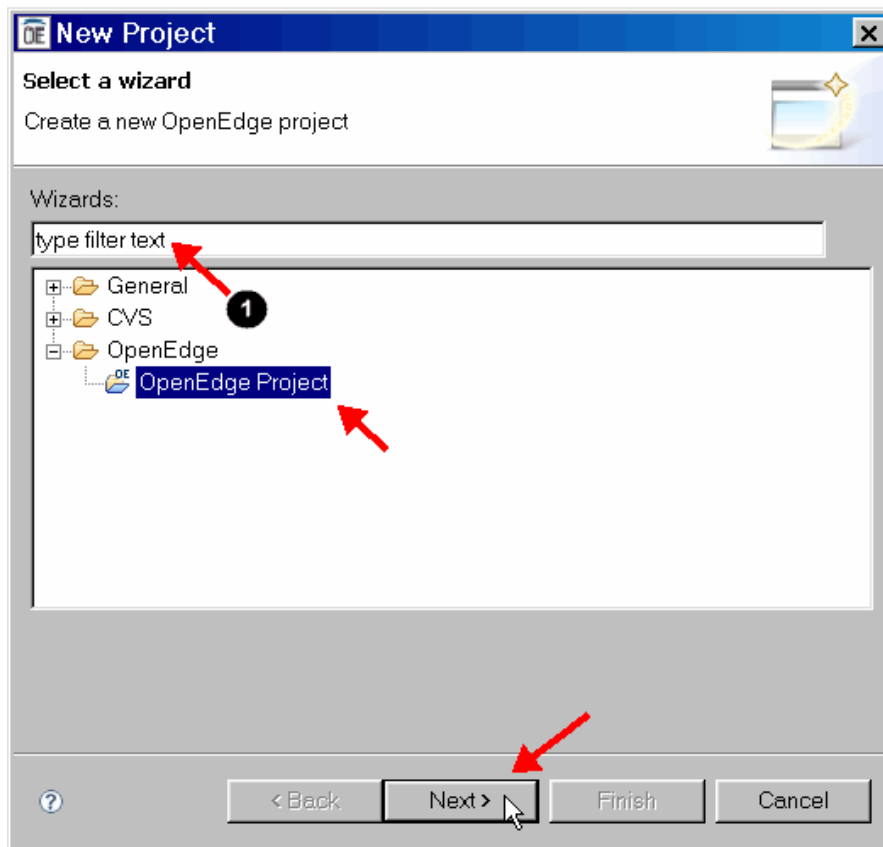
continued on next page


Lab 1 – Setting up the Environment, continued

Creating an OpenEdge Project

A project is a collection of associated application development files. These files can be source code files, data files, documentation files, images, etc. Folders can be used in a project to organize these files. Architect also keeps some metadata for your application files in a project. This section shows you how to create a project.


1. From the Architect menu bar, select **File→New→Project**.
2. In the New Project dialog, expand the OpenEdge node and select **OpenEdge Project**. Click **Next**.

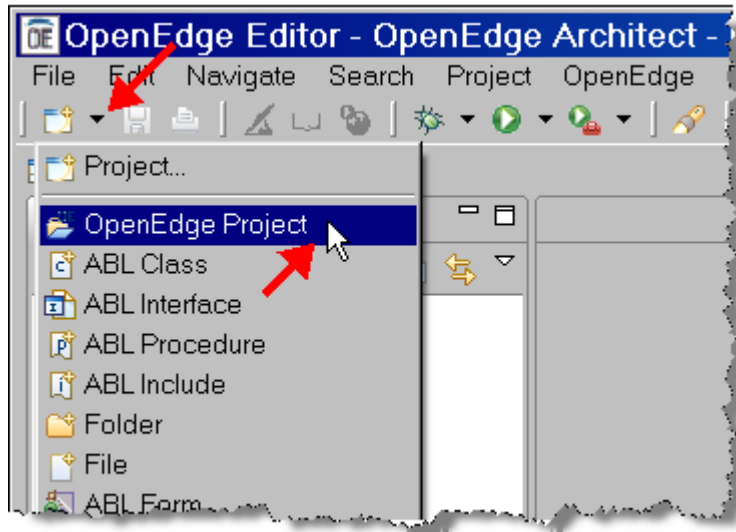


 **Tip:** A common feature in many Architect screens is the fill-in (❶) at the top of a selection which allows you to filter the selection list. For example, in the above screen a user could type an “O” in the Wizards fill-in to reduce the selection list to show only the OpenEdge folder of options. For this case, where there are only three entries, this feature is of limited use. However, if you are working with a lengthy list of options, the filter can be used to quickly locate the selection for which you are searching.

Lab 1 – Setting up the Environment, continued

Creating an OpenEdge Project, continued

 **Tip:** You can skip the New Project wizard screen and move directly to OpenEdge Project dialog by selecting **File→New→OpenEdge Project** or the **New** button and selecting the OpenEdge Project option as shown below:

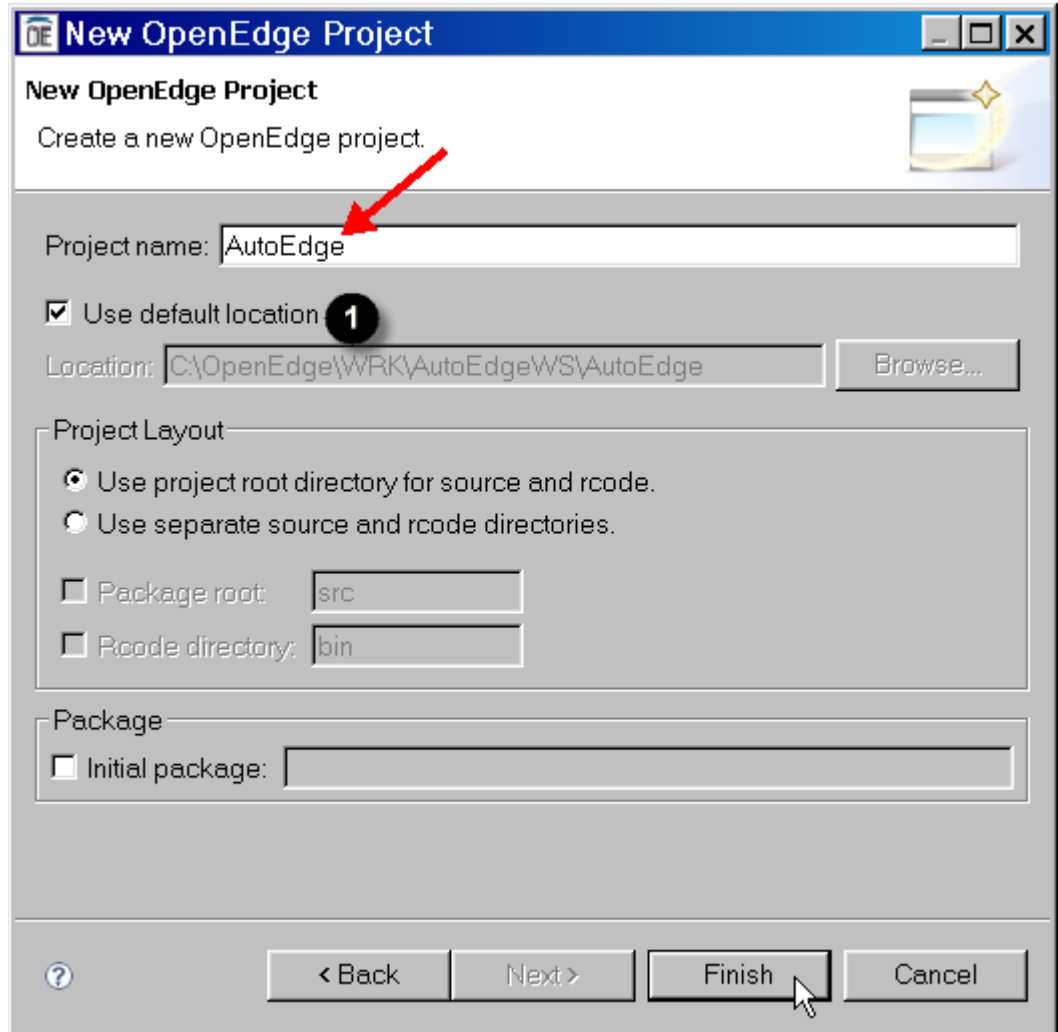



continued on next page

Lab 1 – Setting up the Environment, continued

Creating an OpenEdge Project, continued

3. In the New OpenEdge Project dialog, enter **AutoEdge** for the Project Name.



 **Tip:** The directory path for the new project is shown in the Location area(❶). This is the location where the project and its associated files will be stored physically. By default, Architect creates a subdirectory in the current workspace directory using the name of the project. You can choose to locate the project in a different directory by un-checking the **Use default location** box and then entering a directory in the Location fill-in.

continued on next page

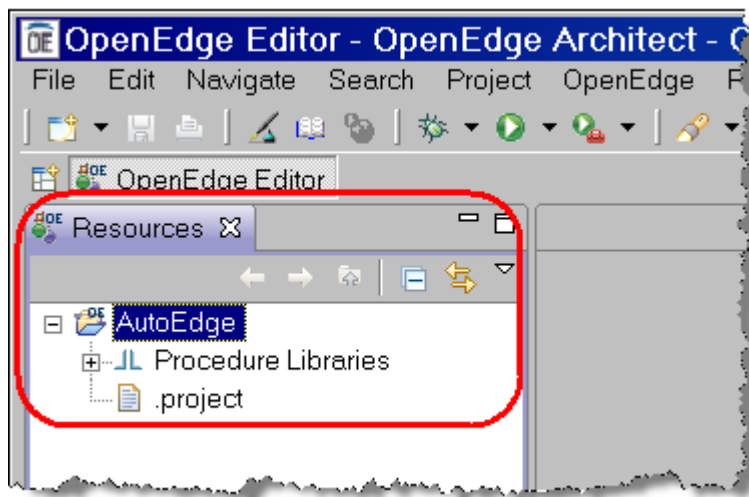
Lab 1 – Setting up the Environment, continued

Creating an OpenEdge Project, continued

4. Click **Finish**. The project will be created. The Powered by Progress splash screen will appear briefly showing that a Progress session is started for the Project.



Once the run-time has started, you will see the project listed in the Resources view:

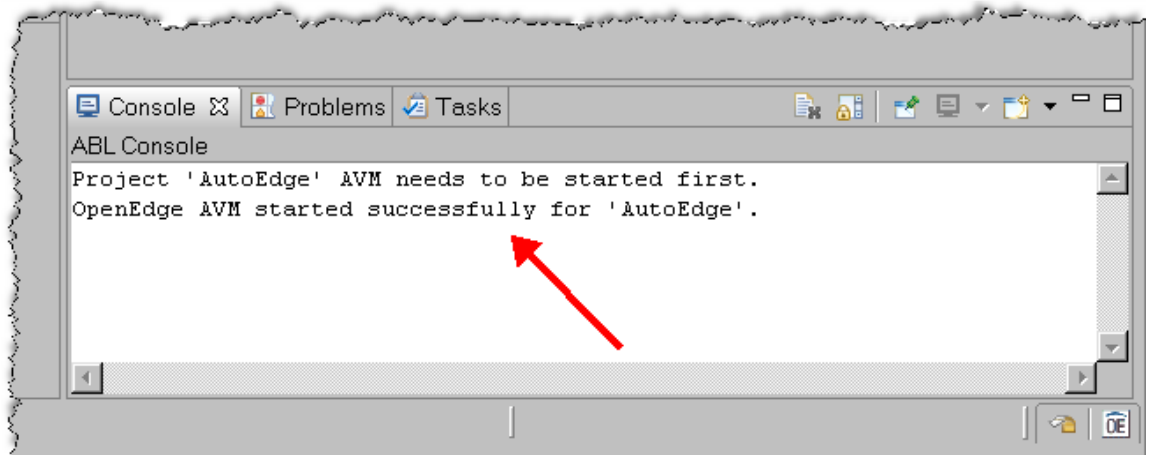


continued on next page

Lab 1 – Setting up the Environment, continued

Creating an OpenEdge Project, continued

You will also see a “started successfully” message in the Console box showing that the AVM was started for the project.



Note: The following are some important notes regarding projects:

- You may have a single source file or many source code and development files associated with a project.
- All projects have a type. By having a specific type, a project can be associated with the tools that are needed to manage the project. Other aspects/details can be tailored for the project, such as how to store metadata or what special configuration options are needed. Most of the time, you will use projects that have an OpenEdge type. However, you could have Sonic, Web, and other types of projects.
- Projects can be shared. They can also be integrated with Source Code Management (SCM) tools for team development. This makes logically grouping application files by module or functional unit very important.
- Projects make coding management easier because project's preferences and setup information can be shared among developers and system wide standards can be implemented.

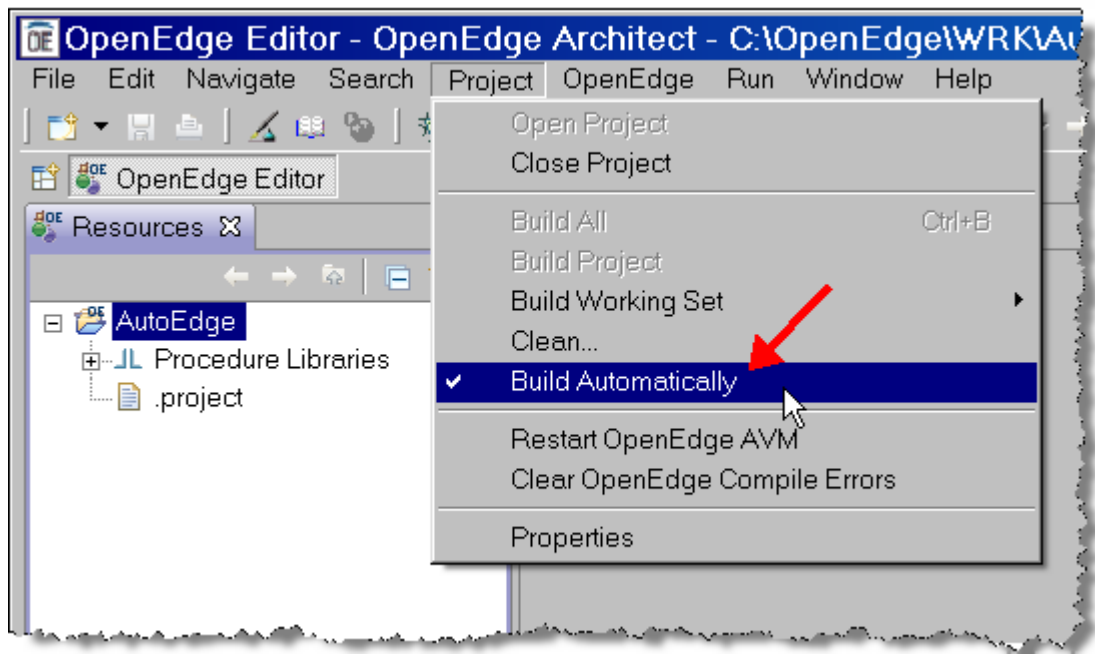
continued on next page

Lab 1 – Setting up the Environment, continued

Importing the AutoEdge application

The best way to move and/or share a project's resources is to use Architect's importing and exporting capabilities. This part of the lab shows you how to import existing application files into the project.

1. Before importing files, disable the Build Automatically option by selecting **Project**→**Build Automatically** from the Architect menu. This will uncheck the option. Disabling this option speeds the importing process, especially when there are a large number of files.

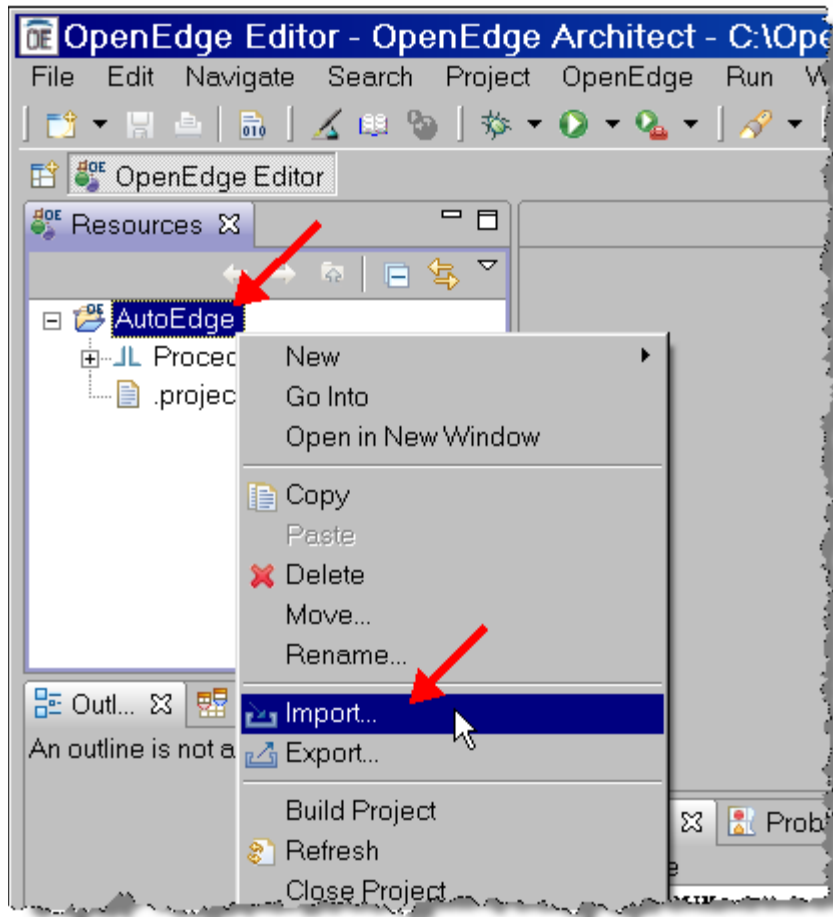


continued on next page

Lab 1 – Setting up the Environment, continued

Importing the AutoEdge application, continued

2. Right-click on the **AutoEdge** Project in the Resources view and select **Import**.

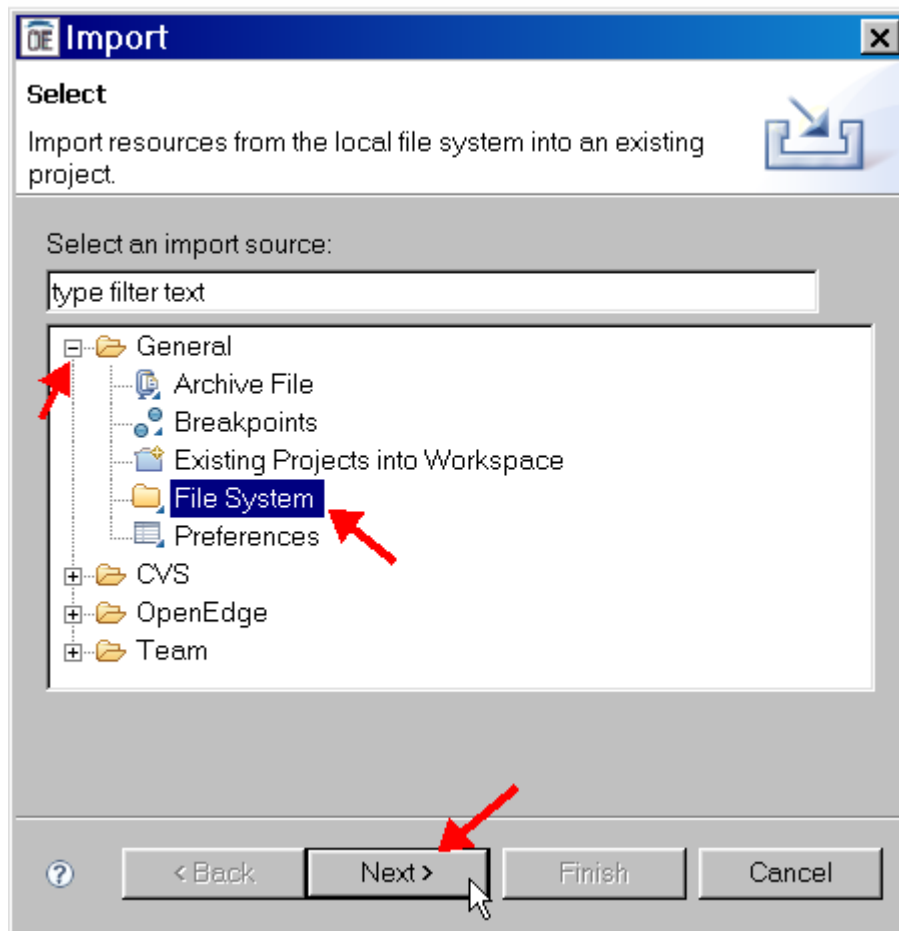


continued on next page

Lab 1 – Setting up the Environment, continued

Importing the AutoEdge application, continued

3. In the Import window, expand the **General** node and select **File System**. Click **Next**.

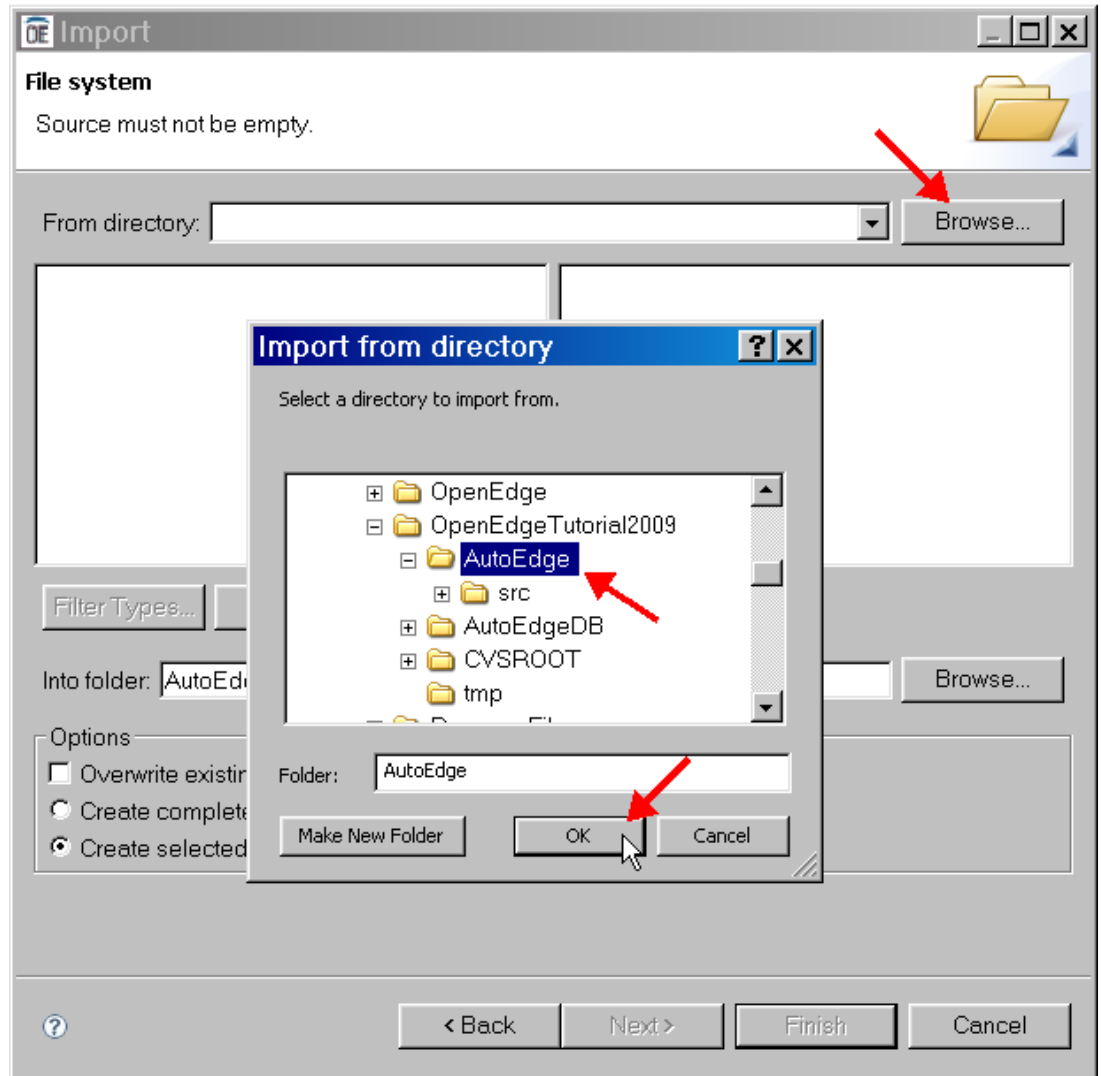


continued on next page

Lab 1 – Setting up the Environment, continued

Importing the AutoEdge application, continued

4. Click on the **Browse** button next to **From Directory** field and select the **C:\OpenEdgeTutorial2009\AutoEdge** directory. Click **OK**.

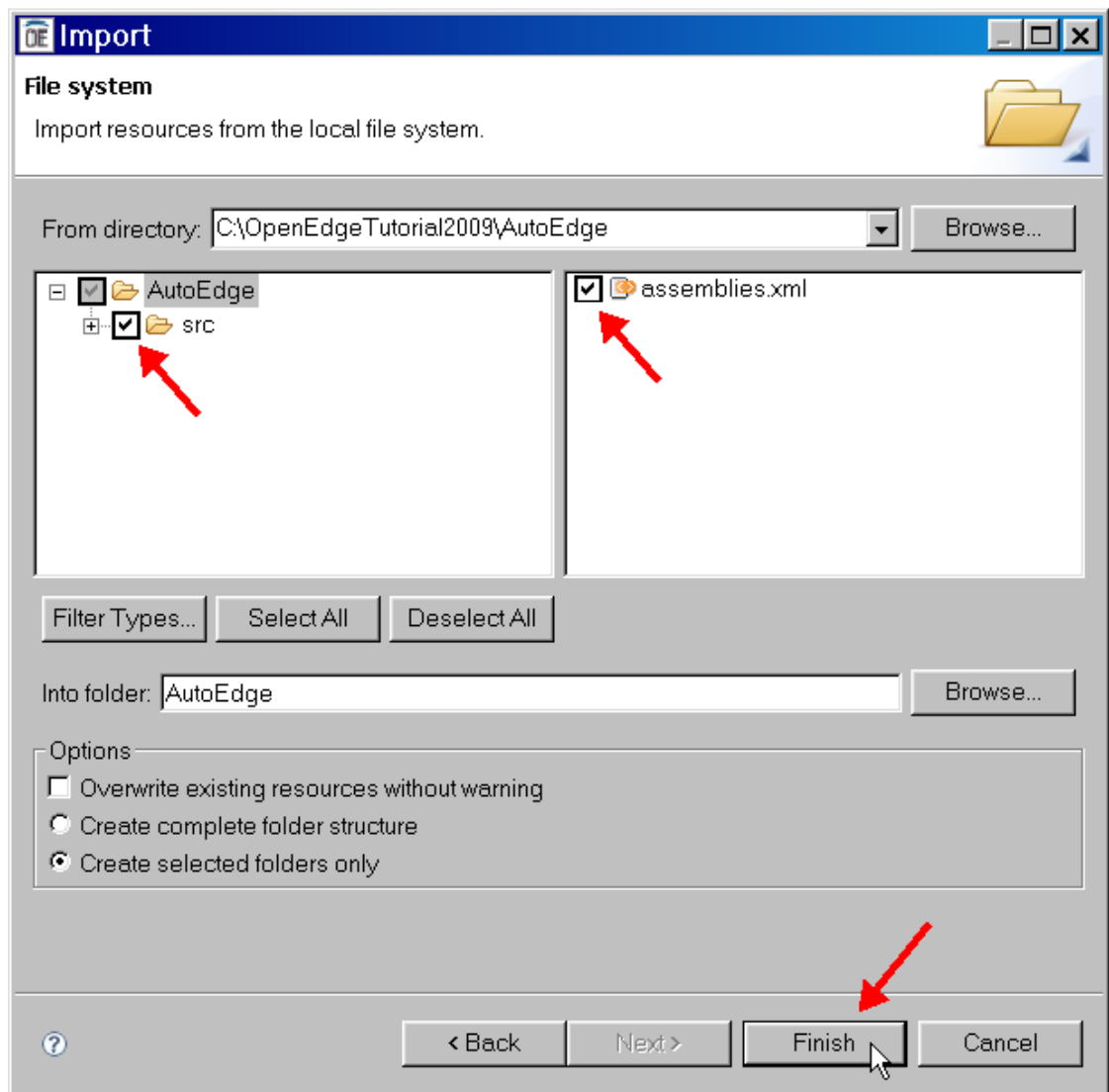


continued on next page

Lab 1 – Setting up the Environment, continued

Importing the AutoEdge application, continued

5. Expand the **AutoEdge** node and select the check box next to the **src** entry. Also check the box next to the **assemblies.xml** file entry in the right pane. This file includes reference pointers to some supporting files needed for the application. Click **Finish**.

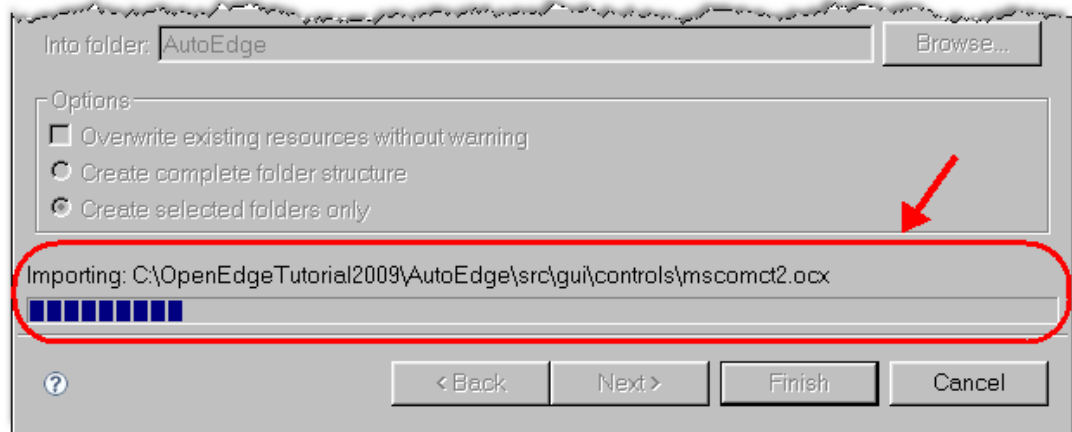


continued on next page

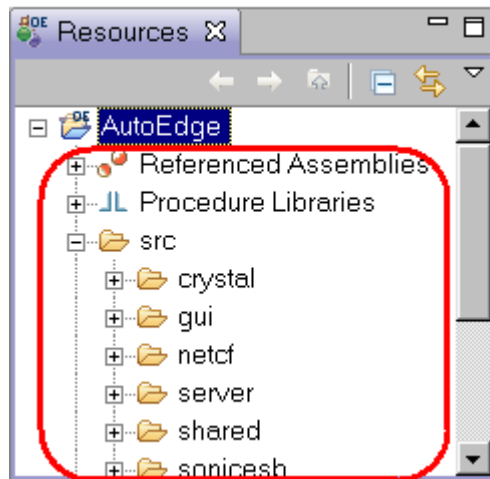
Lab 1 – Setting up the Environment, continued

Importing the AutoEdge application, continued

A bar at the bottom of the Import dialog displays the status and progress of the import.



When finished, the new resources will be shown in the Resources view. You can expand the folder nodes to view the imported resources.

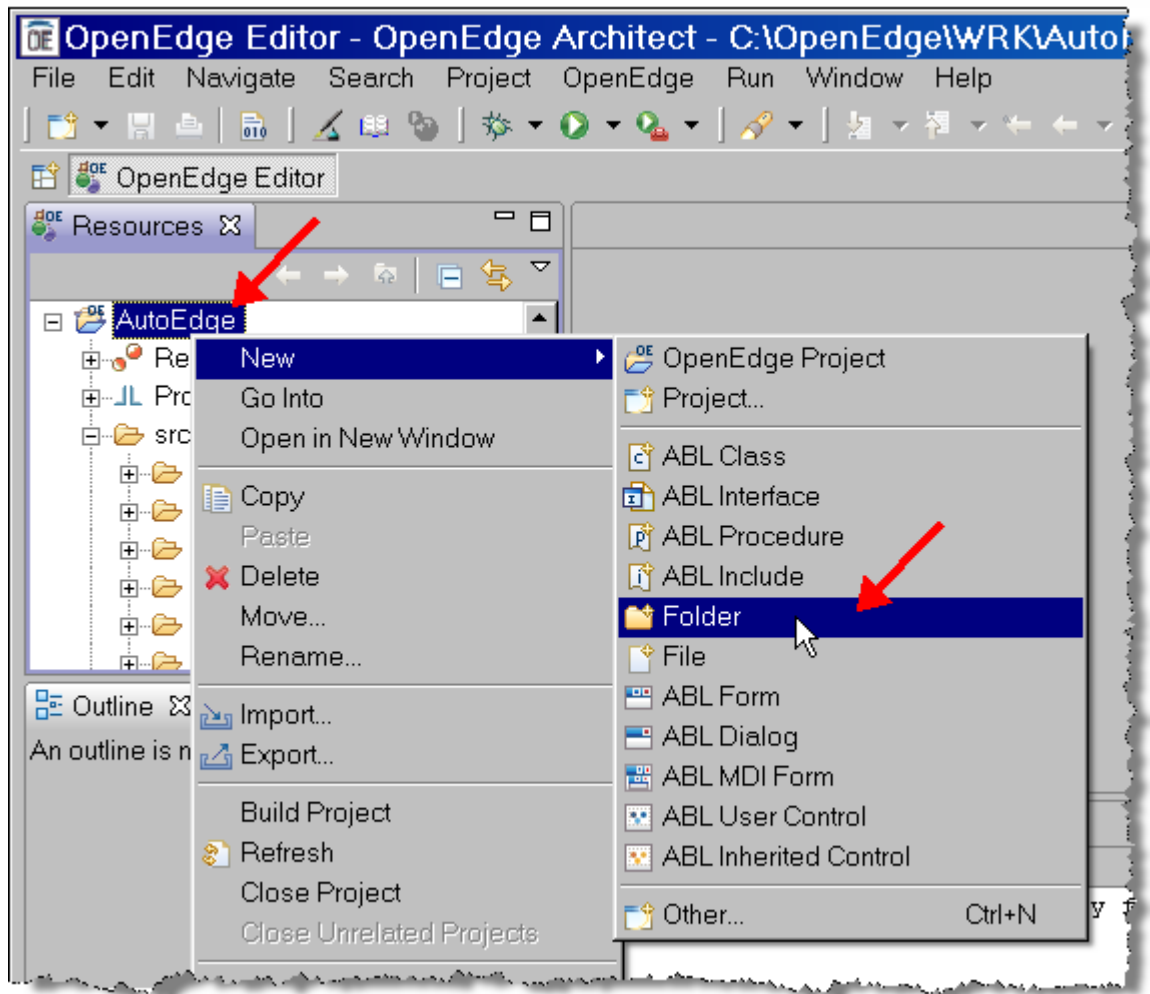


continued on next page

Lab 1 – Setting up the Environment, continued

Importing the AutoEdge application, continued

- Next you need to create a directory to store compiled source code. Right-click on the **AutoEdge** project in the Resources view and select **New→Folder**.

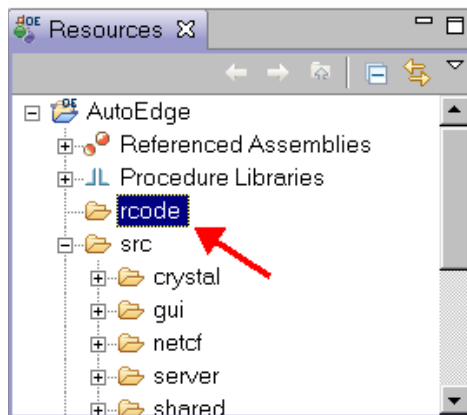
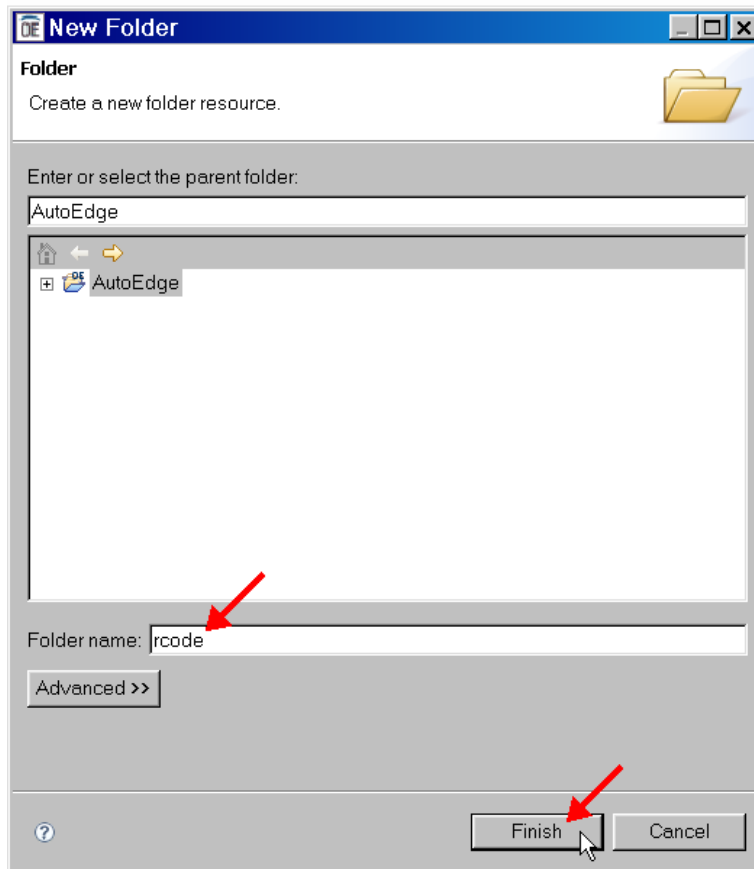


continued on next page

Lab 1 – Setting up the Environment, continued

Importing the AutoEdge application, continued

7. Specify the folder name as **rcode** and click the **Finish** button. In the next section, you will configure Architect to store compiled source code in this directory.



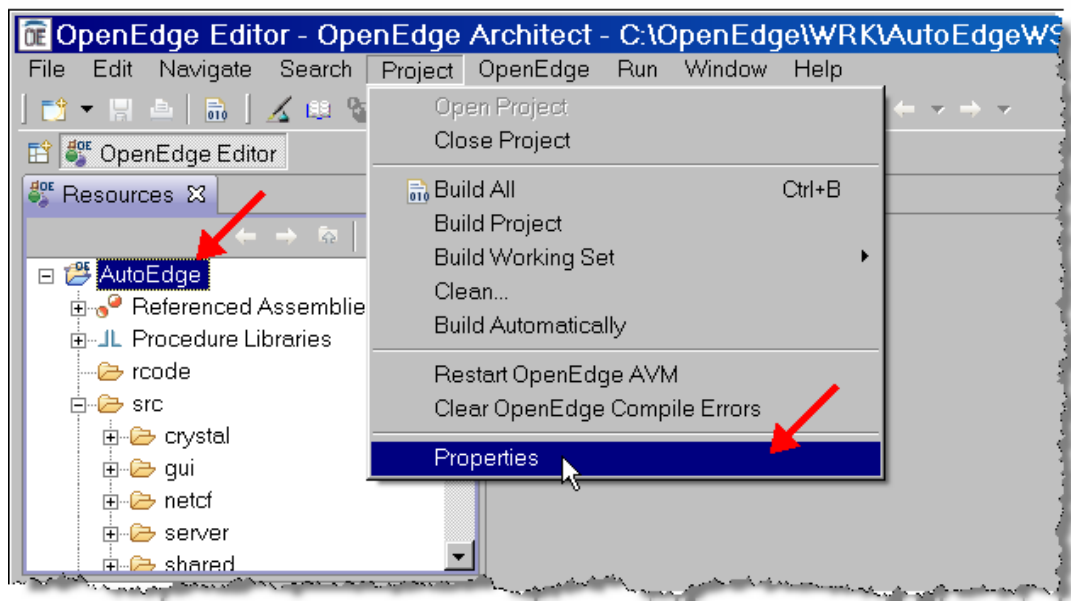
continued on next page

Lab 1 – Setting up the Environment, continued

Setting Project Properties

After importing files into your project, the next thing to do for a new project is to set its project properties. These properties control how the project interacts with the working environment, such as where to store compiled files. This section covers how to set some basic project properties.

1. Make sure that the AutoEdge project is selected in the Resources view. From the Workbench menu, select **Project**→**Properties**.

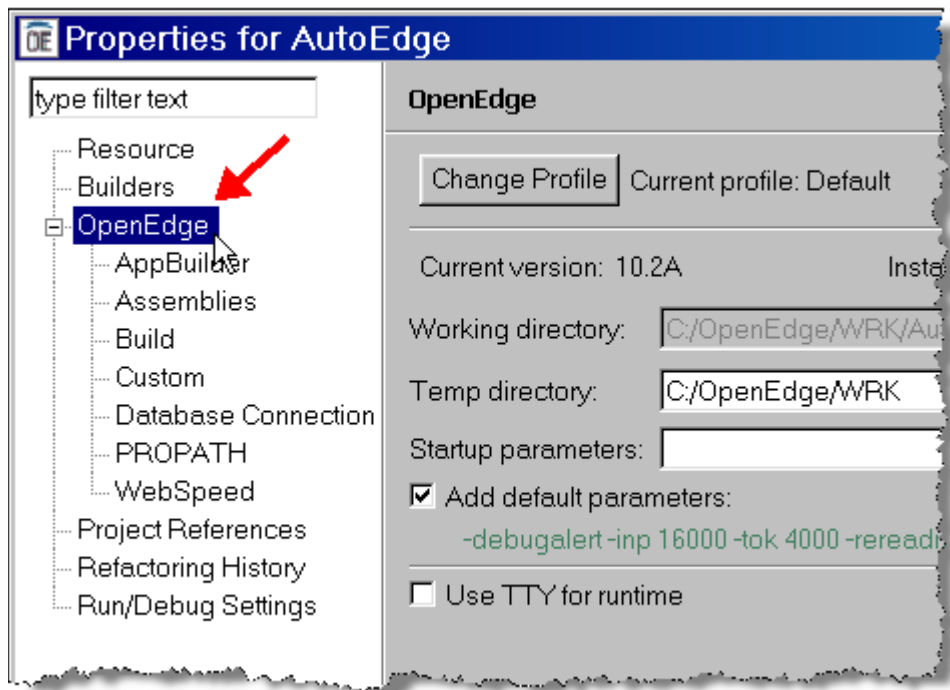


continued on next page


Lab 1 – Setting up the Environment, continued


Setting Project Properties, continued

2. The Properties for AutoEdge dialog appears. Since project properties are properties for a specific project; the project name appears in the title of the dialog. In the left pane of the Properties dialog, select **OpenEdge** to display the OpenEdge project settings for the selected project.



The OpenEdge project settings allow a number of OpenEdge options to be set including the working directory, the temporary directory, and startup parameters for the AVM.

 **Note:** Default startup parameters for projects are inherited from the workspace preferences. This saves time since most developers use a common set of startup options for each of their projects. These inherited parameters can be changed or removed by un-checking **Add default parameters**.

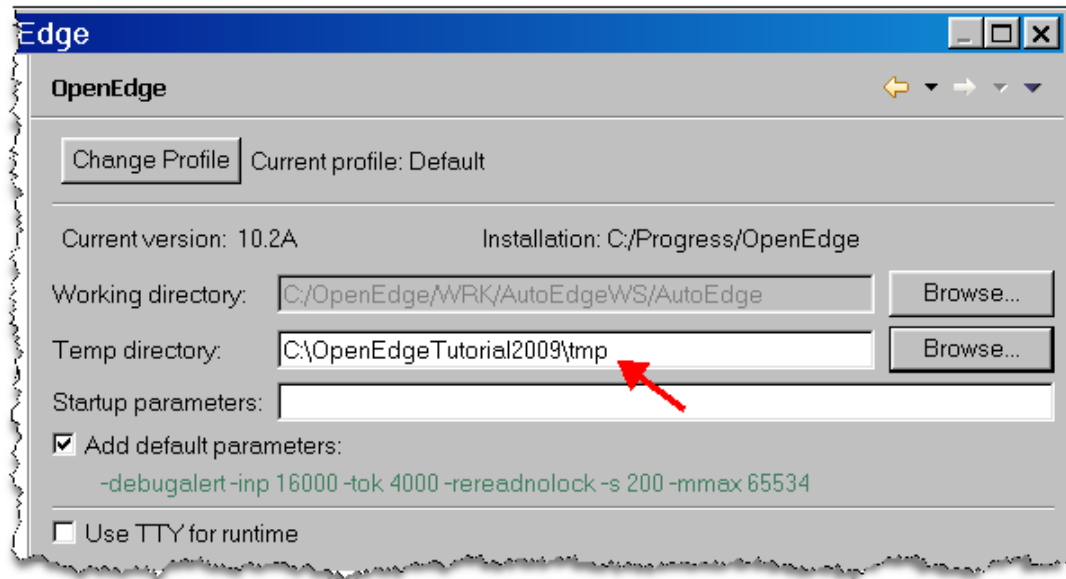
 **Caution:** While you could connect to a database using startup parameters on this page, it is recommended that you create database connections through Connection Profiles (discussed later) for a number of reasons. For instance, it is much more difficult to diagnose connection problems if they occur, using startup parameters. Plus, Connection Profiles can be shared among multiple projects.

continued on next page

Lab 1 – Setting up the Environment, continued

Setting Project Properties, continued

3. Set the temporary directory to **C:\OpenEdgeTutorial2009\tmp**. This helps to separate the temporary files from other files such as logs that are by default located in C:\OpenEdge\WRK. Note that this directory must already exist and was part of the pre-requisite setup for this lab.

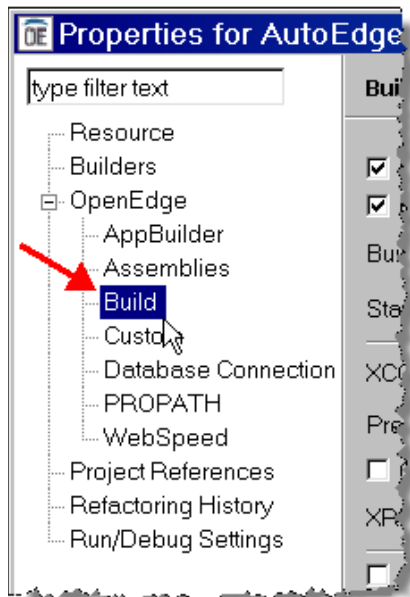


continued on next page

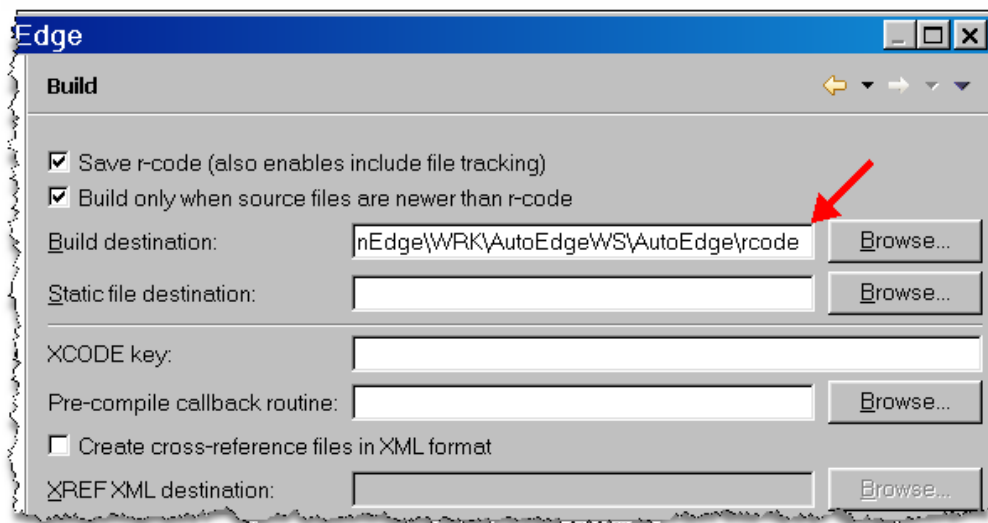
Lab 1 – Setting up the Environment, continued

Setting Project Properties, continued

4. If not already expanded, expand the **OpenEdge** node in the left pane and click on **Build**.



5. Specify the rcode directory that you created in a previous step (**C:\OpenEdge\WRK\AutoEdgeWS\AutoEdge\rcode**) for the Build destination field.

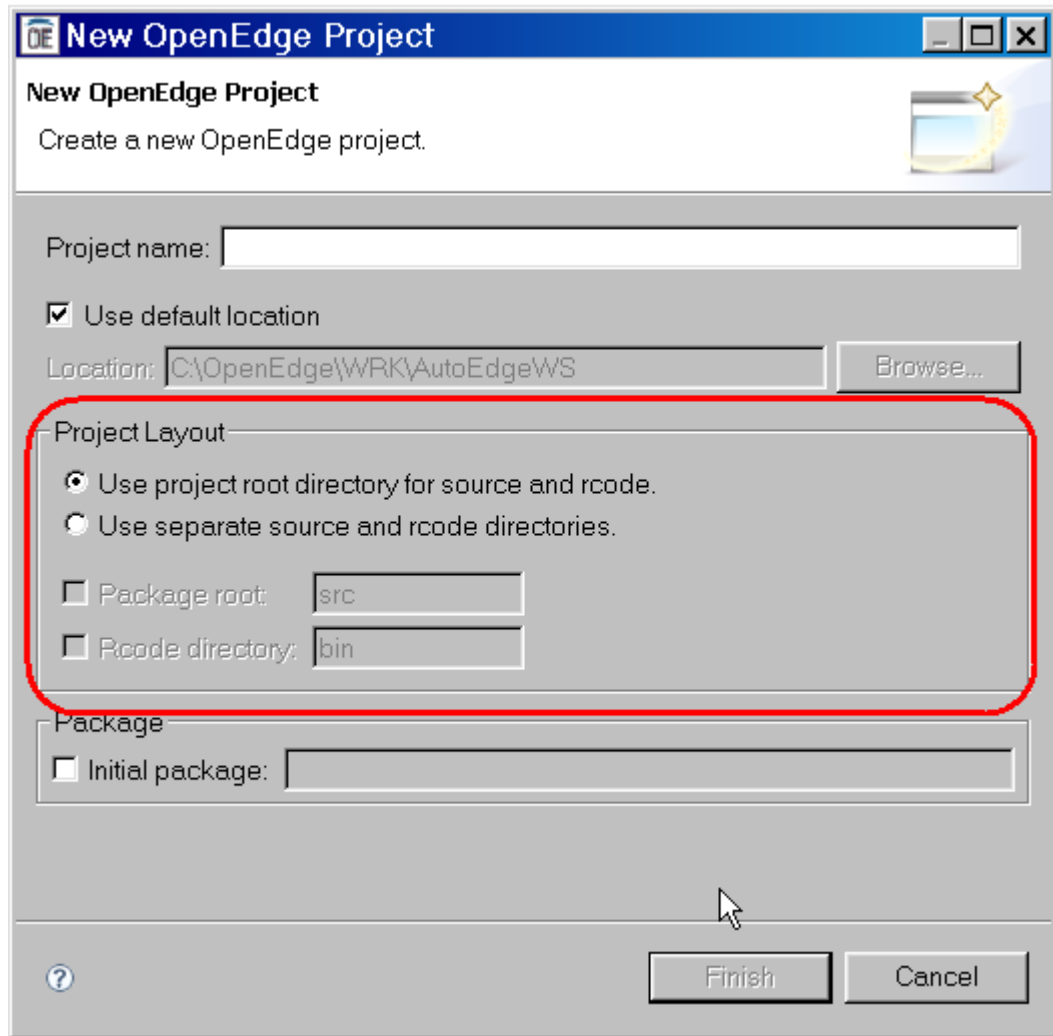


continued on next page

Lab 1 – Setting up the Environment, continued

Setting Project Properties, continued

 **Note:** You can also set up build directories while creating a project in the New Project dialog area shown below.

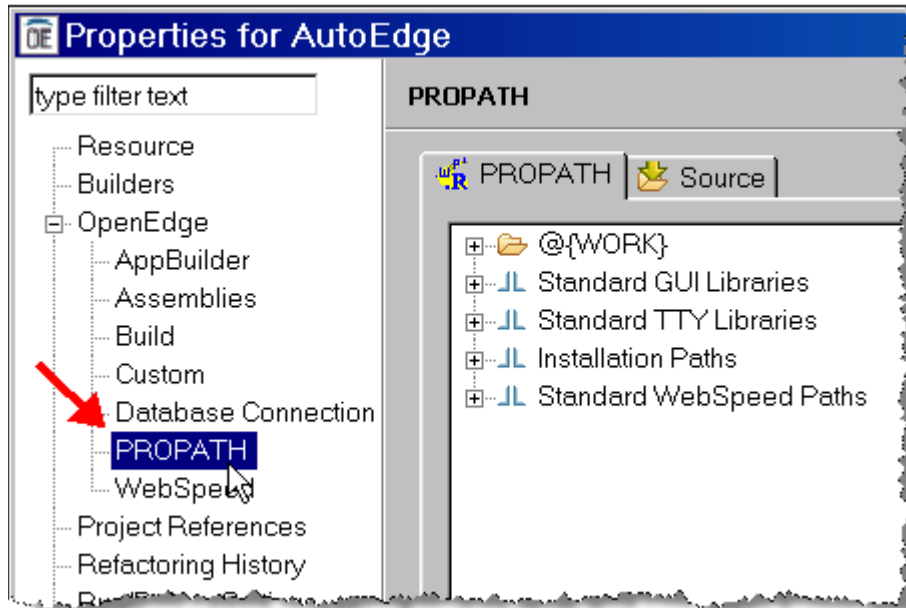


continued on next page

Lab 1 – Setting up the Environment, continued

Setting Project Properties, continued

6. Select the **PROPATH** option in the left pane to open the PROPATH options.

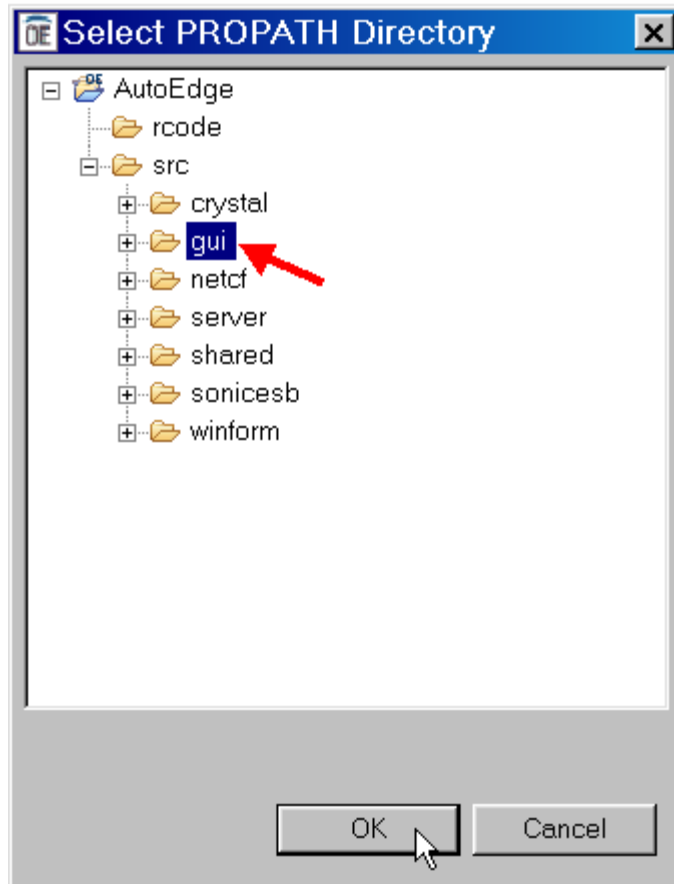


continued on next page

Lab 1 – Setting up the Environment, continued

Setting Project Properties, continued

7. Select the **Add Directory** button. Select the **src→gui** entry and then select **OK**.

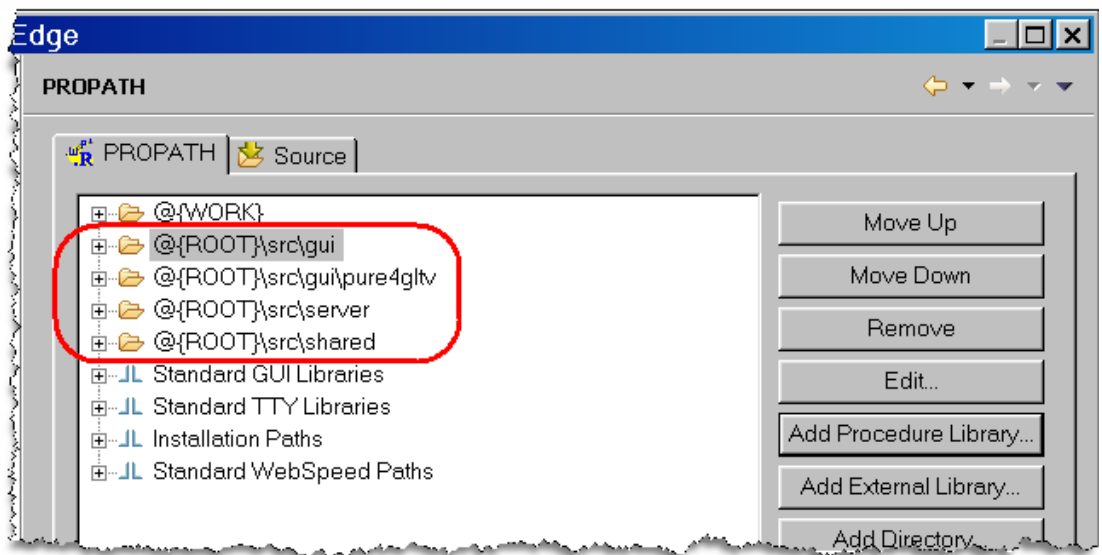


continued on next page

Lab 1 – Setting up the Environment, continued

Setting Project Properties, continued

8. Repeat this process for the following PROPATH entries:
 - src\gui\pure4glv
 - src\server
 - src\shared
9. Use the Move Up and Move Down buttons to locate the added directories in the following order and after the `@{WORK}` PROPATH entry.

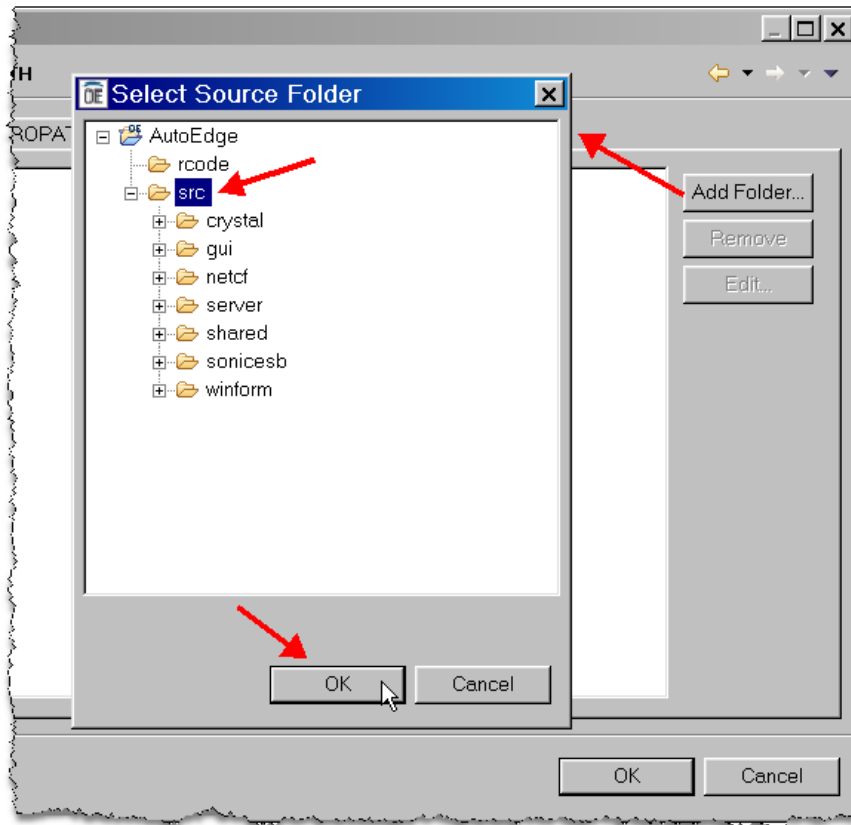


continued on next page

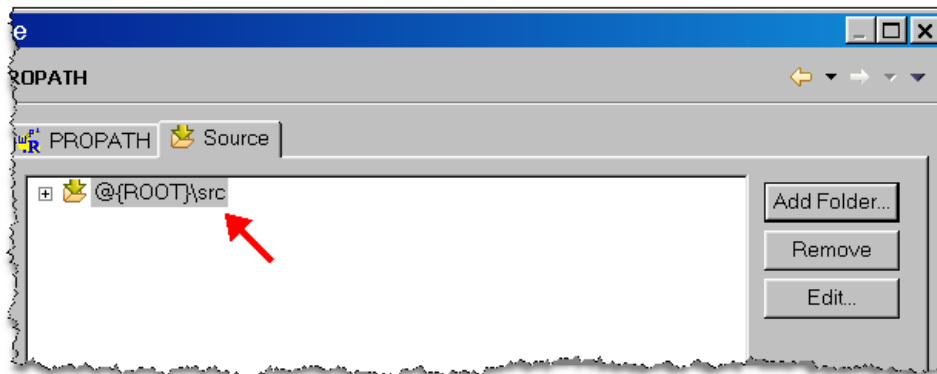
Lab 1 – Setting up the Environment, continued

Setting Project Properties, continued

10. Select the **Source** tab and click on the **Add Folder** button. Select the **src** folder entry and then click **OK**.



The directory will be added to the Source list:



11. Click **OK** to close the Properties for AutoEdge dialog.

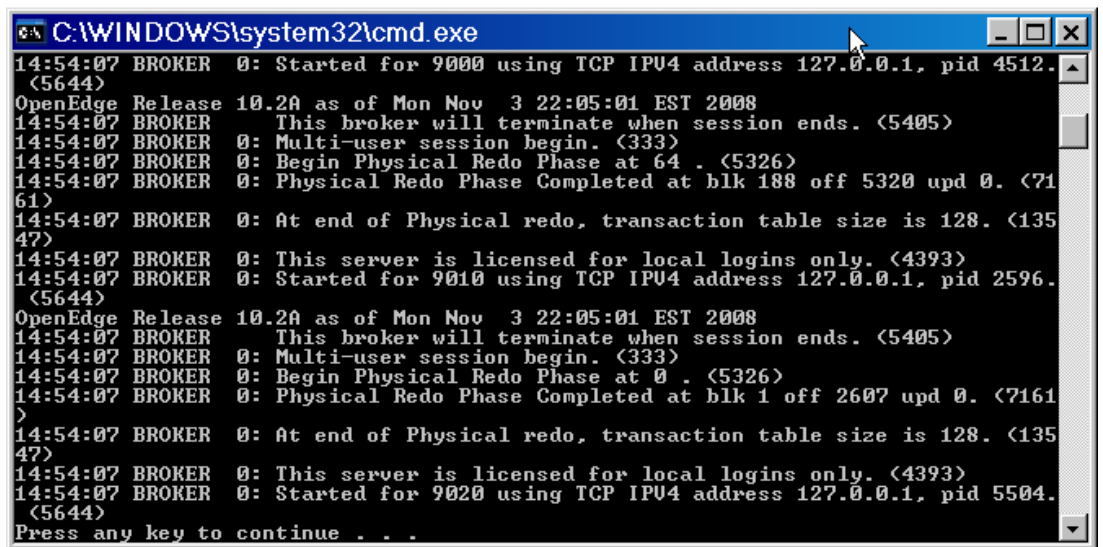
continued on next page

Lab 1 – Setting up the Environment, continued

Starting the Databases and setting up Database Connections

This section shows how to start the databases for the AutoEdge application and create database connection profiles for those databases. These connection profiles are created using Workspace Preferences and can be used by any projects in the workspace that need access to those databases.

1. From Windows Explorer, navigate to **C:\OpenEdgeTutorial2009\AutoEdgeDB** directory and double click on **startdbs.bat** file to start all the databases required. Press the spacebar once the script has finished executing.



```
C:\WINDOWS\system32\cmd.exe
14:54:07 BROKER 0: Started for 9000 using TCP IPV4 address 127.0.0.1, pid 4512.
(5644)
OpenEdge Release 10.2A as of Mon Nov 3 22:05:01 EST 2008
14:54:07 BROKER 0: This broker will terminate when session ends. (5405)
14:54:07 BROKER 0: Multi-user session begin. (333)
14:54:07 BROKER 0: Begin Physical Redo Phase at 64 . (5326)
14:54:07 BROKER 0: Physical Redo Phase Completed at blk 188 off 5320 upd 0. (71
61)
14:54:07 BROKER 0: At end of Physical redo, transaction table size is 128. (135
47)
14:54:07 BROKER 0: This server is licensed for local logins only. (4393)
14:54:07 BROKER 0: Started for 9010 using TCP IPV4 address 127.0.0.1, pid 2596.
(5644)
OpenEdge Release 10.2A as of Mon Nov 3 22:05:01 EST 2008
14:54:07 BROKER 0: This broker will terminate when session ends. (5405)
14:54:07 BROKER 0: Multi-user session begin. (333)
14:54:07 BROKER 0: Begin Physical Redo Phase at 0 . (5326)
14:54:07 BROKER 0: Physical Redo Phase Completed at blk 1 off 2607 upd 0. (7161
)
14:54:07 BROKER 0: At end of Physical redo, transaction table size is 128. (135
47)
14:54:07 BROKER 0: This server is licensed for local logins only. (4393)
14:54:07 BROKER 0: Started for 9020 using TCP IPV4 address 127.0.0.1, pid 5504.
(5644)
Press any key to continue . . .
```

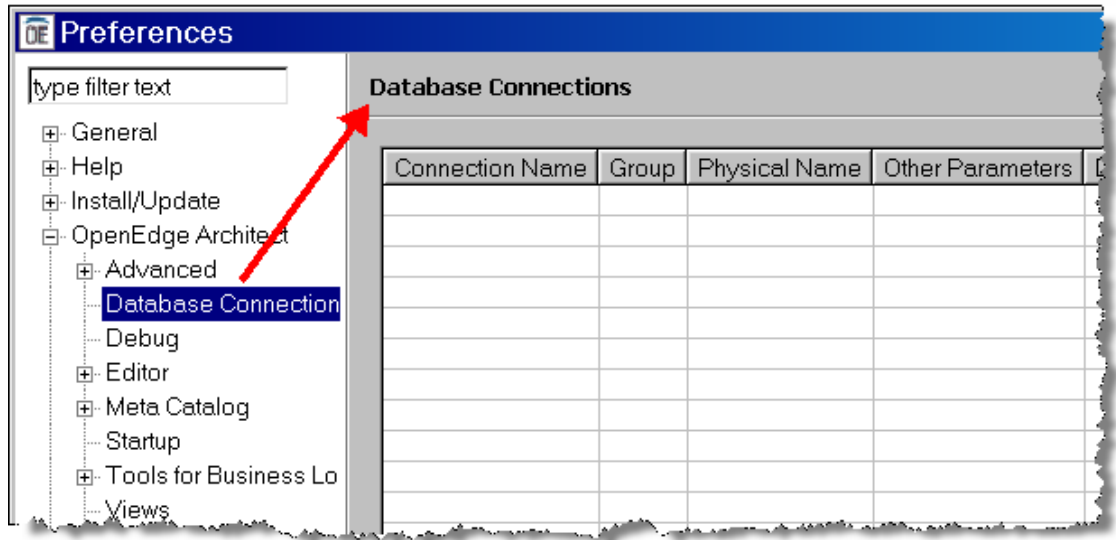
2. From Architect's menu bar select **Window→Preferences**. The Preference dialog for the Workbench is displayed. Preferences that are set here apply to the entire workspace.

continued on next page

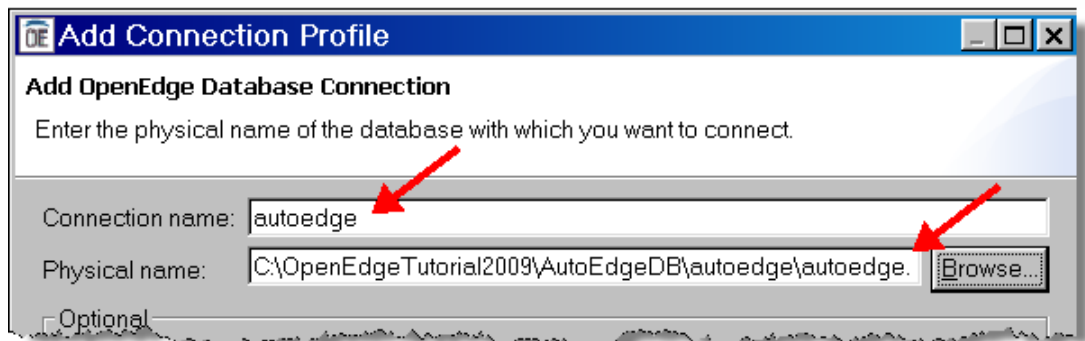
Lab 1 – Setting up the Environment, continued

Starting the Databases and setting up Database Connections, continued

3. In the left pane, select **OpenEdge Architect**→**Database Connections**. The Database Connections page is displayed on the right.



4. Click the **New** button to add a new Profile Connection. The Add Connection Profile wizard is displayed. This first page of the wizard allows connection information to be entered for an ABL connection to an OpenEdge RDBMS.
5. For the Connection name enter **autoedge**. For the physical name enter **C:\OpenEdgeTutorial2009\AutoEdgeDB\autoedge\autoedge.db**. The Browse button can be used to locate the database.



continued on next page

Lab 1 – Setting up the Environment, continued

Starting the Databases and setting up Database Connections, continued

6. For the other parameters enter:
 - Description: **Main AutoEdge Database**
 - Host name: **localhost**
 - Service/Port: **9000**

Connection name: autoedge

Physical name: C:\OpenEdgeTutorial2009\AutoEdgeDB\autoedge\autoedge.

Optional

Description: Main AutoEdge Database


Logical name:

Host name: localhost Service/Port: 9000

User ID: Password:

Aliases: Group:

Other parameters:

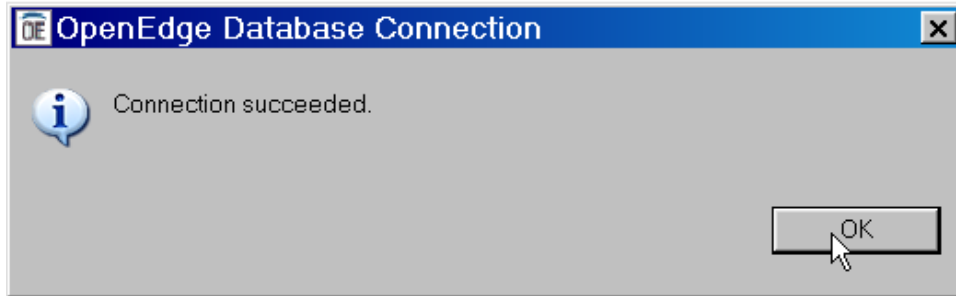
 **Note:** The Service/Port field will accept either a service name or a port number. If using a service name, make sure that it is defined in the services file. The services file is typically located in the following directory:
C:\WINDOWS\system32\drivers\etc\services.


continued on next page

Lab 1 – Setting up the Environment, continued


Starting the Databases and setting up Database Connections, continued

7. Click the **Test Connection** button. The OpenEdge ABL database connection is tested against the AVM associated with the AutoEdge project. If the connection can be made, a “Connection succeeded” message will be shown. Click **OK**.



 **Note:** If there are multiple projects in a Workspace, then a Select Project dialog is displayed and you must pick one of the workspace’s projects to test the connection against. If no project is defined for the Workspace, the Test Connection cannot be performed since the connection is tested against an AVM which must be started by an OpenEdge project.

8. If the connection fails, then confirm that that database is started and is using the correct port number. Try the test again.

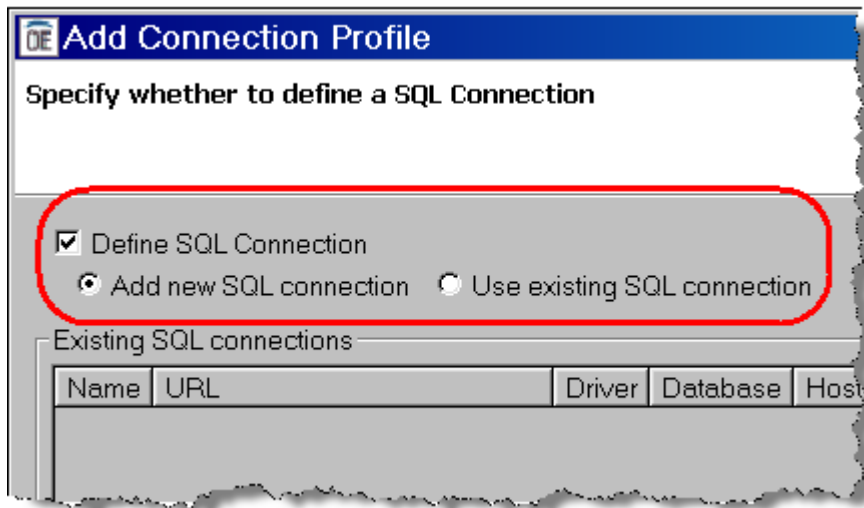
 **Note:** The test connection is made using a unique logical database name that is generated internally. So even if the selected project’s AVM is already connected to the database, the test connection can still be run.


continued on next page


Lab 1 – Setting up the Environment, continued

Starting the Databases and setting up Database Connections, continued

9. Click **Next** to go to the next page of the wizard. This page is for entering SQL Connection information for the database.



 **Caution:** If a SQL connection is not needed, then the **Define SQL Connection** can be unchecked and you can move to the next page of the wizard. However, it is recommended that a SQL connection be configured in most cases since it is used by a number of the tools in Architect. Without a SQL connection, some of the tools, such as the DB Navigator, will not connect to that database. Also, if there is a SQL connection already defined for the database, then you can select the **Use existing SQL connection** option and select the entry from the **Existing SQL connections** list.

 **Note:** As a general rule, ABL connections are used to run code against the database, such as through the OpenEdge Editor and OpenEdge Debugger tools. SQL connections are used to gather schema information and change database schema, such as through the DB Navigator tool.

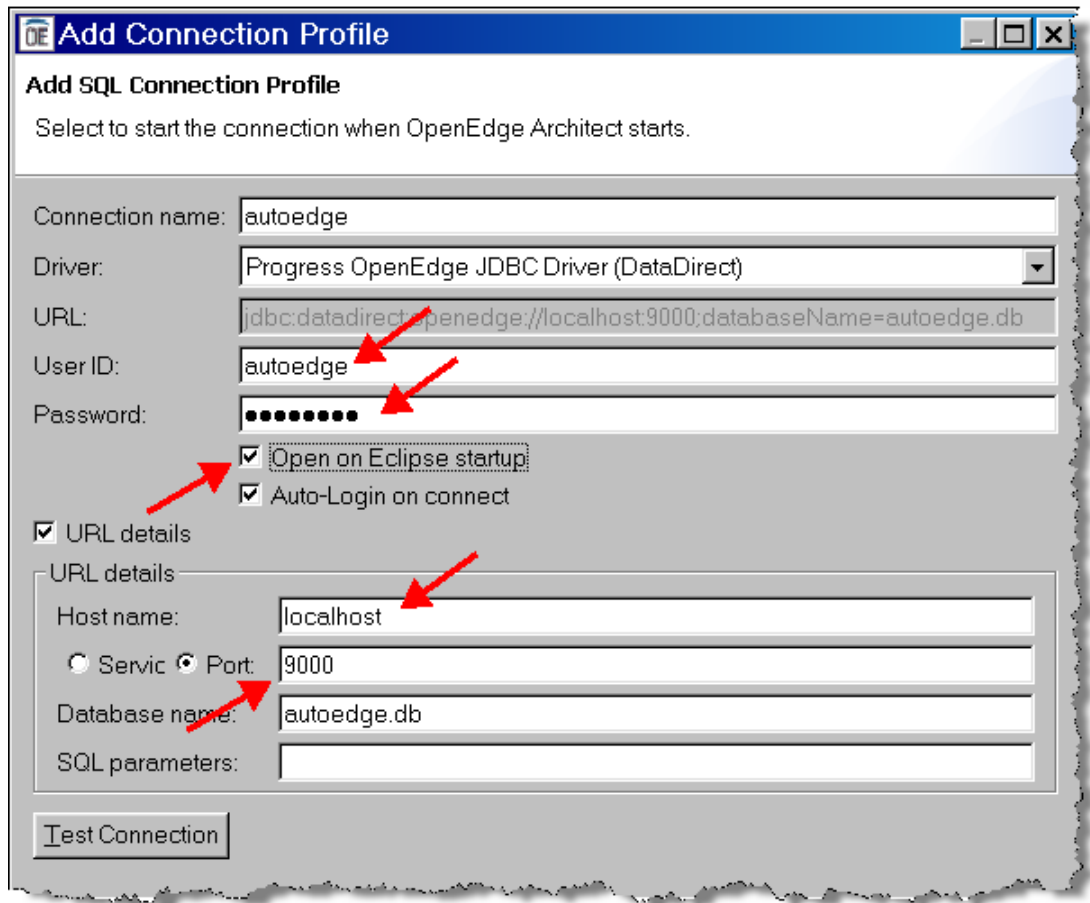
10. Confirm that the option to **Add new SQL connection** is selected and then click **Next**. The Add Connection Profile page for the SQL connection is displayed.

continued on next page

Lab 1 – Setting up the Environment, continued

Starting the Databases and setting up Database Connections, continued

11. A nice feature of the wizard is that all of the information that was entered on the wizard page for the ABL connection is automatically filled in for the SQL connection. Confirm that the Host name is **localhost** and that the Port is **9000**. Enter **autoedge** for the User ID and **password** for the Password. Finally, check the **Open on Eclipse startup** option. This will start the SQL connection each time the Workspace is opened.



12. Click the **Test Connection** button. The SQL connection does not require a project, so no project selection is displayed. If the connection succeeded, a success dialog will be displayed. Click **OK** to close the message.


continued on next page


Lab 1 – Setting up the Environment, continued

Starting the Databases and setting up Database Connections, continued

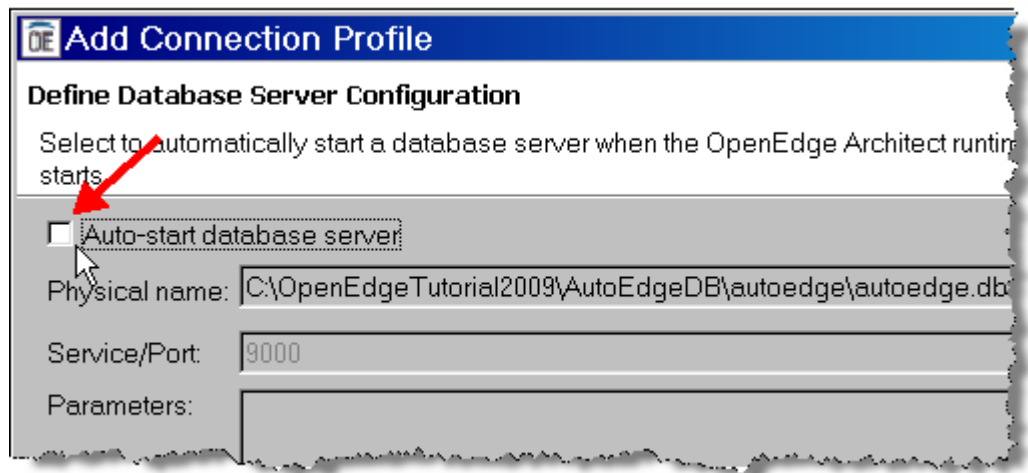
13. Click **Next**. The Define Database Server Configuration page is displayed.


This area allows you to set the database broker to be started whenever the connection profile is used, such as when opening a project that is tied to the connection profile. Since there already is a database broker started outside of Architect for this database (you did this by running the startdbs.bat script), this option is not needed.

 **Note:** For cases in which a database broker is already started and the options are configured to auto-start a database server, the options are ignored.

 **Tip:** For cases where you decide to have the database broker started using this option, the option on the SQL connection wizard page to **Open on Eclipse startup** should be unchecked for this database, since a broker will not be running when Architect initially starts up.

14. Uncheck **Auto-start database server**.



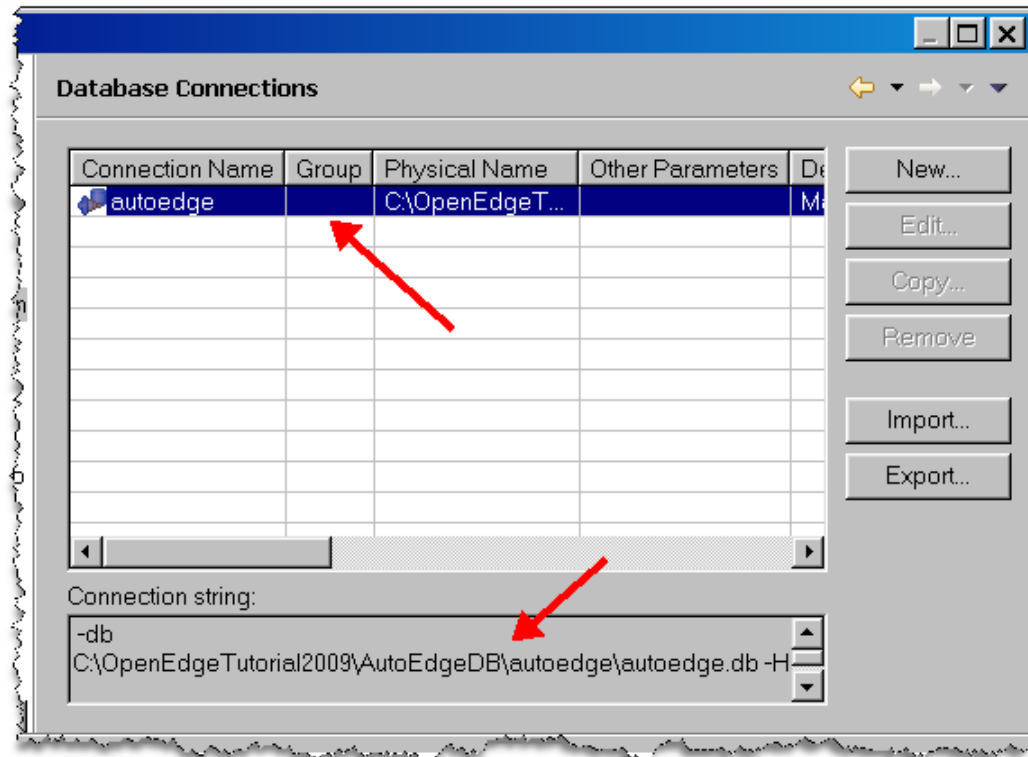
 **Note:** As noted above, the database has already been started outside Architect and so you don't need to start it from here. If you had not started the Database before, you could leave **Auto-start database server** check box checked and the database would be started when you assign the connection to a project.

continued on next page

Lab 1 – Setting up the Environment, continued

Starting the Databases and setting up Database Connections, continued

15. Click **Finish** to complete the wizard. The newly created database connection profile is listed in the Database Connections grid. The connection information also is shown in the Connection string area.



continued on next page

Lab 1 – Setting up the Environment, continued

Starting the Databases and setting up Database Connections, continued

16. Follow the same procedure to create connection profiles for the **oerpcore** and **temp-db** databases.

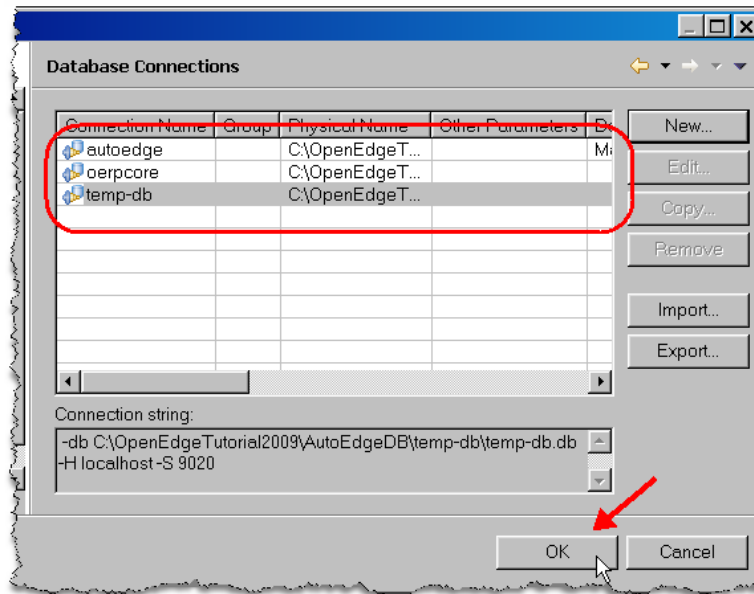
Oerpcore:

- Connection name: **oerpcore**
- Physical name: **C:\OpenEdgeTutorial2009\AutoEdgeDB \oerpcore\oerpcore.db**
- Hostname: **localhost**
- Port: **9010**
- User ID (SQL only): **autoedge**
- Password (SQL only): **password**

Temp-db:

- Connection name: **temp-db**
- Physical name: **C:\OpenEdgeTutorial2009\AutoEdgeDB \temp-db\temp-db.db**
- Hostname: **localhost**
- Port: **9020**
- User ID (SQL only): **autoedge**
- Password (SQL only): **password**

17. Check that all of the new entries are listed in the Database Connections grid. Click **OK** to close the Preferences dialog.



continued on next page

Lab 1 – Setting up the Environment, continued

Associating database connections with projects

The previous section of this lab showed how to create a database connection profile. Connection profiles are very useful since they allow both ABL and SQL database connections to be defined once for the entire workspace. However, these connection profiles are only definitions. A connection profile must be associated with a project before the information can be used to actually make a connection to the database. This section shows how to make these associations.



Caution: Add only the database connections to a project that are needed. Always keep in mind that system resources are limited. If you have a large number of projects that are always open and those projects each connect to a number of databases, then it may take a lot of time to start OpenEdge Architect. Also consider that there are defaults and limits on the number of allowed database connections, depending on the OpenEdge products that are installed and how they are configured.

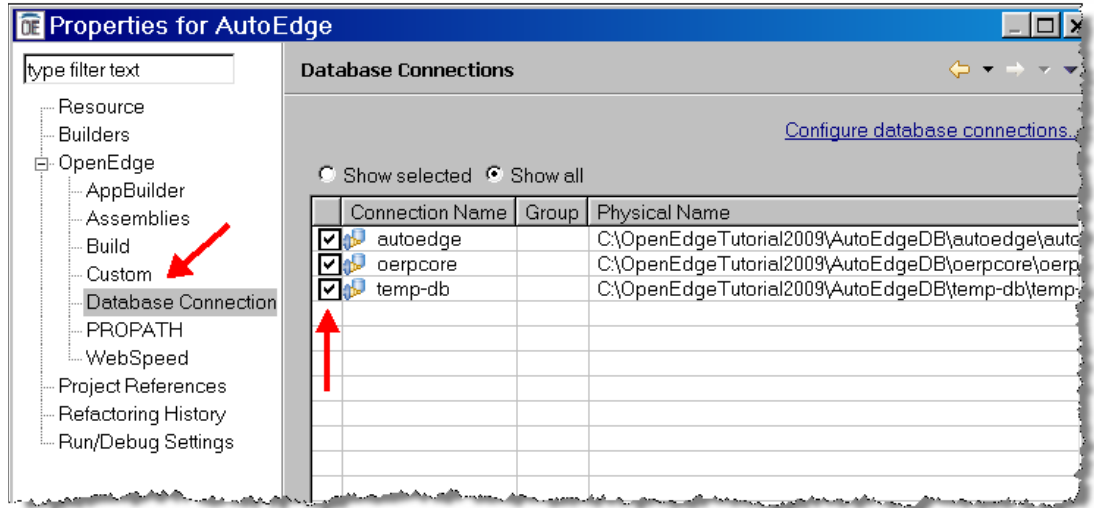
1. Select the **AutoEdge** project in the Resources view.
2. Right-click on **AutoEdge** and select **Properties**. The project properties dialog opens.
3. Select **OpenEdge→Database Connections** in the left pane. The databases that are defined for the Workspace are displayed in the Database Connections grid.

continued on next page

Lab 1 – Setting up the Environment, continued

Associating database connections with projects, continued

4. Check the boxes in front of **autoedge**, **oerpcore**, and **temp-db**.

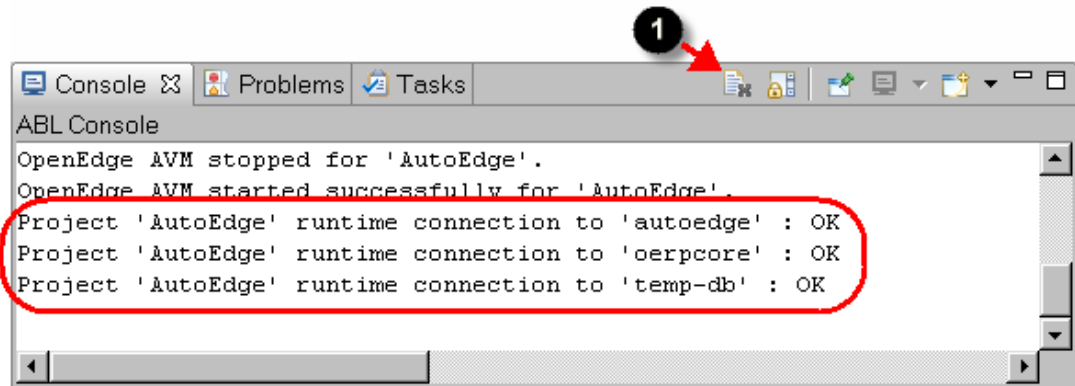



continued on next page


Lab 1 – Setting up the Environment, continued

Associating database connections with projects, continued

5. Click **OK**. The Properties dialog closes and the AVM associated with project is restarted with the new property changes. A splash screen displays briefly as the AVM is restarted. The Console view will show messages that the runtime was started and the status of the connections to the autoedge, oerpcore and temp-db databases that were associated to the project in the project's properties.




 **Note:** You may receive an alert box stating that the AVM is busy. Click **Yes** to restart the session.

 **Tip:** It can be useful at times to clear the messages in the Console view in order to focus just on the newest messages. Click the Clear Console button (❶) on the Console view toolbar to clear all of the messages.

Compile and Run the Application

This section shows how to compile and run the AutoEdge project from OpenEdge Architect and demonstrates the Run configuration.

1. In the Architect menu, select **Project→Build Automatically** selection to restart the automatic building that was suspended before importing the application files. In the console view, you will see the compile status for all of the project's files.

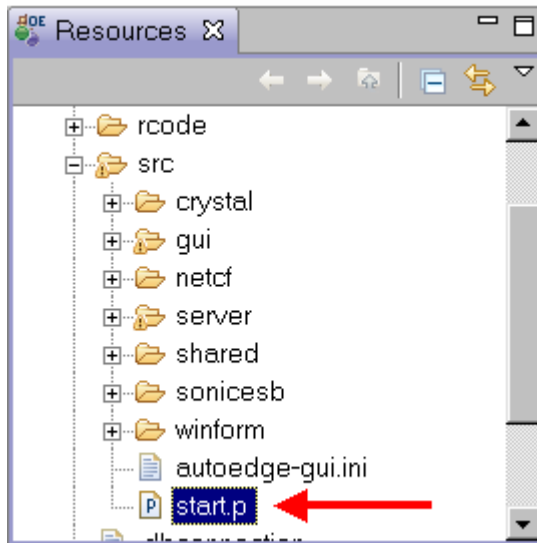
 **Note:** The compiled source code will be placed in the **src** directory that you created earlier and set up as the build directory. Subdirectories will be created for you if they do not already exist.

continued on next page

Lab 1 – Setting up the Environment, continued

Compile and Run the Application, continued

2. Expand **AutoEdge**→**src** in the Resources view and double-click on **start.p** file to open the file in the OpenEdge Editor

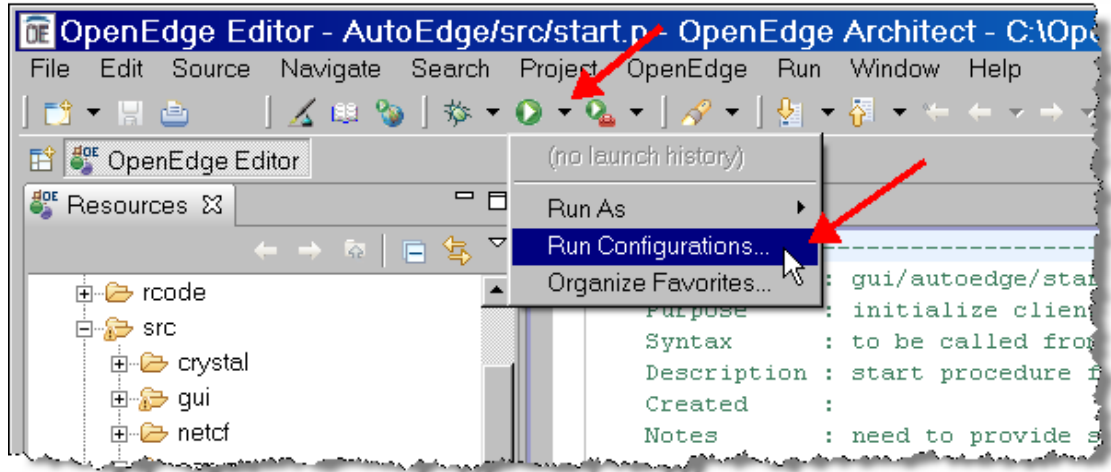


continued on next page


Lab 1 – Setting up the Environment, continued

Compile and Run the Application, continued

3. Select **Run Configurations** from the Run button's drop-down menu.



A Run dialog is shown which allows a developer to configure all of the information that is needed to run an application independent of any project setting. This includes information such as startup parameters, database connections, PROPATH, and the OpenEdge version to be used.

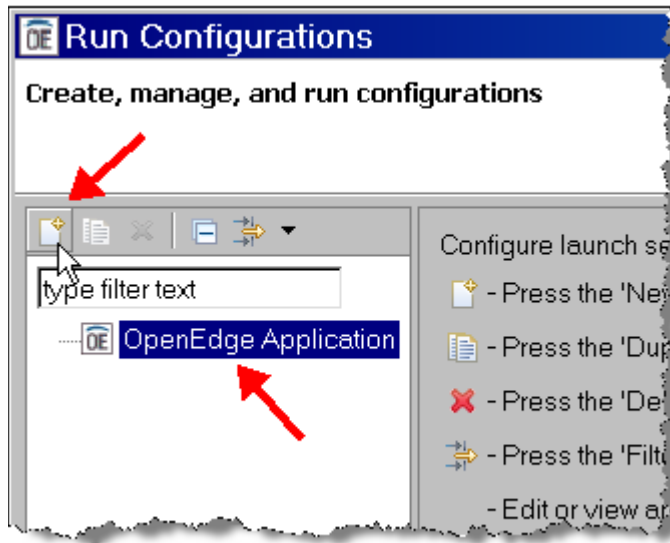
 **Tip:** You might use Run Configurations as a way to define and save environments for testing an OpenEdge application. You can have a number of Run Configurations defined for different types of testing for the same application code.

continued on next page

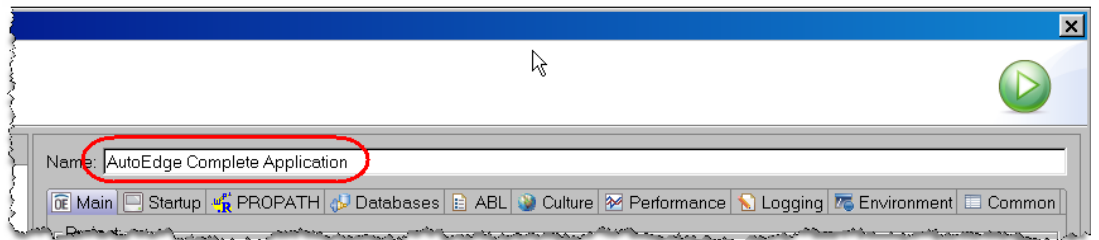
Lab 1 – Setting up the Environment, continued

Compile and Run the Application, continued

4. Click on the **OpenEdge Application** node and select the **New** button to create a new configuration.



5. Enter **AutoEdge Complete Application** for the Name. Click **Apply**.

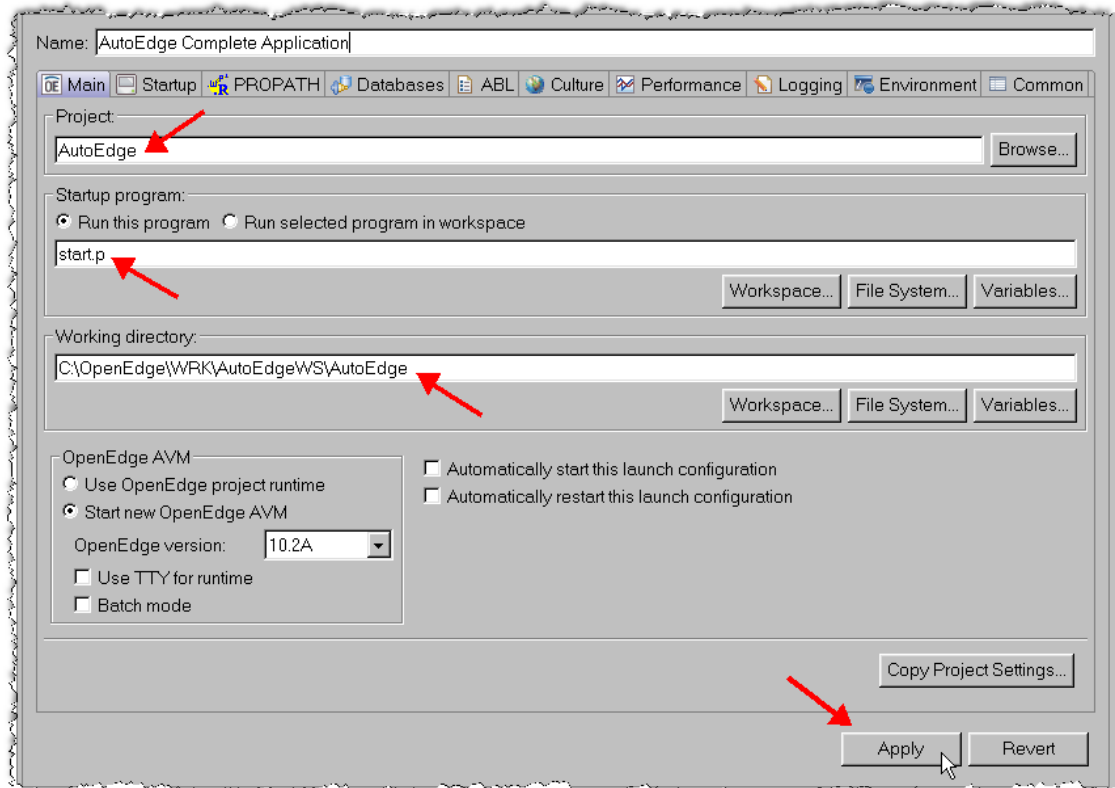



continued on next page

Lab 1 – Setting up the Environment, continued

Compile and Run the Application, continued

6. In **Main** tab, add or verify the following details:
 - Project: **AutoEdge**
 - Run this program: **start.p**
 - Working directory: **C:\OpenEdge\WRK\AutoEdgeWS\AutoEdge**



 **Note:** This start.p program will be used to start the application every time this configuration is run.

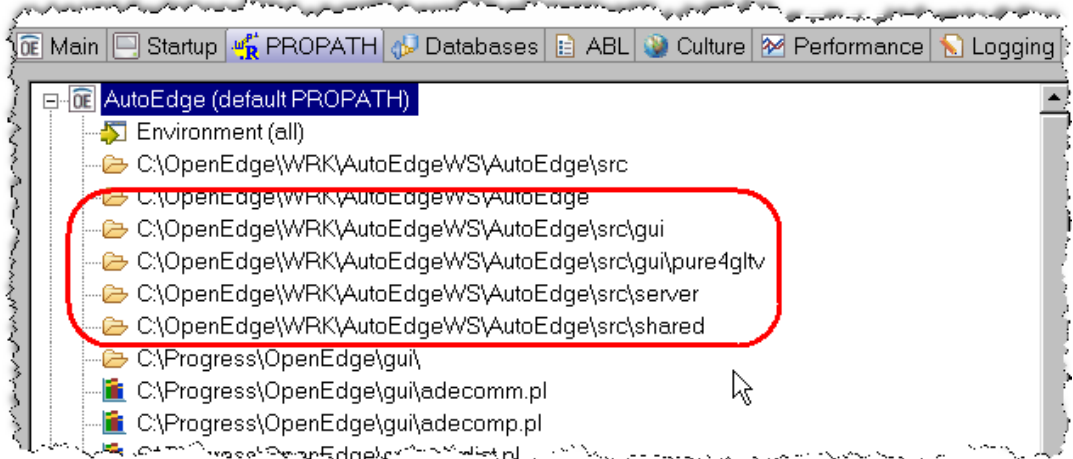
7. Select the **Apply** button.

continued on next page

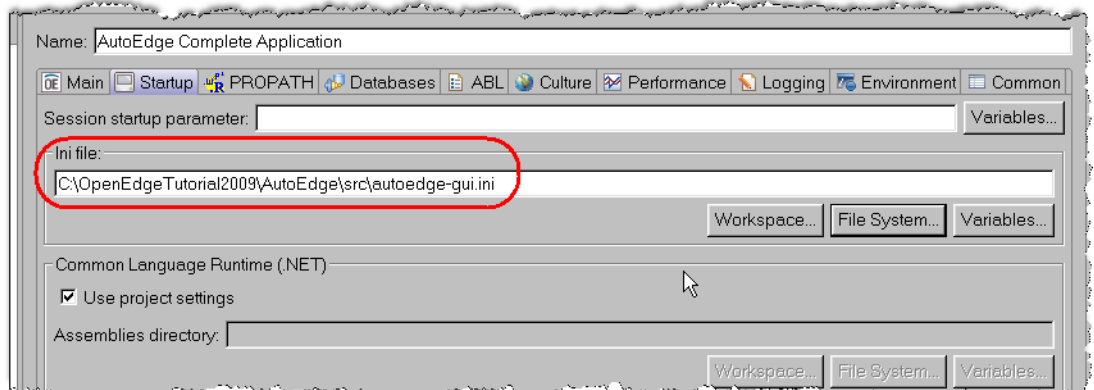
Lab 1 – Setting up the Environment, continued

Compile and Run the Application, continued

8. Select the **PROPATH** tab. Expand the **AutoEdge** node. Verify that all the PROPATH entries corresponding to the AutoEdge Project have been pulled over from the project properties settings.



9. Select the **Startup** tab. In the Ini file area, enter **C:\OpenEdgeTutorial2009\AutoEdge\src\autoedge-gui.ini** in the field. This startup file contains setup information for the application's GUI screens.

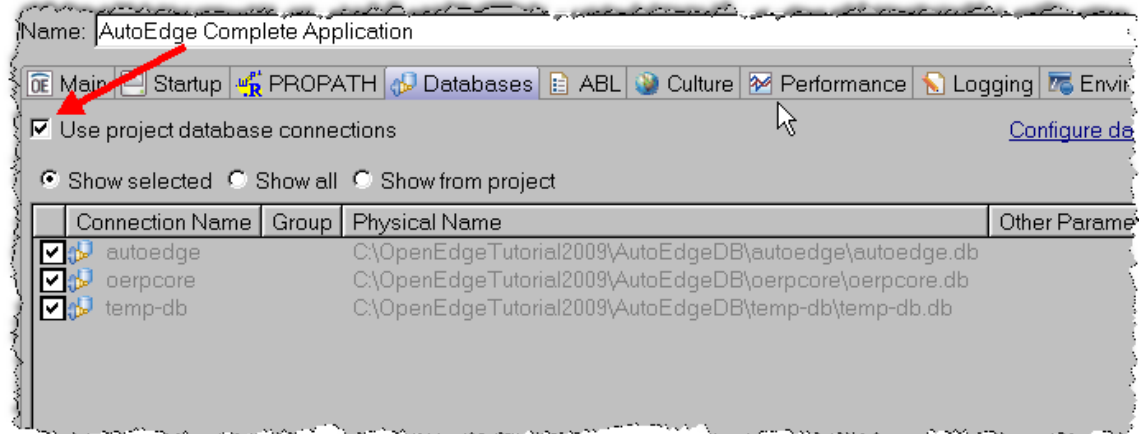


continued on next page

Lab 1 – Setting up the Environment, continued

Compile and Run the Application, continued

10. Select the **Databases** tab and make sure the **Use project database connections** option is checked. The individual database will be automatically checked for you.



11. Click the **Apply** button.
12. Click the **Run** button.

continued on next page

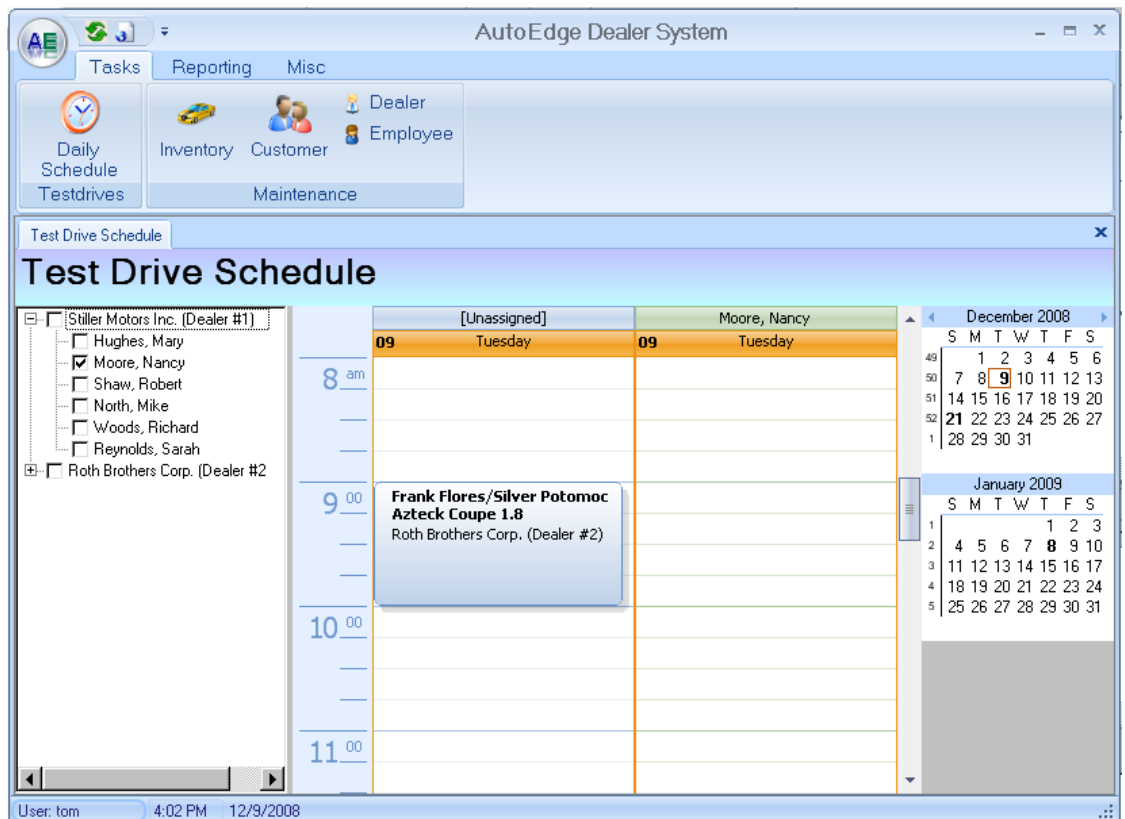
Lab 1 – Setting up the Environment, continued

Compile and Run the Application, continued

- The AutoEdge Login screen will be shown with the User and Password fields already populated. The Container Style will also be set to **Windows .NET GUI**. Click **Login**.



The Daily Schedule screen for the AutoEdge application will be displayed.





continued on next page

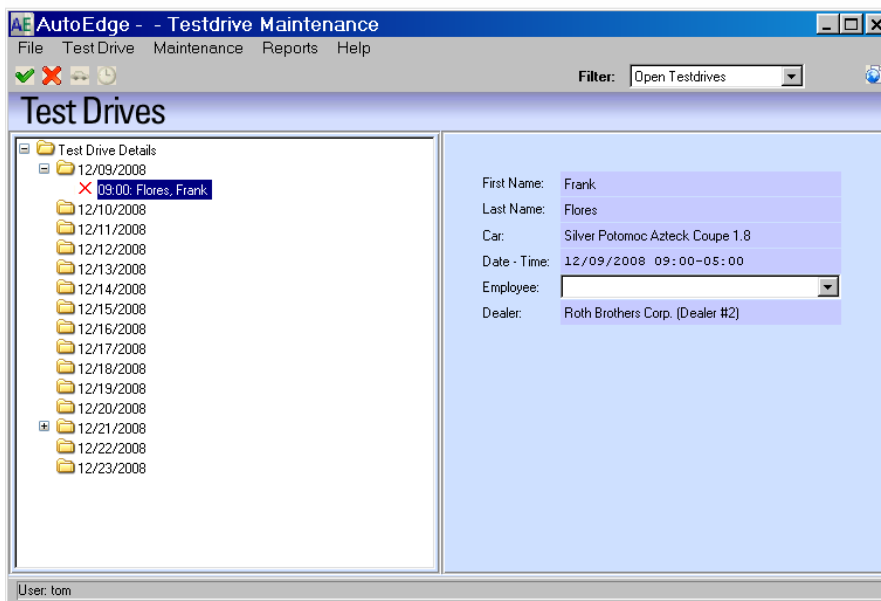
Lab 1 – Setting up the Environment, continued

Compile and Run the Application, continued

14. You can continue using the application to familiarize yourself with its functionality.

 **Note:** Although many parts of the AutoEdge application are functional, the application is not completely implemented as it is a reference application used for demonstration and training purposes.

 **Note:** Selecting the Windows ABL GUI option for the Container Style will demonstrate a version of the AutoEdge application that does not use the new OpenEdge GUI for .NET and Visual Designer functionality.



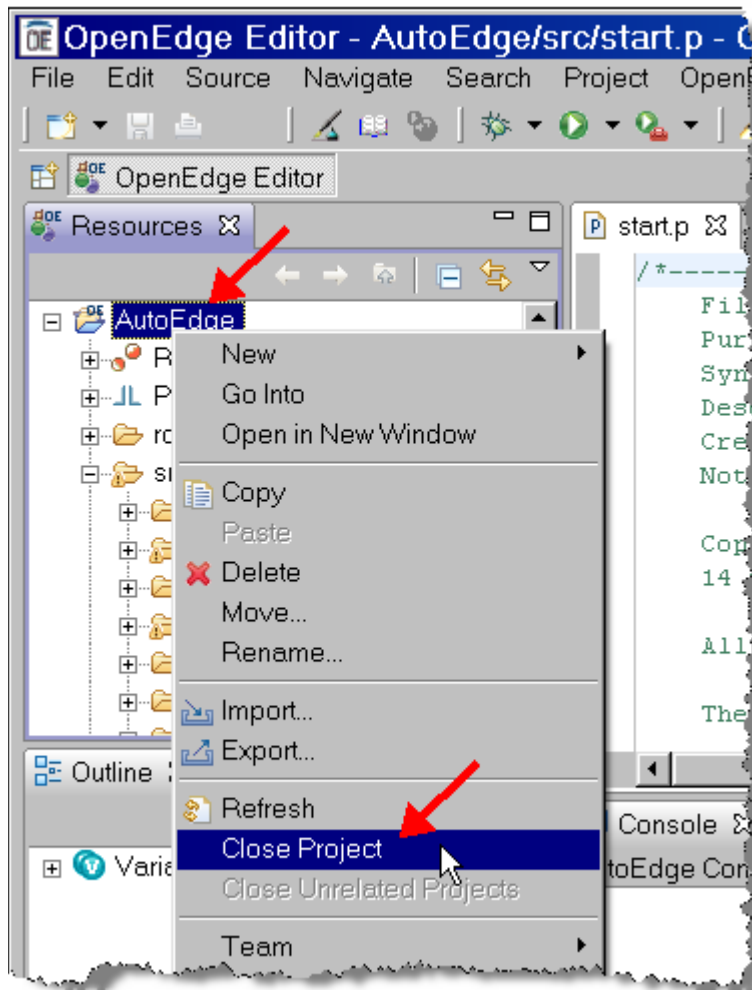
continued on next page

Lab 1 – Setting up the Environment, continued

Closing and opening projects

Workspaces use varying amounts of system resources depending upon how they are configured. You can reduce the amount of resources used by a workspace by closing projects that you are not currently using. Potential benefits of closing unused projects include freeing up database connections and speeding up certain tasks, such as searching. The following steps show how to close and open projects.

1. Select the **AutoEdge** project in the Resources view.
2. Right-click on the **AutoEdge** project and select **Close Project**. The project is closed.

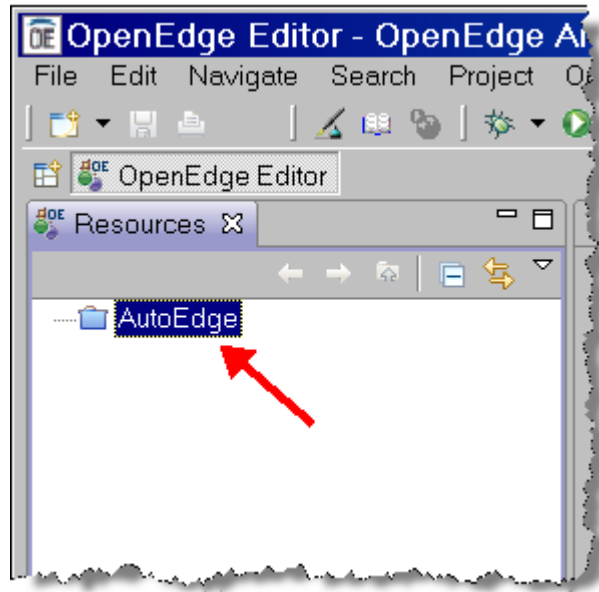


continued on next page

Lab 1 – Setting up the Environment, continued

Closing and opening projects, continued

If there were any database connections open for this project they would also be closed. Also, the AutoEdge project node in the Resources view will become inactive.



3. To reopen the project, right-click on the inactive **AutoEdge** project entry in the Resources view and select **Open Project**. The project is reopened.
-

Lab 2 – Working with Views and Perspectives

Overview

OpenEdge Architect implements tools through the use of views, editors, and perspectives. These components allow developers to easily create, navigate, view, and update the resources and meta information that comprise an application.

This lab shows how to customize perspectives and create new ones by changing the layout of the views in a perspective and modifying the options that are available.

Modifying Perspectives

This section covers some of the options available for working with OpenEdge Architect perspectives.

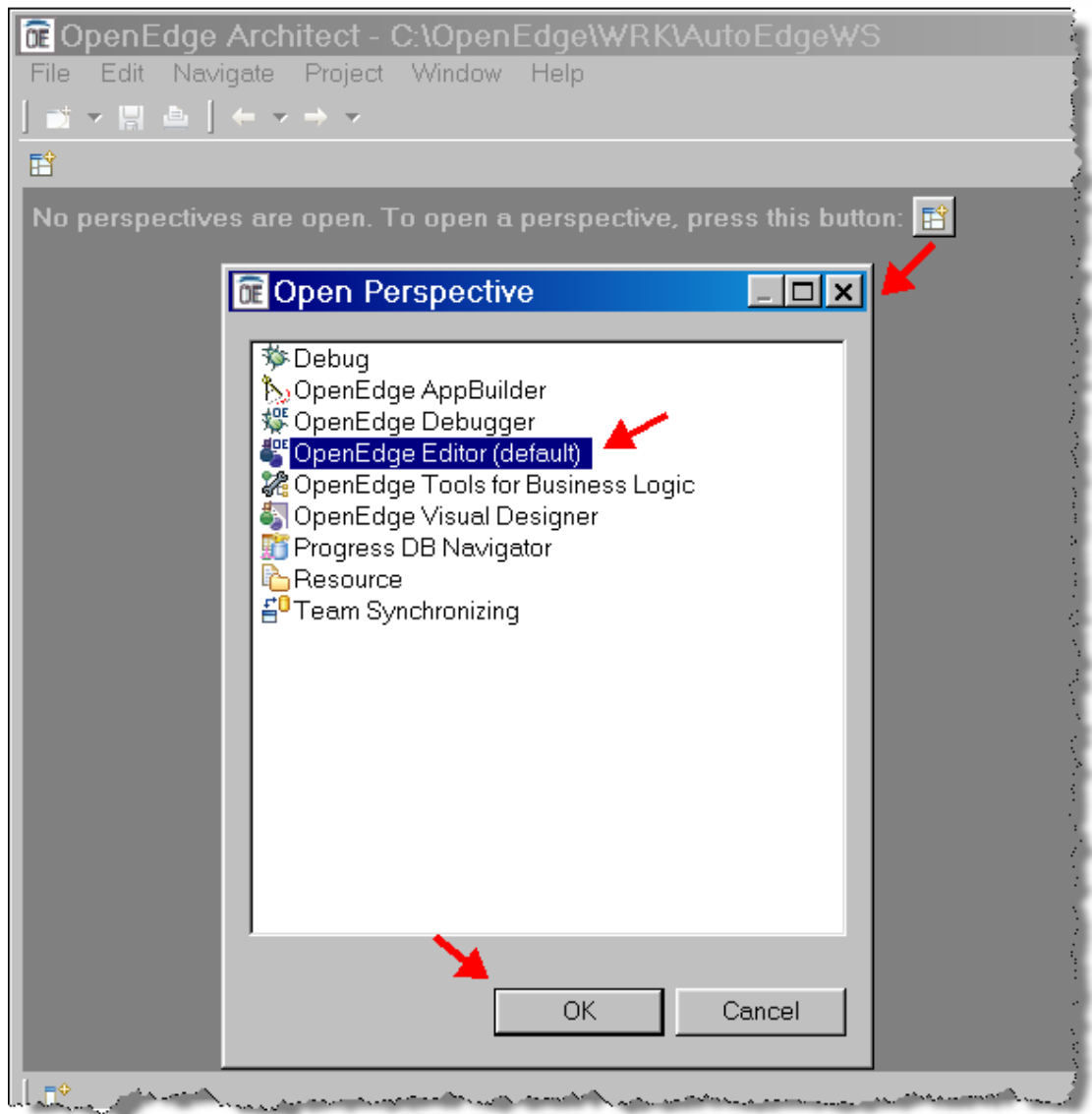
1. From the Architect menu bar, select **Window→Close All Perspectives** to close all the open perspectives.
-

continued on next page

Lab 2 – Working with Views and Perspectives, continued

Modifying Perspectives, continued

2. Since all perspectives are closed, you will see a blank screen with a message indicating that no perspectives are open. Using either the provided button for opening a perspective or the **Window→Open Perspective** menu option, open the **OpenEdge Editor** perspective.

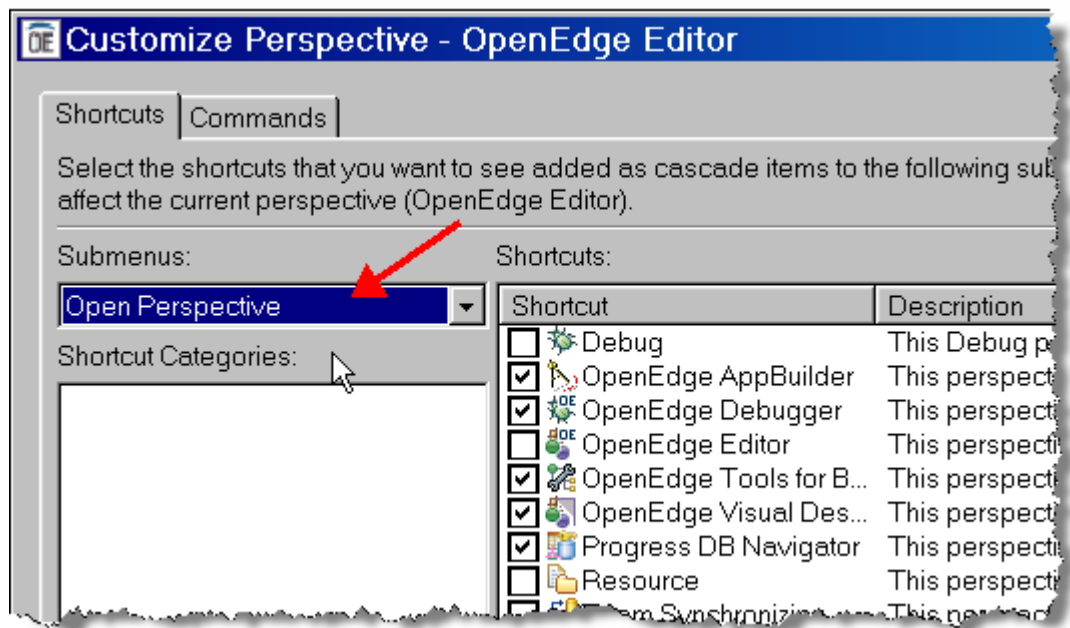


continued on next page

Lab 2 – Working with Views and Perspectives, continued

Modifying Perspectives, continued

3. Choose **Window**→**Customize Perspective**. The **Customize Perspective** dialog appears. This option lets you customize menus and toolbars for the current perspective.
4. From the **Submenus** drop down, select **Open Perspective**.

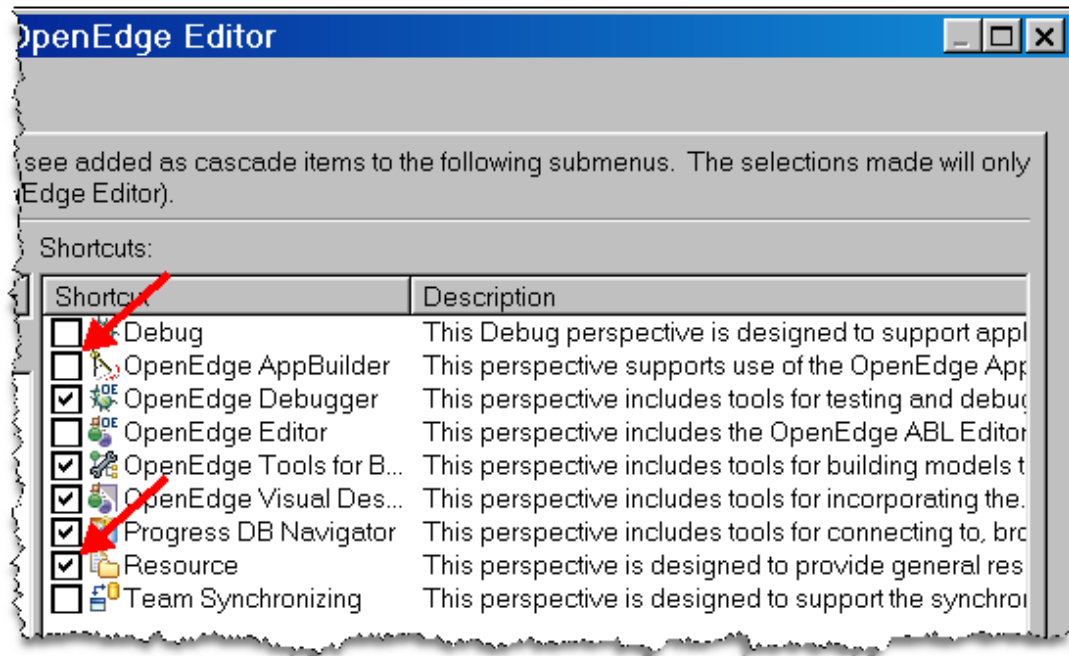


continued on next page

Lab 2 – Working with Views and Perspectives, continued

Modifying Perspectives, continued

5. In the Shortcuts list, un-check **OpenEdge AppBuilder** and check **Resource**:

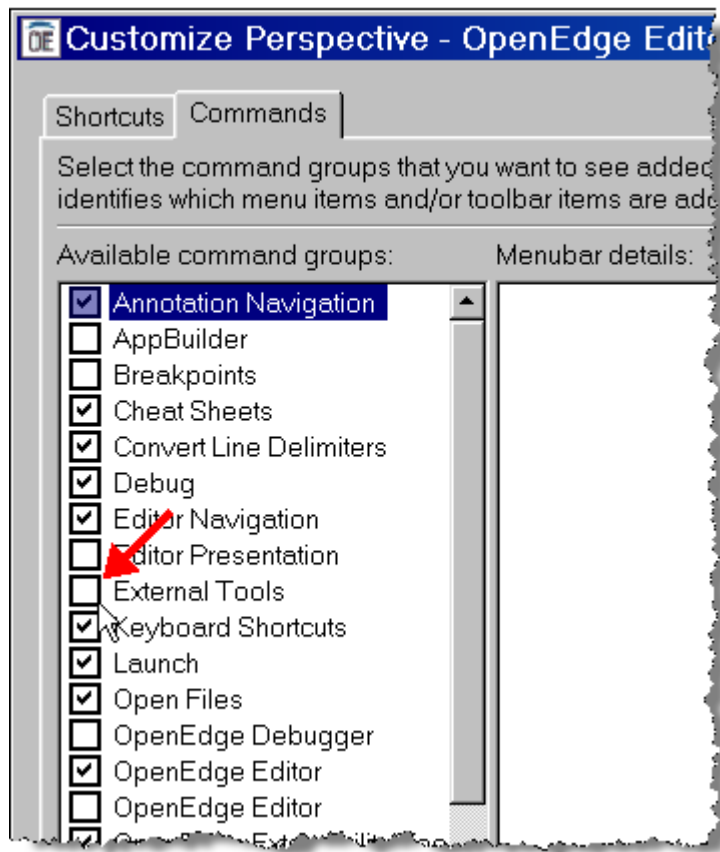


continued on next page

Lab 2 – Working with Views and Perspectives, continued

Modifying Perspectives, continued

6. Select the **Commands** tab in the Customize Perspectives dialog.
7. Un-Check **External Tools** to remove this option from the menu and toolbar options. Consider doing this with other options you rarely use.



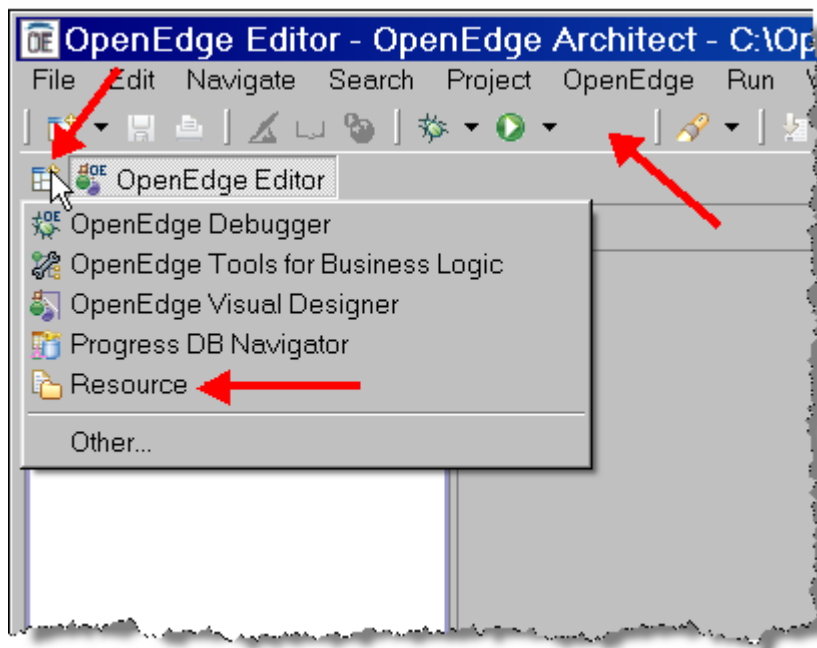
8. Click **OK** to accept the changes.


continued on next page

Lab 2 – Working with Views and Perspectives, continued

Modifying Perspectives, continued

9. Select the **Open Perspective** button, which is next to the list of open perspectives. Notice that the OpenEdge AppBuilder perspective is no longer listed, but the Resource perspective is listed. Also notice that the External Tools toolbar button is no longer shown.



 **Note:** Customizations, such as these, only affect the perspective in which they were set.

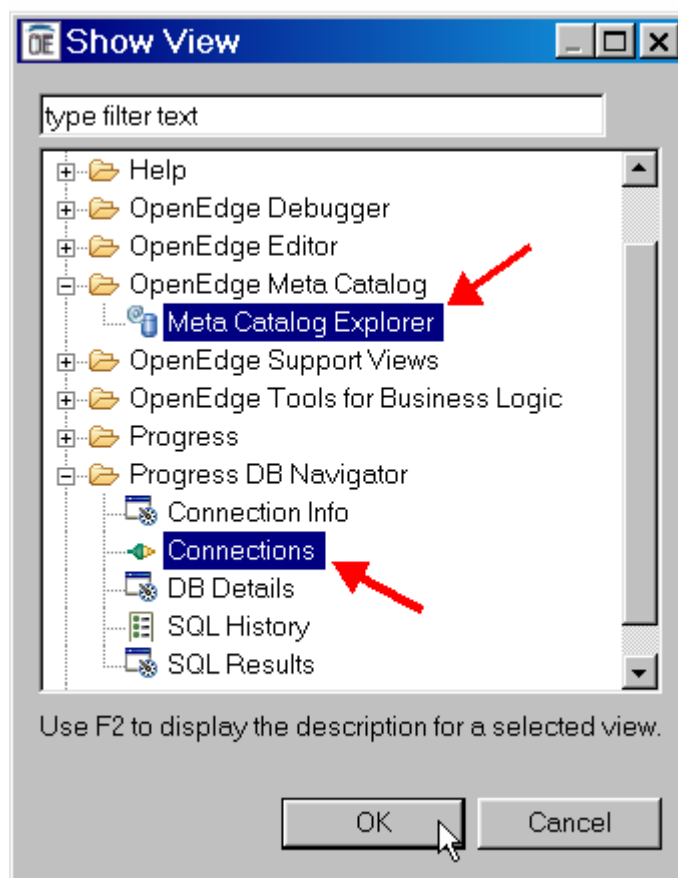
10. From Architect's menu bar, select **Window**→**Show View**→**Other**. The Show View dialog appears.

continued on next page

Lab 2 – Working with Views and Perspectives, continued

Modifying Perspectives, continued

11. Expand the **OpenEdge Meta Catalog** node and select the **Meta Catalog Explorer** option.
12. Expand the **Progress DB Navigator** node. Hold down the Ctrl key and select the **Connections** option. Use this technique whenever you want to select multiple options.

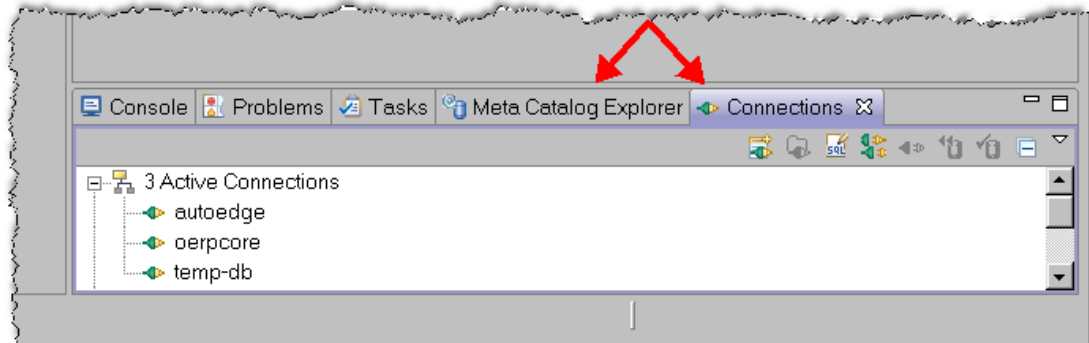


continued on next page

Lab 2 – Working with Views and Perspectives, continued

Modifying Perspectives, continued

13. Click **OK**. Both the Connections view and the Meta Catalog Explorer view are opened.

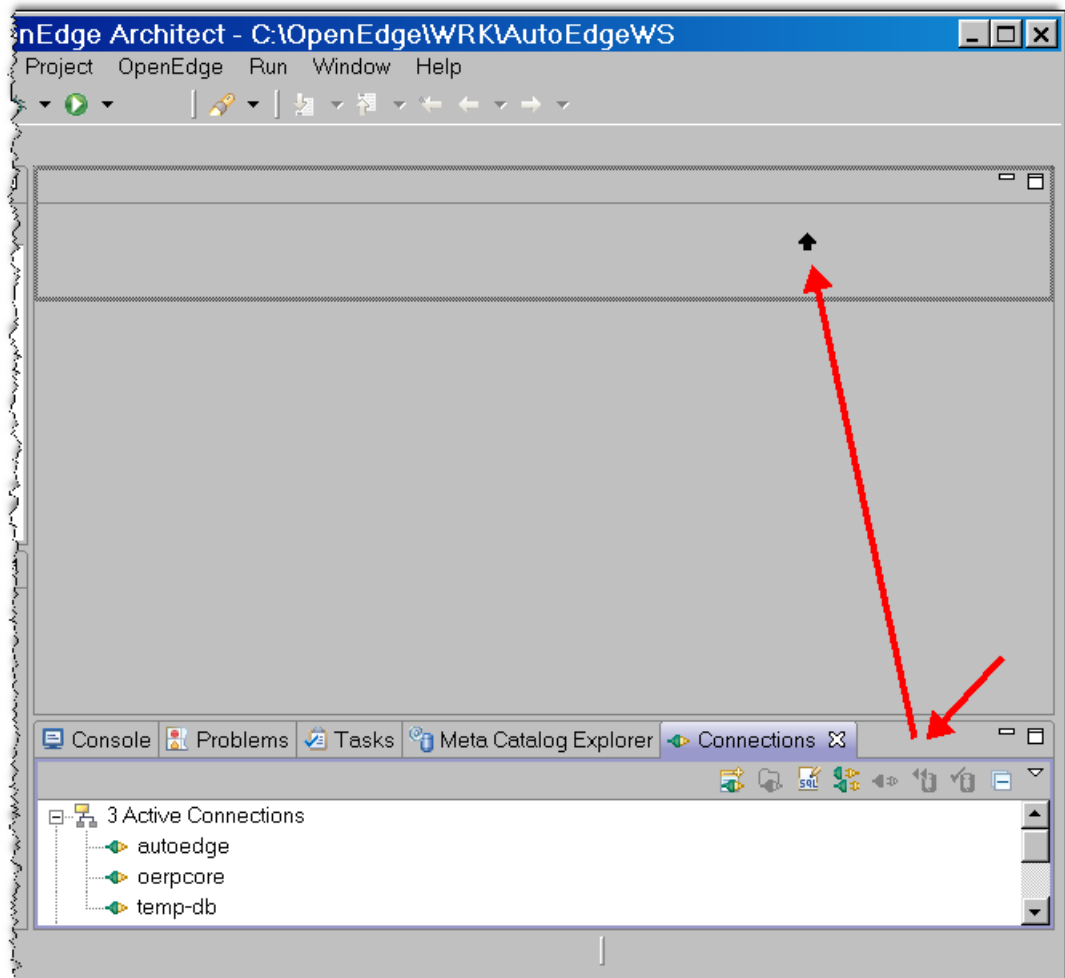



continued on next page

Lab 2 – Working with Views and Perspectives, continued

Modifying Perspectives, continued

14. Select the group of views that includes the Console through to the Connections views by clicking just **past** the last tab and holding down the mouse button. Drag the group to the top of the Workspace above the Editor area. All of the views will be moved and docked there.



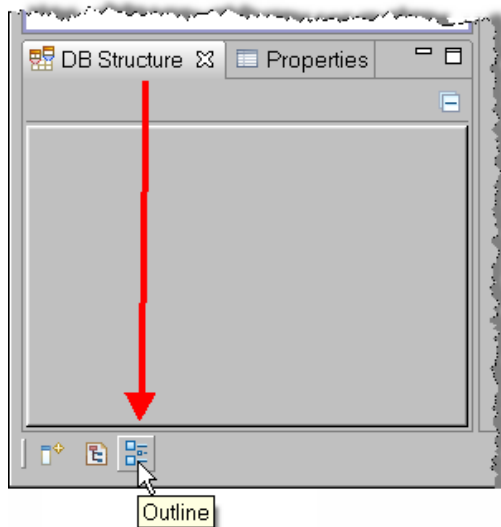
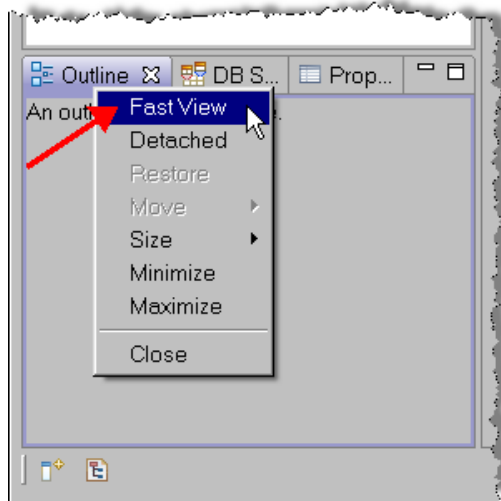
 **Note:** You can also move individual views by clicking on the view's tab and dragging it to the desired location.


continued on next page

Lab 2 – Working with Views and Perspectives, continued

Modifying Perspectives, continued

15. Right-click on the **Outline** view and select **Fast View**. The view is minimized to the lower left of the Workbench.



 **Tip:** Clicking on a Fast View icon, such as the Outline view, displays the view. Clicking again anywhere outside the view minimizes the view again. This provides a nice way to keep a view easily accessible without taking up space when it isn't being used. A Fast View can be turned off by right-clicking on the icon and un-checking Fast View.

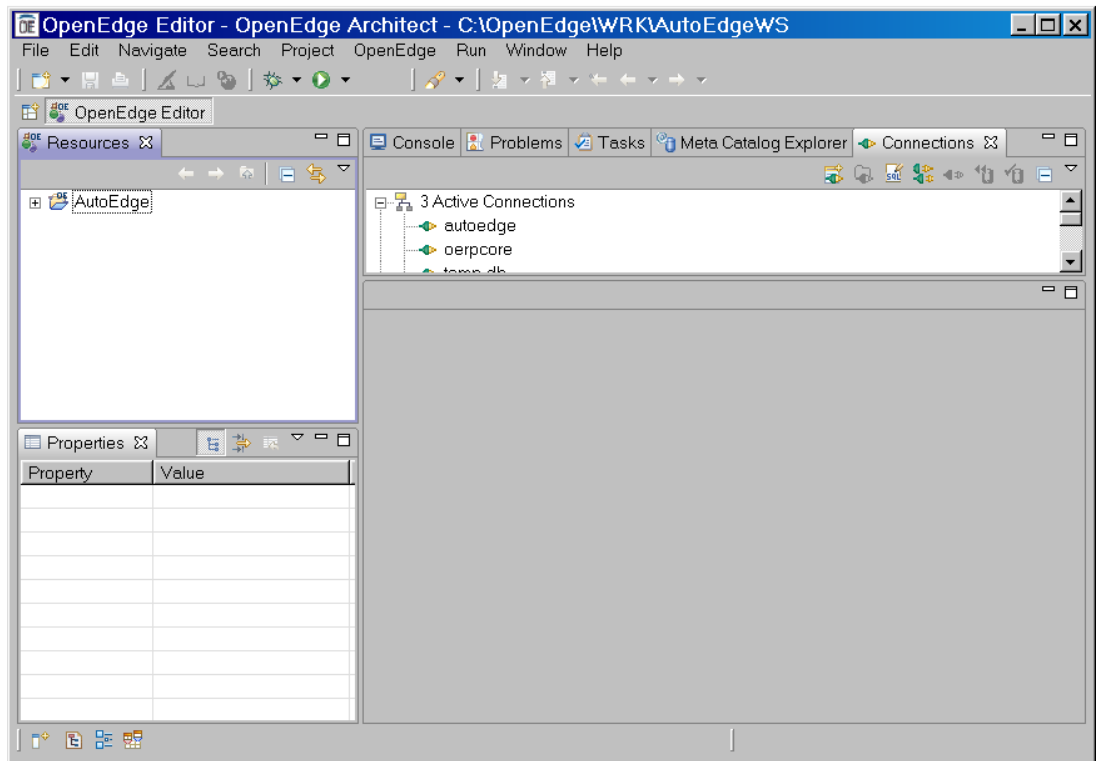
continued on next page

Lab 2 – Working with Views and Perspectives, continued

Modifying Perspectives, continued

16. Change the DB Structure view so that it too is a **Fast View**.

The workbench should look similar to the following. Note that it does not matter if it slightly different, since the goal is just to create a perspective different than the default OpenEdge Editor perspective.



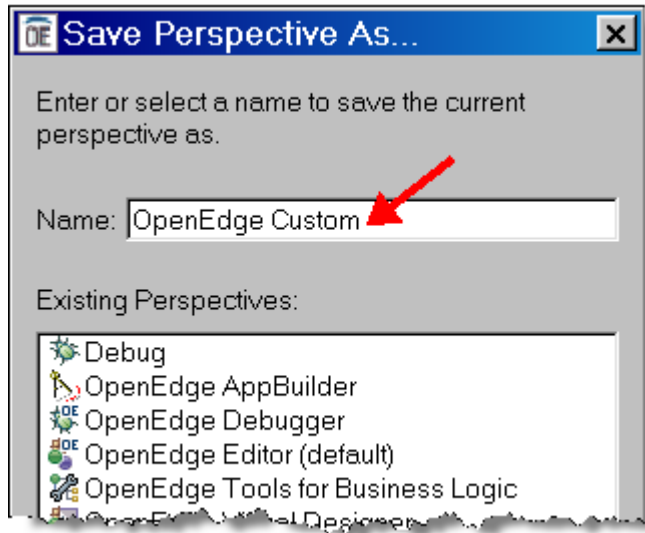
continued on next page


Lab 2 – Working with Views and Perspectives, continued

Saving Customized Perspectives

You can continue to use perspectives that have been modified; however, if the perspective is ever reset (Windows→Reset Perspective) you will lose all the changes you have made. Therefore, if you want to make the changes permanent, it is suggested that you save the modified perspective. It is also suggested that you save the changes in a new perspective so that the default perspectives shipped with Architect are not affected.

1. From the Architect menu bar, choose **Window→Save Perspective As**. The Save Perspective As dialog appears.
2. Enter the **OpenEdge Custom** as the Name.



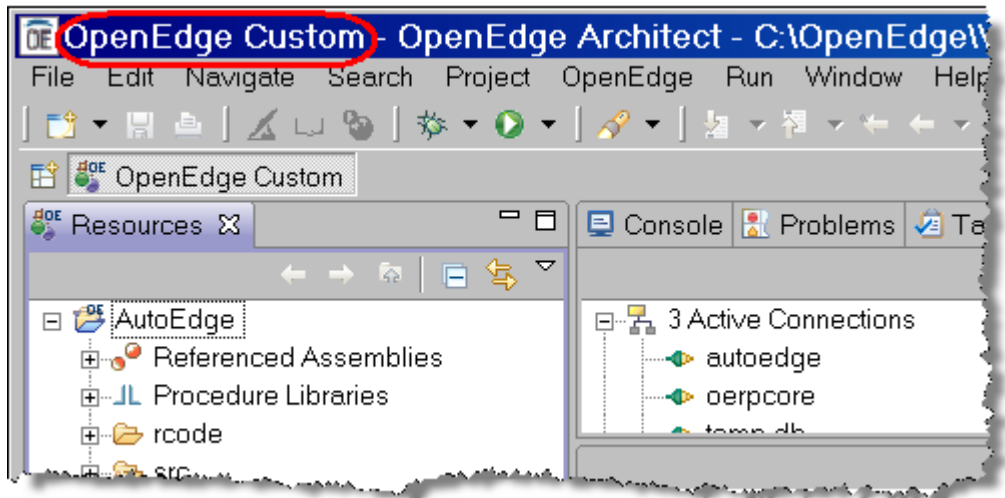
 **Caution:** It is important to change the name when creating a custom perspective. If changes are saved to the same name as an existing perspective (in this case, the OpenEdge Editor perspective), then there is no way to restore that perspective back to its default layout.

continued on next page

Lab 2 – Working with Views and Perspectives, continued

Saving Customized Perspectives, continued

3. Click **OK**. The perspective name is changed from **OpenEdge Editor** to **OpenEdge Custom**.

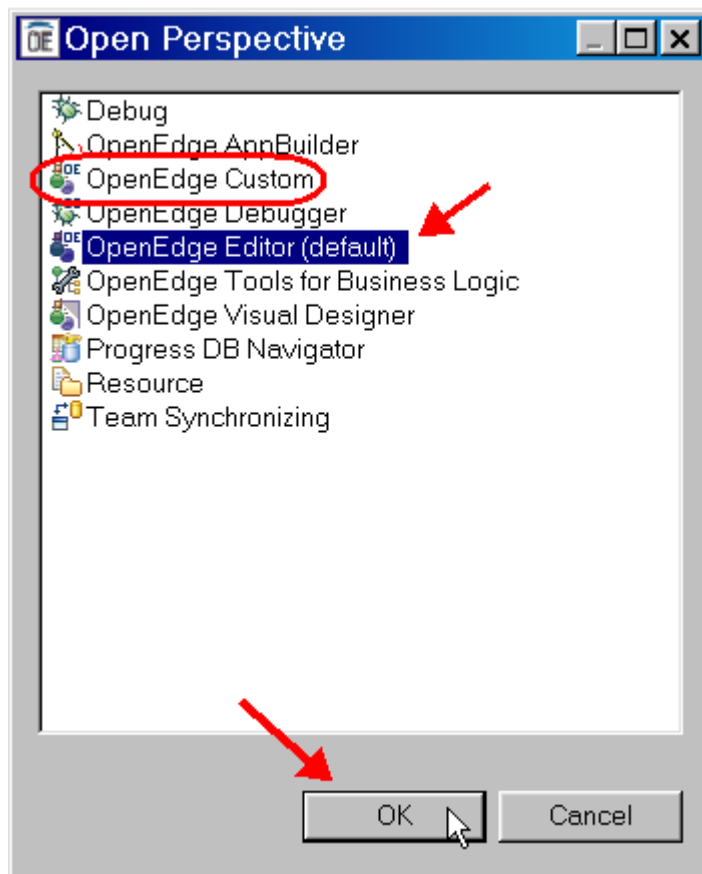


continued on next page

Lab 2 – Working with Views and Perspectives, continued

Saving Customized Perspectives, continued

4. Select **Window**→**Open Perspective**→**Other**. The OpenEdge Custom perspective is listed as one of the available perspectives. Open the **OpenEdge Editor** perspective.



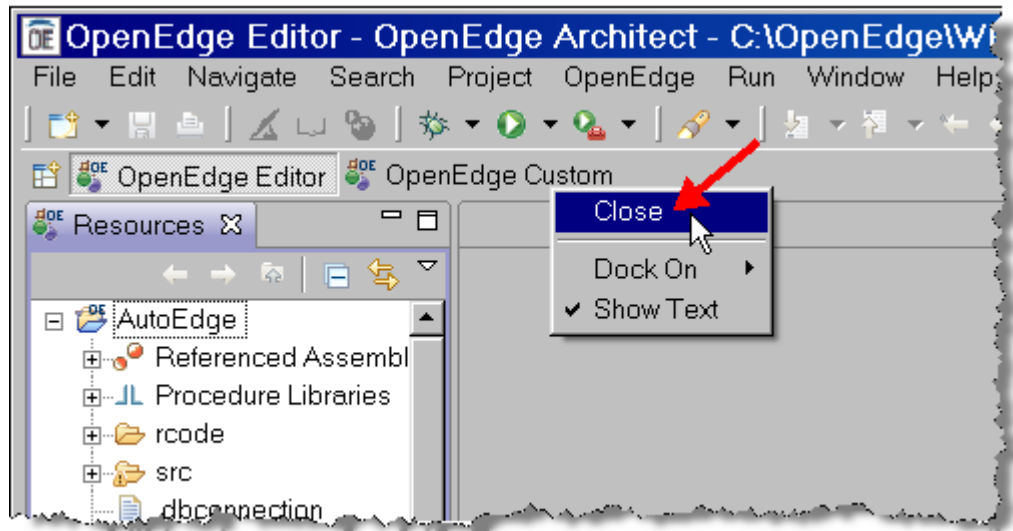
The OpenEdge Editor perspective remains unchanged.

continued on next page

Lab 2 – Working with Views and Perspectives, continued

Saving Customized Perspectives, continued

5. Right-click on **OpenEdge Custom** perspective in the perspective toolbar. Select the **Close** option to close the perspective.



Lab 3 – Using Architect’s Help Features

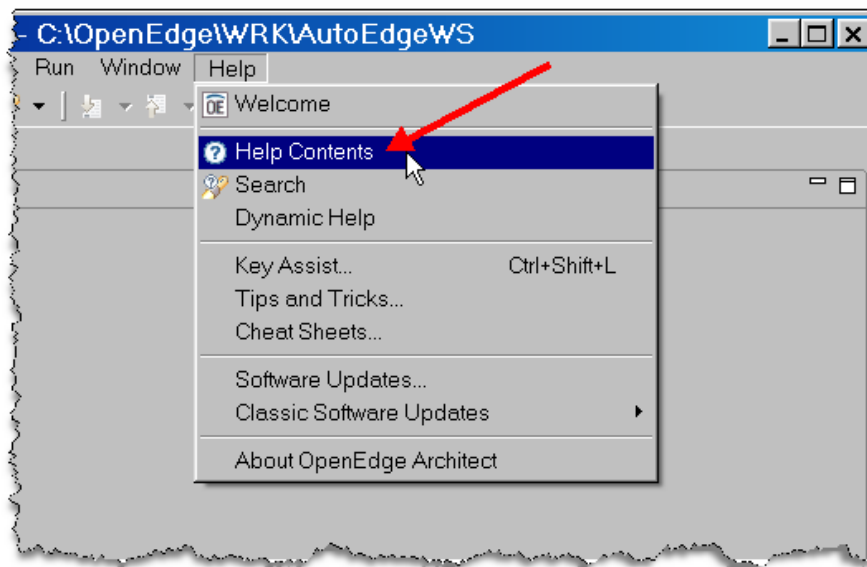
Overview


OpenEdge Architect includes extensive on-line help capabilities and help for both the Eclipse Framework (Workbench User Guide) and help specific to Architect (OpenEdge Architect Guide). If you add additional functionality via plug-ins, then additional topics are likely to appear under the help contents. You can specify which parts of the help you would like to search. This can speed up searching and reduce the number of unwanted matches.


Accessing On-Line Help

This section shows how to use the basic help features and how to set filtering options for searching.

1. From the Workbench menu, select **Help→Help Contents** option to open the OpenEdge Architect Help window.



 **Note:** Architect also includes additional help features from this menu, including Key Assist, which displays a quick reference pop-up of all available keyboard short-cuts, and Cheat Sheets, which displays the steps for some common tasks in Architect.

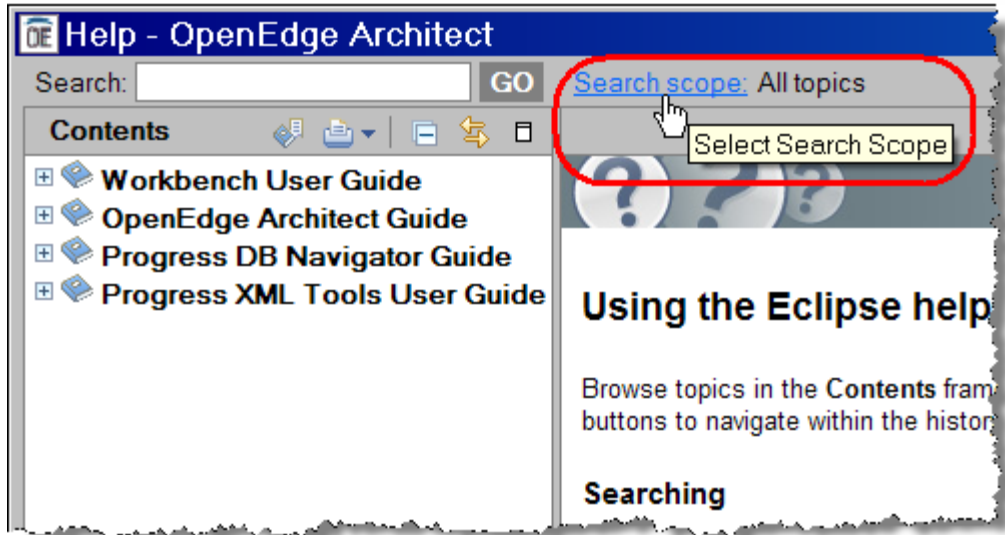
 **Note:** ABL syntax help is also available through the Keyword Help view. You can access this view by pressing Shift-F2 while the cursor is over a keyword in an ABL editor.

continued on next page

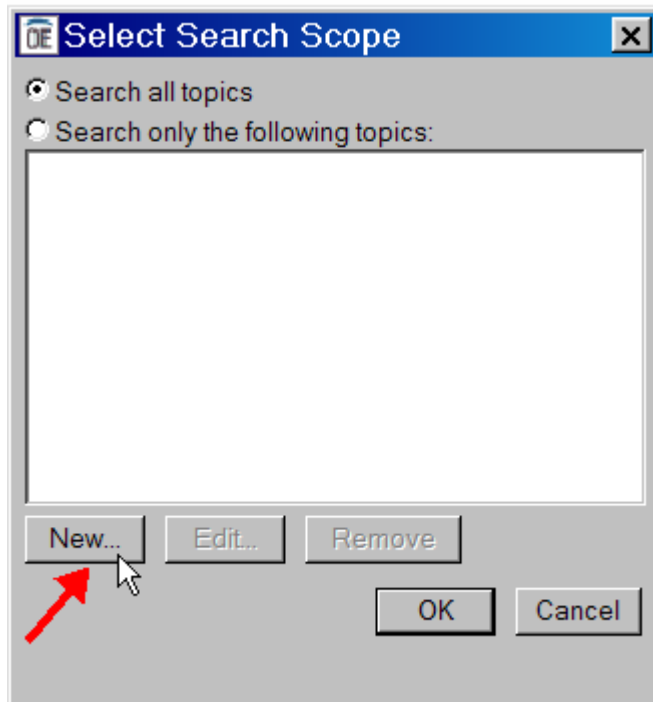
Lab 3 – Using Architect’s Help Features, continued

Accessing On-Line Help, continued

2. Select the **Search scope** link to open the Select Search Scope dialog.



3. Click **New** to open the New Search List dialog.

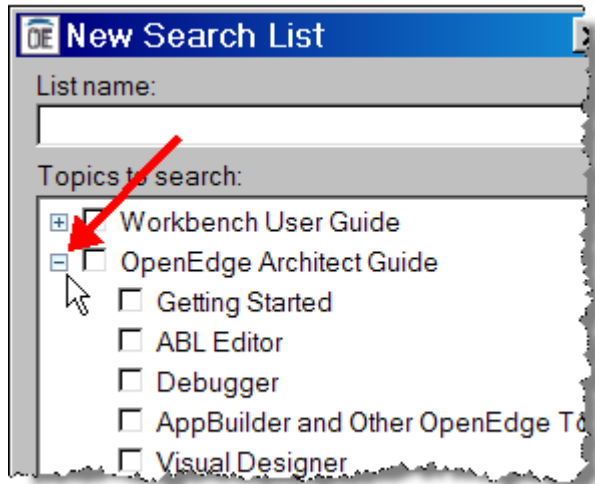


continued on next page

Lab 3 – Using Architect’s Help Features, continued

Accessing On-Line Help, continued

4. Expand the **OpenEdge Architect Guide** node. The nodes displayed show the level of granularity that can be specified for searching. You can select multiple nodes to be included in the search.

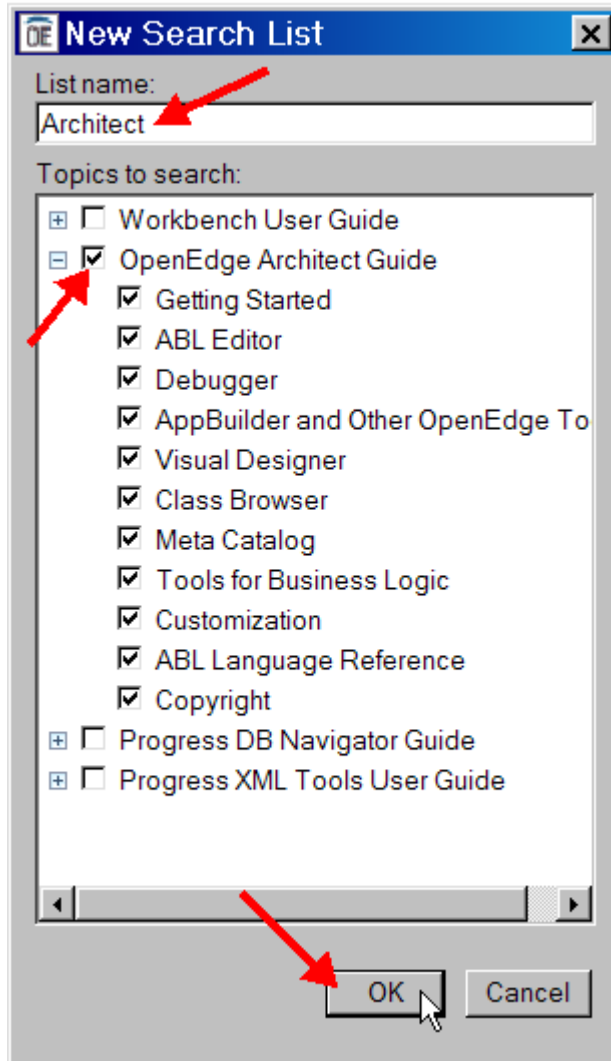


continued on next page

Lab 3 – Using Architect’s Help Features, continued

Accessing On-Line Help, continued

5. Type **Architect** in the List name field. This will be the displayed name in the list of topic options. Check **OpenEdge Architect Guide** option. This search scope will include all of the OpenEdge documentation. Click **OK**.

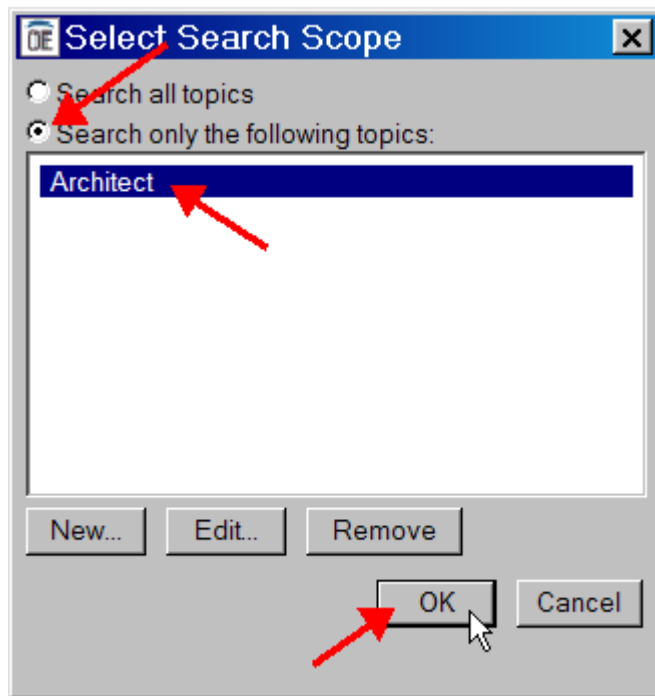


continued on next page

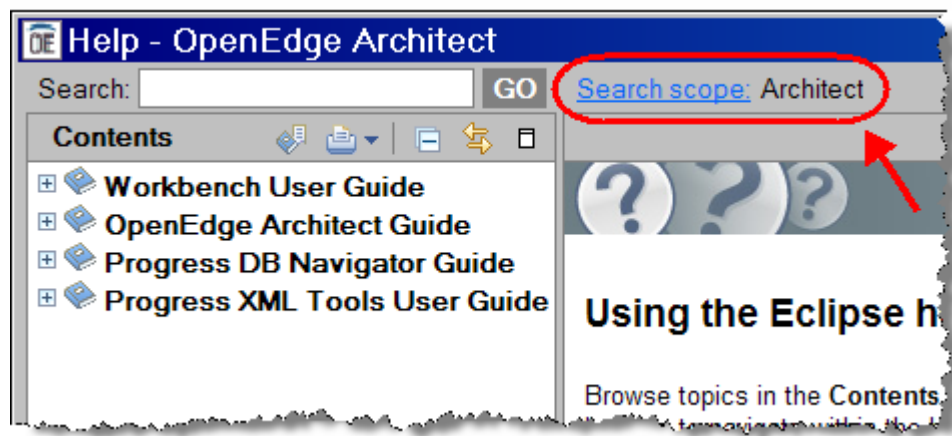
Lab 3 – Using Architect’s Help Features, continued


Accessing On-Line Help, continued

6. Confirm that **Search only the following topics** option is selected and that **Architect** (the topic you just created) is highlighted. Click **OK**.



The help window now shows the search scope set to Architect.



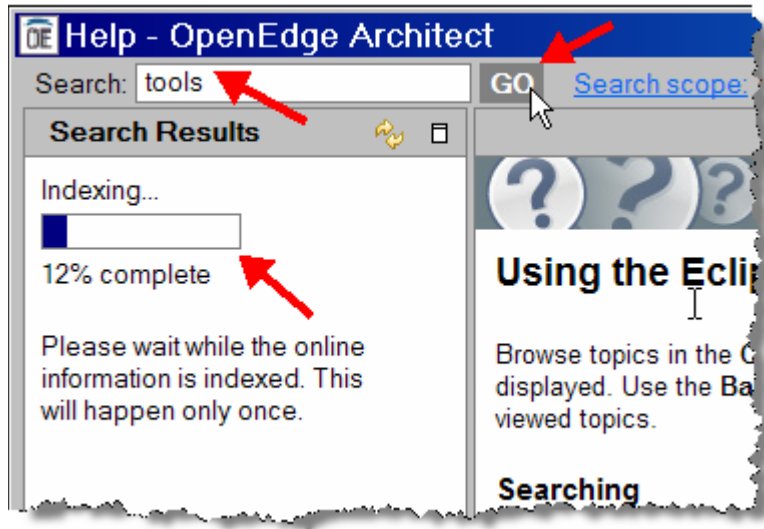
 **Note:** The search scope is retained, so the next time you start this workspace and open the help screen, the search scope will still be set to Architect.

continued on next page

Lab 3 – Using Architect’s Help Features, continued

Accessing On-Line Help, continued

7. Type **tools** in the Search field or any topic of your choice. Click **GO**. The first time help is accessed, indexing will be performed and may take a few seconds to complete.

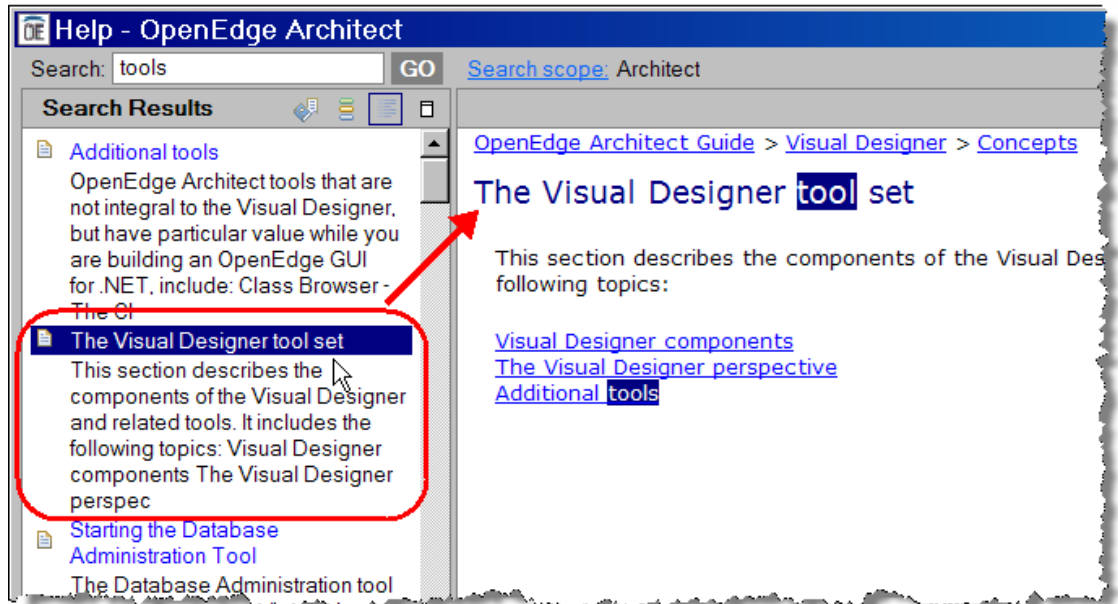


continued on next page

Lab 3 – Using Architect’s Help Features, continued

Accessing On-Line Help, continued

After the indexing is complete the search results are displayed. You can click on one of the results to open the corresponding help topic in the right pane.



If you enter in a second search request, you'll notice it completes much faster since the help has now been indexed.

8. Close the Help screen.
-

Lab 4 – Using the DB Navigator

Overview

The DB Navigator provides access to any SQL compliant database and includes the following features:

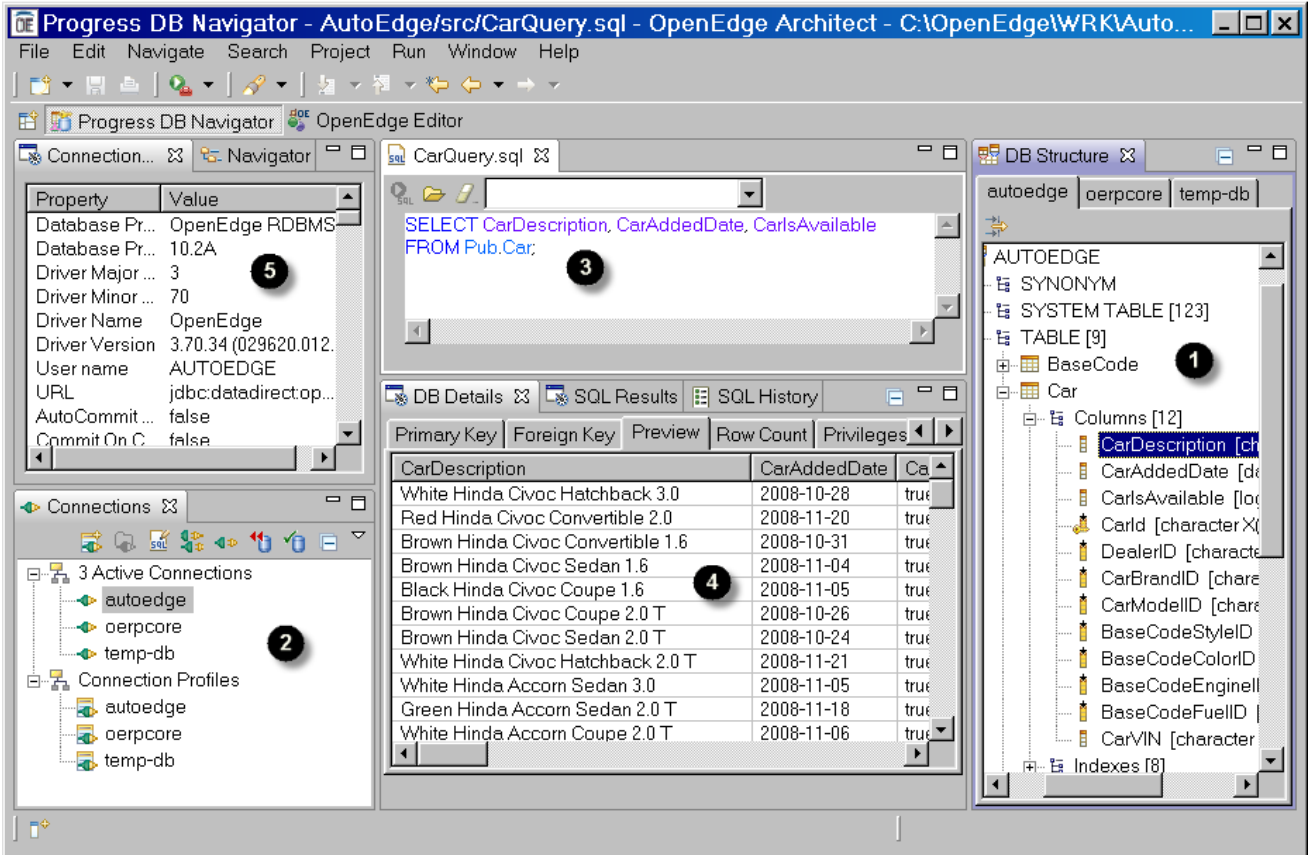
- Provides a hierarchical view of database schema.
- Includes a SQL editor which allows you to write SQL code against a database.
- Offers code generation tools to create SQL Create Table statements.
- Offers tools for maintaining database schema, including the OpenEdge RDBMS.
- Offers tools for maintaining OpenEdge database sequences and triggers.

The goal of this lab is to explore some of the functionality of the OpenEdge Architect DB Navigator.

continued on next page

Lab 4 – Using the DB Navigator, continued

The DB Navigator Perspective



Key:

- ❶ DB Structure View
- ❷ Connections View
- ❸ SQL Editor
- ❹ DB Details, SQL Results, and SQL History Views
- ❺ Connection Info View

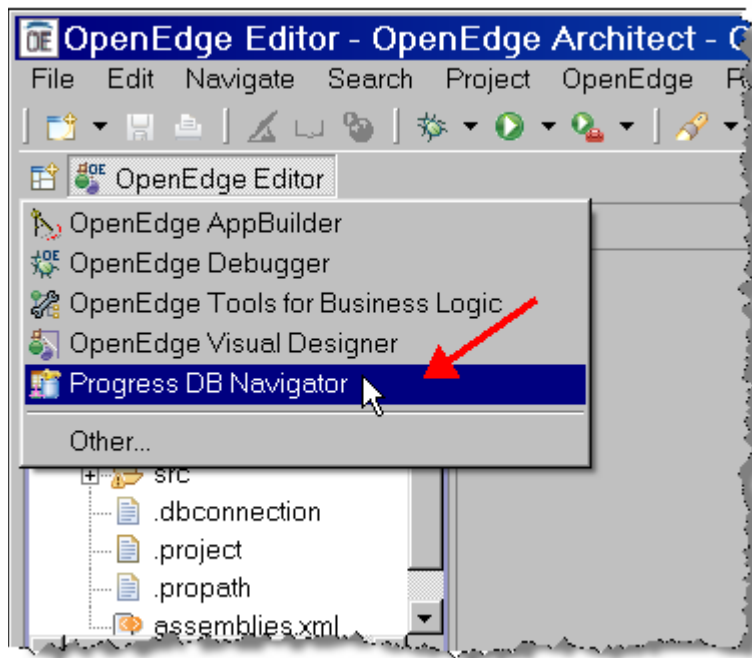
continued on next page


Lab 4 – Using the DB Navigator, continued

Manually connecting databases

In *Lab 1 – Setting up the Environment*, you set up the SQL connection options so that the defined databases are automatically connected at startup. However, if the workspace has not been restarted since the Database Connections were created, then the databases may not be connected. There are other times when you may find it necessary to force a SQL connection to a database, such as a database that you have not set to automatically connect at startup because it is not often used. This section shows how to make a manual SQL connection to a database using an existing Database Connection Profile.

1. Click the Open Perspective button in Perspectives toolbar and select the **Progress DB Navigator** option. The Progress DB Navigator perspective opens.



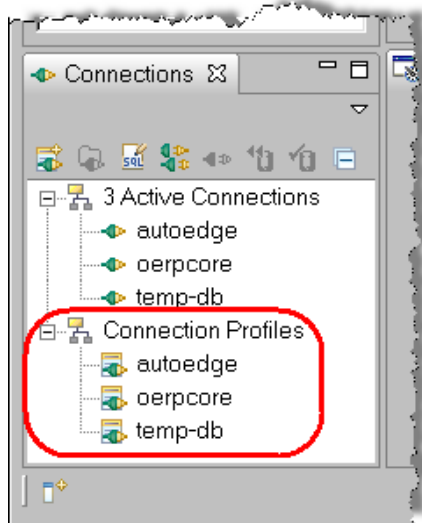
 **Note:** You can also use the **Window→Open Perspective→Progress DB Navigator** menu option to open the perspective.

continued on next page

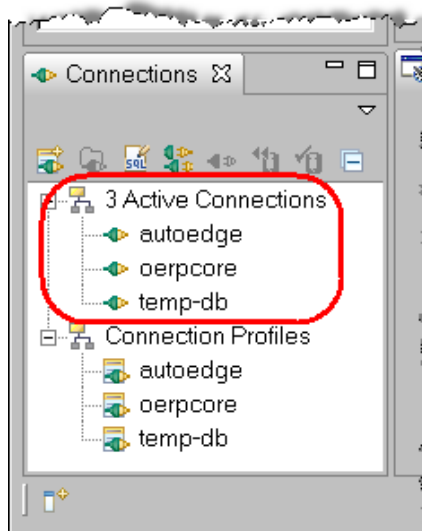
Lab 4 – Using the DB Navigator, continued

Manually connecting databases, continued

2. In the **Connections** view in the lower left of the perspective, verify that the **autoedge**, **oerpcore** and **temp-db** entries are shown under the **Connection Profiles** node. These entries are for the connection profiles that were defined in the Migrating an Application lab.



3. Also verify the **autoedge**, **oerpcore** and **temp-db** databases are listed in the Active Connections list. If a connection is listed as active, then it means that a SQL connection is active for the Connection Profile.

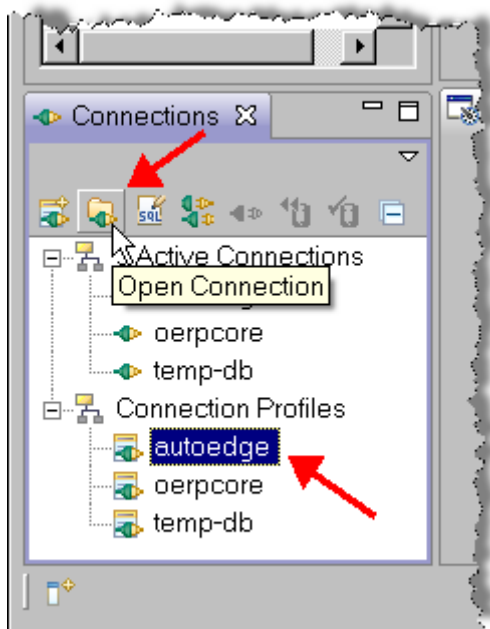


continued on next page

Lab 4 – Using the DB Navigator, continued

Manually connecting databases, continued

4. If a database is not connected, then select the database's Connection Profile and click on the **Open Connection** button.



A brief status screen will be displayed. If the connection was successful, the database will be listed under the Active Connections node. Repeat this step for any other databases the need to be connected

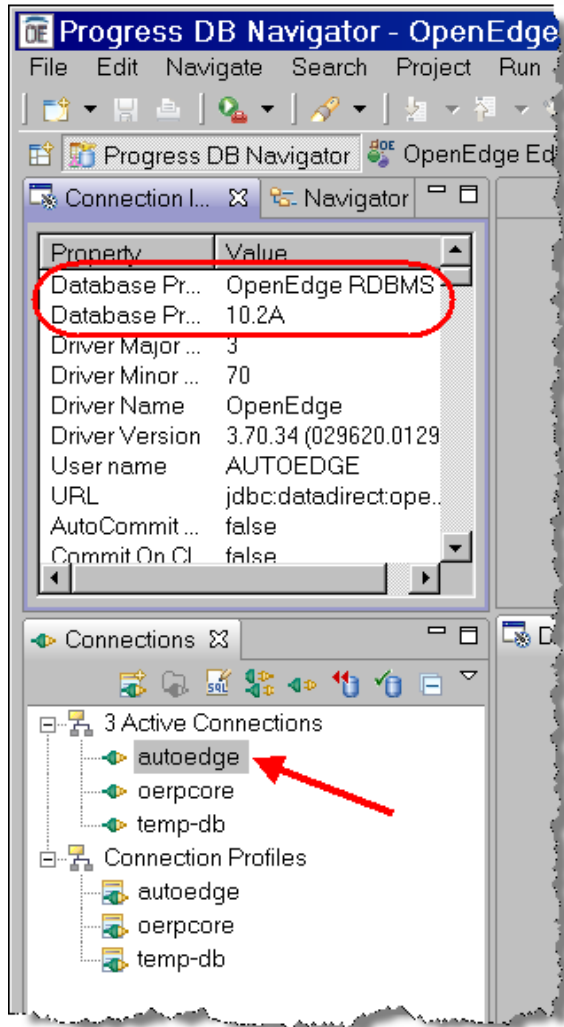
continued on next page

Lab 4 – Using the DB Navigator, continued

Connection Info view

The Connection Info view shows general information for the database that is currently selected under the Active Connections node.

1. In the Connections view, select the **autoedge** entry under the **Active Connections** node.



The Connection Info view shows database information about the selected database. For example, for the autoedge database you can see that it is an OpenEdge RDBMS that was created with version 10.2A.

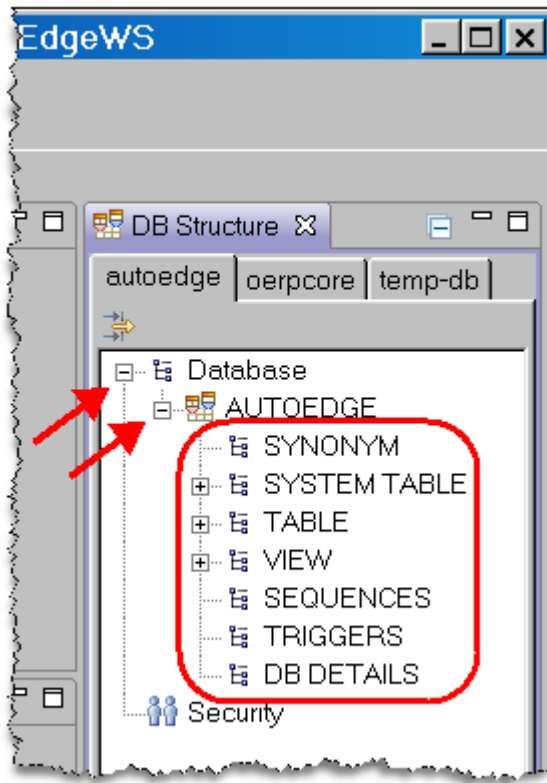
continued on next page

Lab 4 – Using the DB Navigator, continued

Exploring database schema using DB Navigator

The steps in this section demonstrate some of the database schema information that can be viewed using the DB Navigator.

1. The **DB Structure** view contains tab folders for each connected database. Select the **autoedge** tab and expand the **Database** and **AUTOEDGE** nodes. The top level schema nodes are displayed.

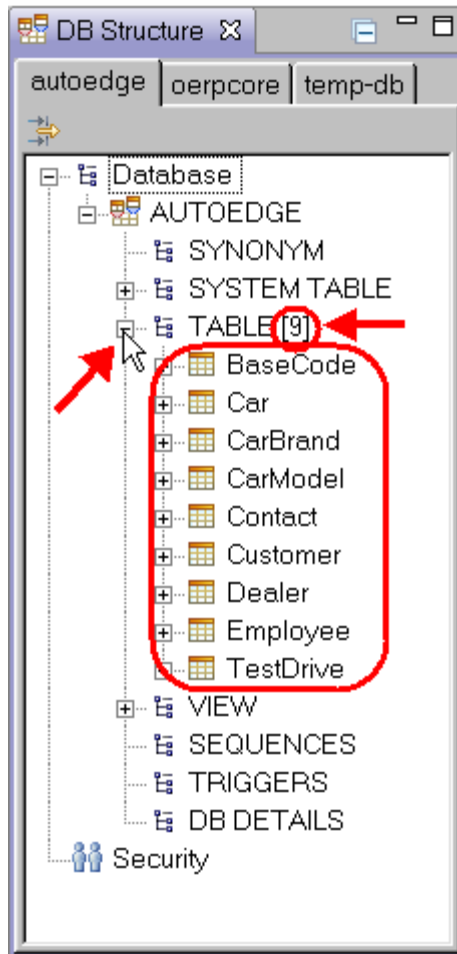


continued on next page

Lab 4 – Using the DB Navigator, continued

Exploring database schema using DB Navigator, continued

2. Expand the **TABLE** node.



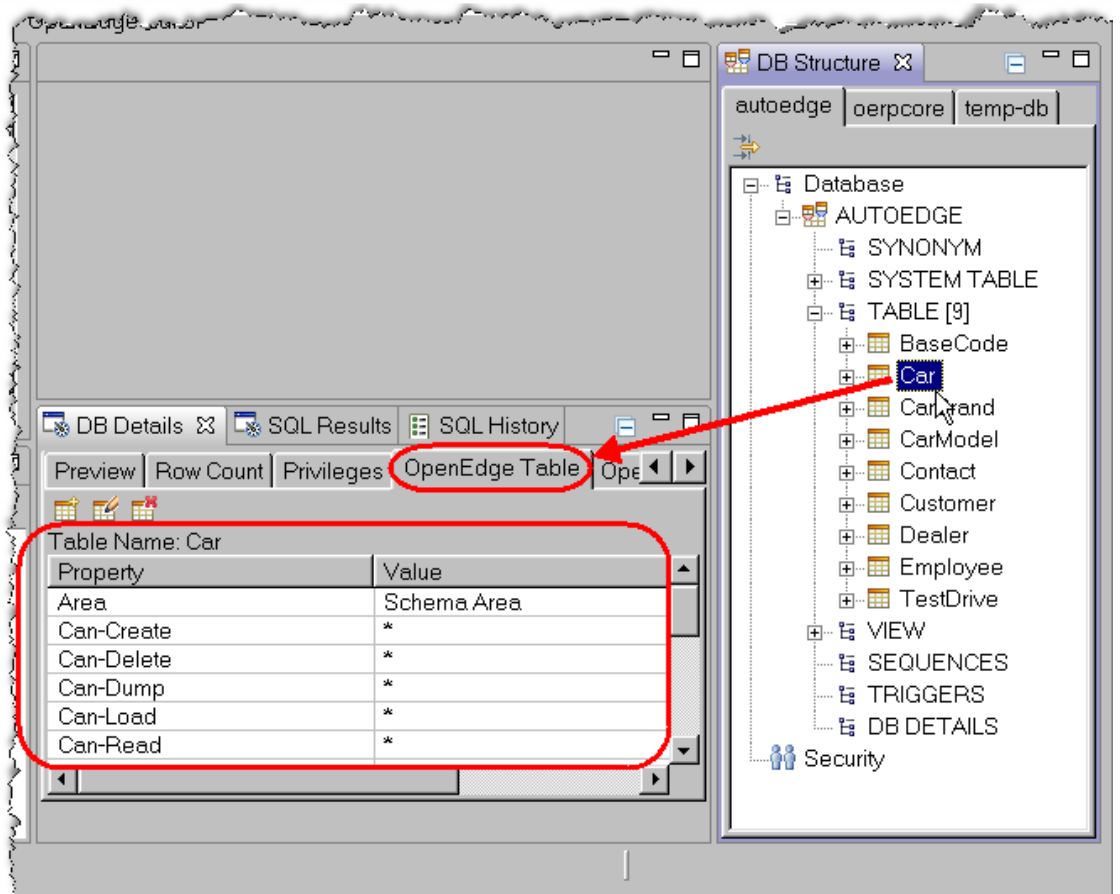
Note that the number of non-system tables in the database is shown in brackets next to the **TABLE** node. For the Autoedge database, there are nine tables.


continued on next page

Lab 4 – Using the DB Navigator, continued

Exploring database schema using DB Navigator, continued

3. Select the **Car** table under the TABLE node. When the table is selected, the **DB Details** view is populated with schema information about the table. In this case, the attributes for the table are shown in the **OpenEdge Table** tab of the DB Details view.



 **Note:** DB Navigator can be used with any JDBC compliant database. The tabs in the DB Details view for **OpenEdge Table**, **OpenEdge Columns**, and **OpenEdge Indexes** are specific to OpenEdge and displayed only when an OpenEdge database is connected. All other tabs are displayed regardless of the database type.

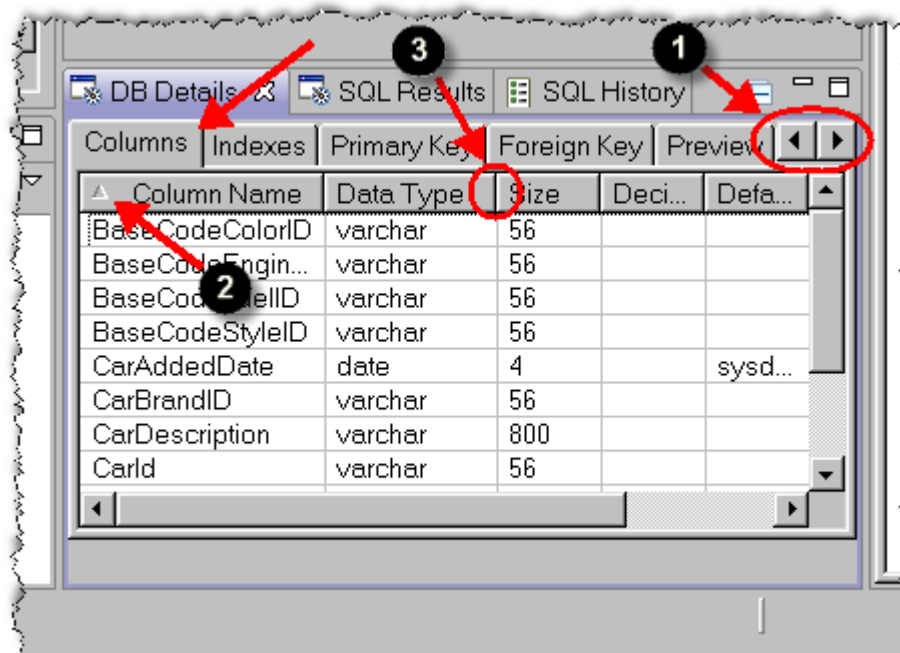
continued on next page


Lab 4 – Using the DB Navigator, continued

Working with the DB Details view

The DB Details view displays an enormous amount of information about a connected database across a number of tabs. This section explains the purpose and features of some of these tabs.

1. In the DB Details view, select the **Columns** tab. If the tab is not visible, use the arrow buttons (❶) to navigate the tabs until it is displayed. All of the columns and column attributes for the Car table are listed in the grid. This tab shows SQL information pertaining to the columns.



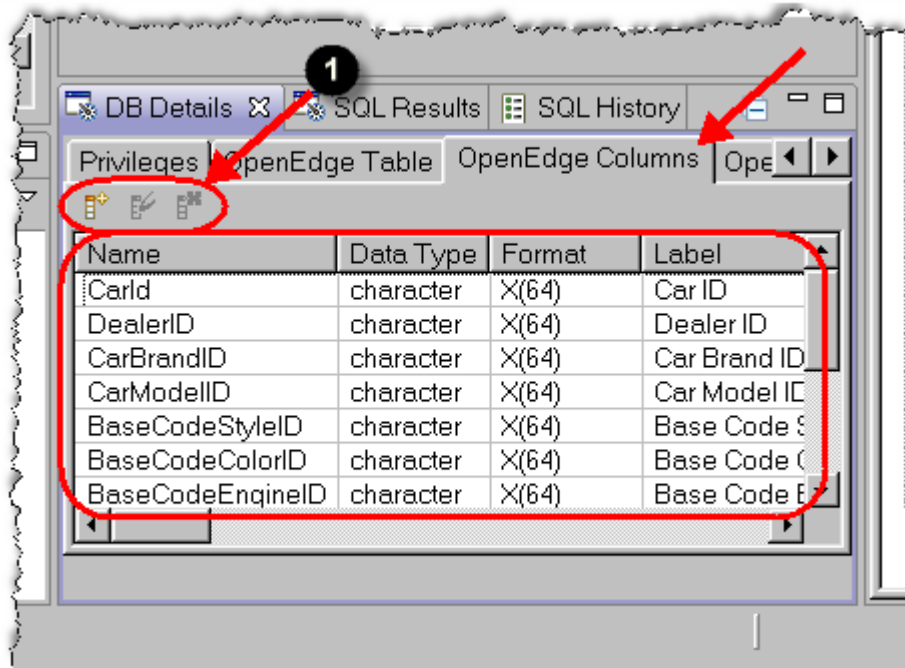
 **Tip:** You can sort the grid by attribute ascending or descending order by clicking on any column header (❷). A small arrow will be shown in the column header indicating that that column is used to sort the grid. This allows you to sort the grid by such attributes as Column Name or Data Type. Grid columns can also be resized by moving the mouse to the line separating the two columns (❸) and clicking-and-dragging the column to the preferred width.


continued on next page

Lab 4 – Using the DB Navigator, continued

Working with the DB Details view, continued

2. Select the **OpenEdge Columns** tab. Unlike the Columns tab, which shows SQL information about the columns in a database, the OpenEdge Columns tab shows column information that is OpenEdge specific.



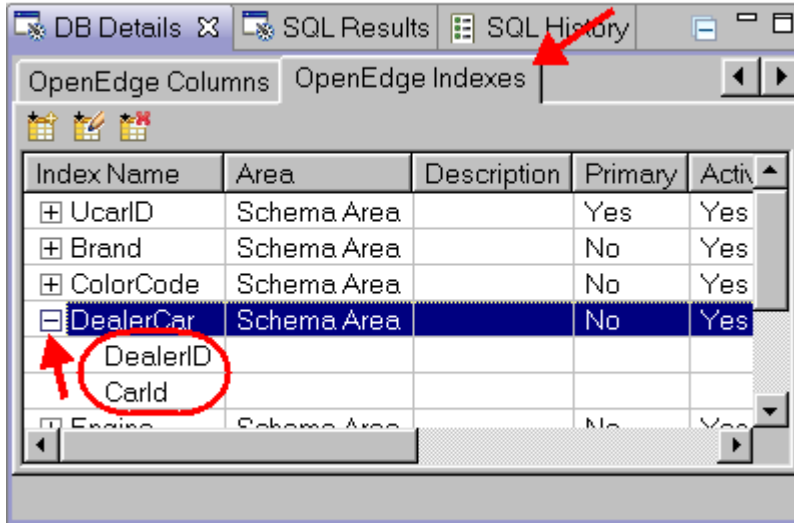
 **Tip:** The toolbar above the grid (1) allows you to add, update, and delete columns in the database. Toolbars to change schema are available in the OpenEdge Columns, OpenEdge Indexes, and OpenEdge Tables tabs.

continued on next page

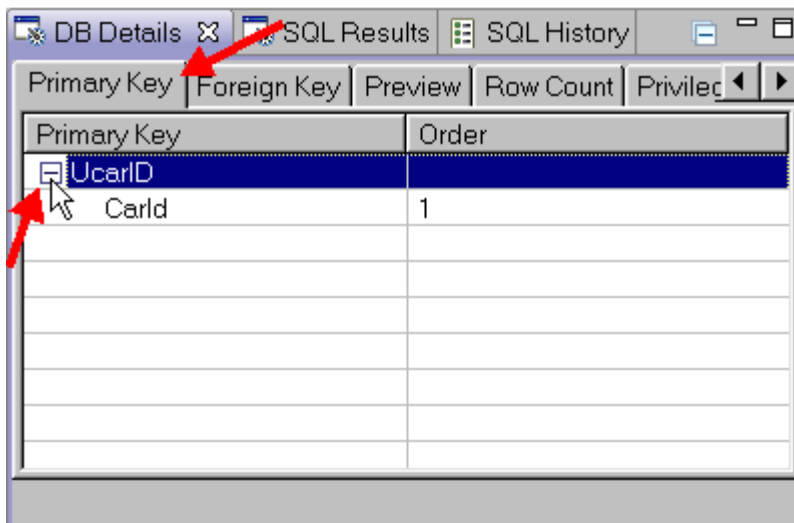
Lab 4 – Using the DB Navigator, continued

Working with the DB Details view, continued

3. Select the **OpenEdge Indexes** tab. All of the indexes for the table are listed. Each index node can be expanded to show the components that makeup the index. Click on the plus sign (+) next to the DealerCar index entry to see its components.



4. Select the **Primary Key** tab and expand the node. The primary key for the Car table is shown.

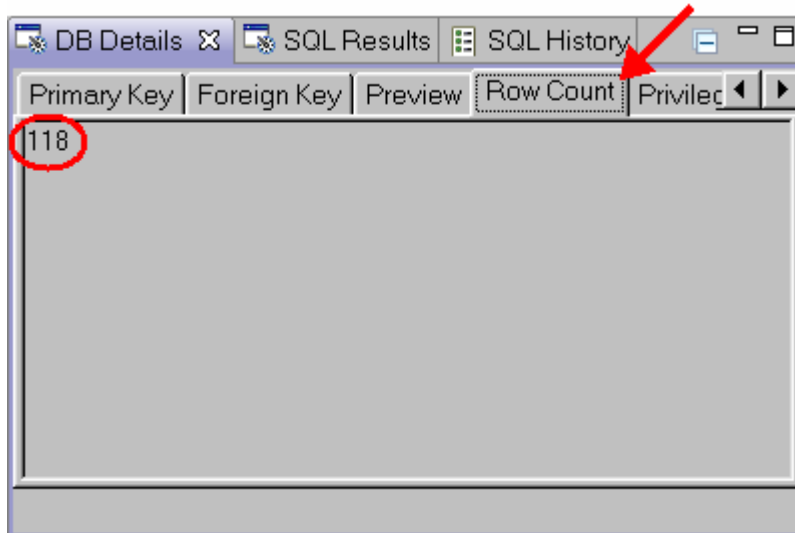


continued on next page

Lab 4 – Using the DB Navigator, continued

Working with the DB Details view, continued

5. Select the **Row Count** tab. This shows the number of rows, or records, in the table. It is the same result that is obtained by running a SQL COUNT command. As with all databases that are accessed from the DB Navigator, the data and metadata shown is obtained from the OpenEdge database via a SQL connection.

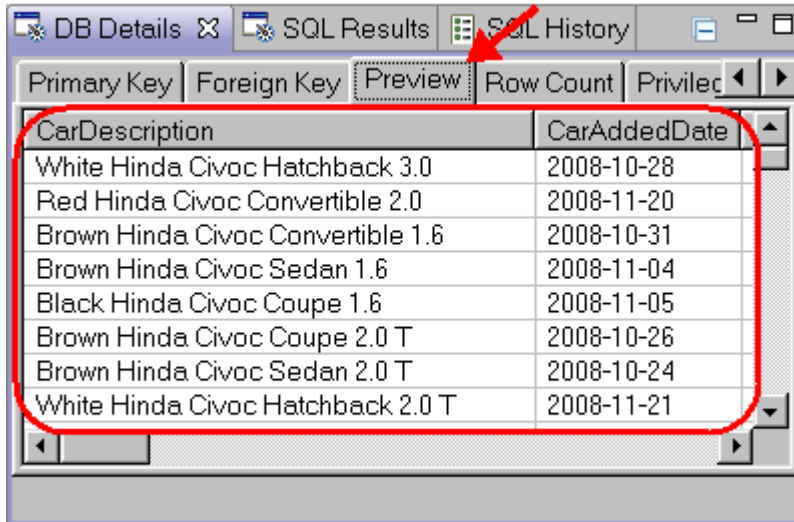



continued on next page


Lab 4 – Using the DB Navigator, continued

Working with the DB Details view, continued

6. Select the **Preview** tab. You will be shown the actual data stored in the database for the Car table.



 **Tip:** The maximum number of preview records returned is the first 150 records for the table. This value is configurable in the Preferences under **Progress DB Navigator** by setting the **Preview row limit**. For this change to take effect, you need to disconnect and reconnect to the database using the Connections view.

 **Note:** So far, we have only seen information from the Car table in the DB Details view. At any time you can return to the DB Structure view and select another available table. The information in the DB Details view will change accordingly.

continued on next page

Lab 4 – Using the DB Navigator, continued

Working with the DB Structure View

This section demonstrates additional features of the DB Structure view.

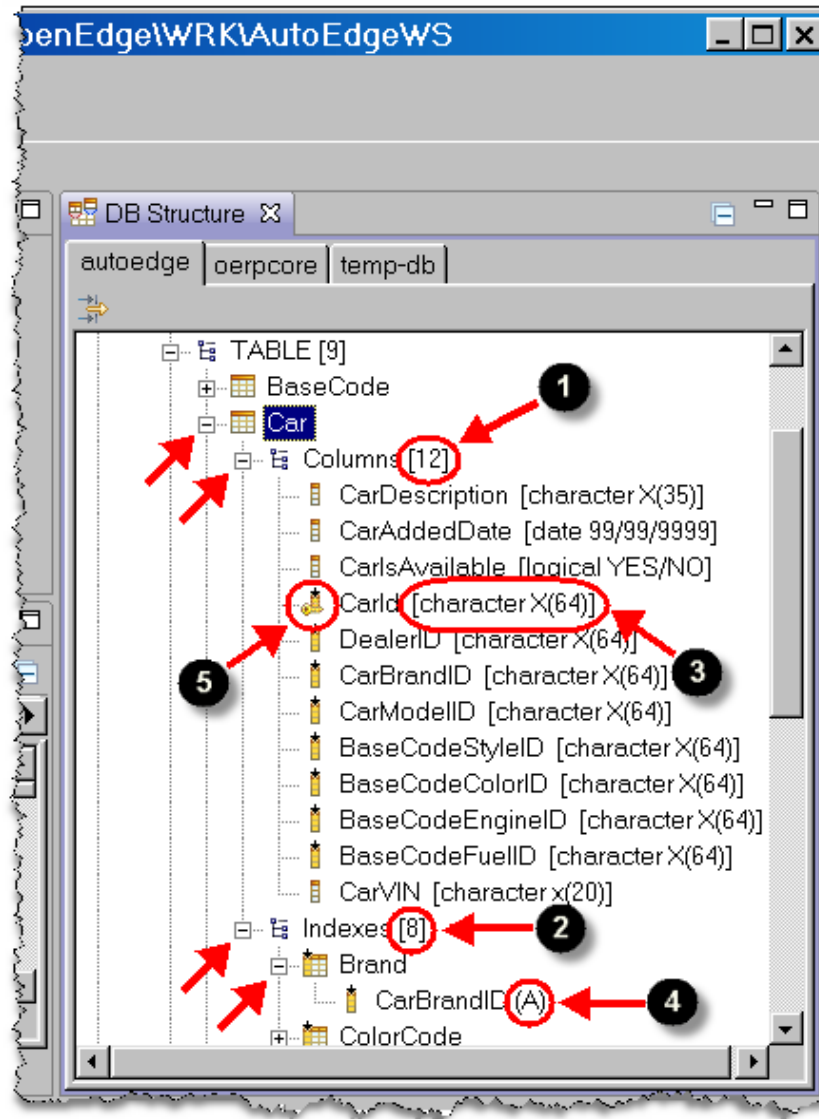
1. Make sure that the **AUTOEDGE**→**TABLE** nodes are expanded in the autoedge tab for the DB Structure view.
2. Expand the **Car** node. The Columns and Indexes nodes are displayed.

continued on next page

Lab 4 – Using the DB Navigator, continued

Working with the DB Structure View, continued

- Expand both the **Columns** and **Indexes** nodes. Also expand the **Brand** index node under Indexes.



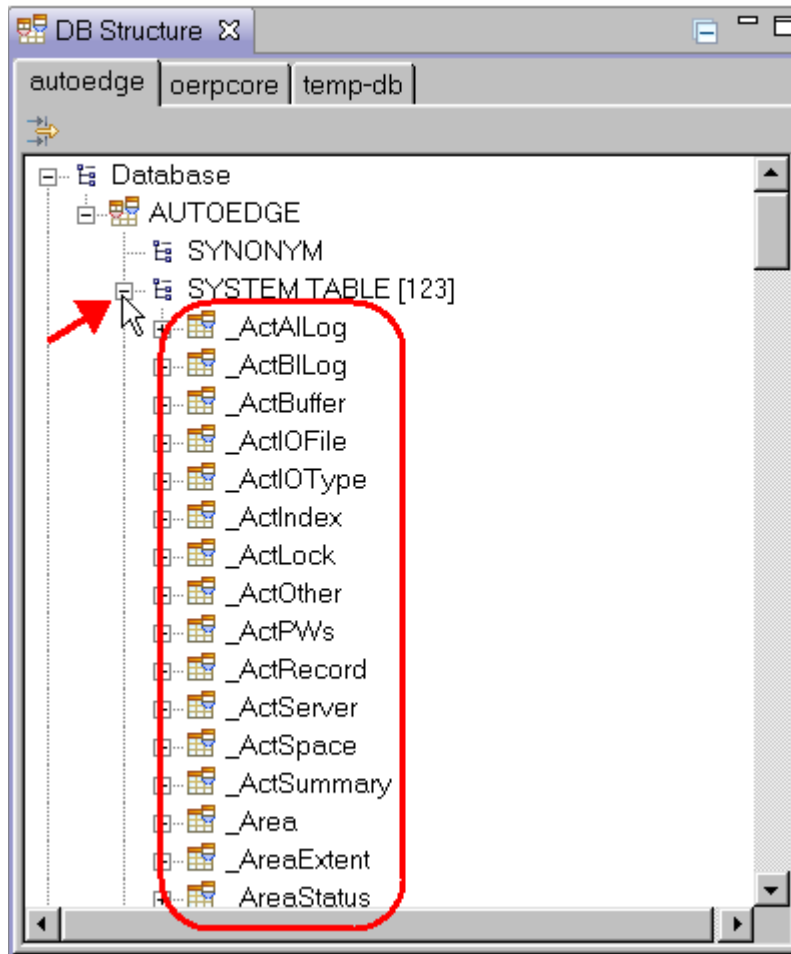
Architect displays the total number of columns (❶) and indexes (❷) next to their respective entries. Other information included in the nodes for quick reference include the data type (❸) for each column and the sorting order (❹), ascending or descending, for each index component. Also note that different icons are used to signify special characteristics, such as the key icon for the CarID column (❺) which signifies that the column is part of the primary key index.

continued on next page

Lab 4 – Using the DB Navigator, continued

Working with the DB Structure View, continued

4. To see the system tables for the OpenEdge RDBMS, expand the **SYSTEM TABLE** node. This is where the underscore tables, such as `_File`, can be found.




continued on next page

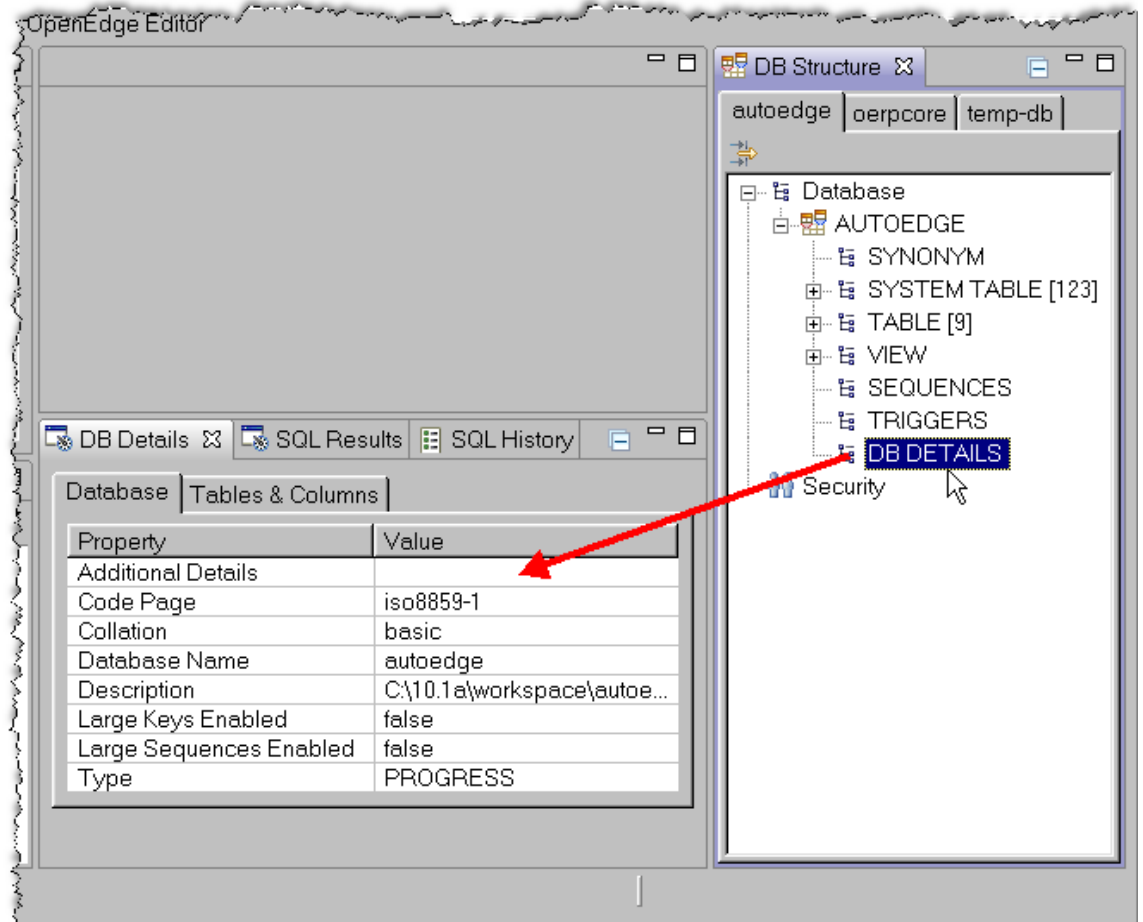
Lab 4 – Using the DB Navigator, continued

Working with the DB Structure View, continued

5. The last node available in the DB Structure view for the AutoEdge catalog is the **DB DETAILS** node. Select this node. You can collapse the other nodes to make it easier to find.

 **Tip:** Don't confuse the DB Details node in the DB Structure view with the DB Details view.

In the **DB Details** view, information on the overall database properties for the selected OpenEdge RDMBS is displayed under the **Database** tab.

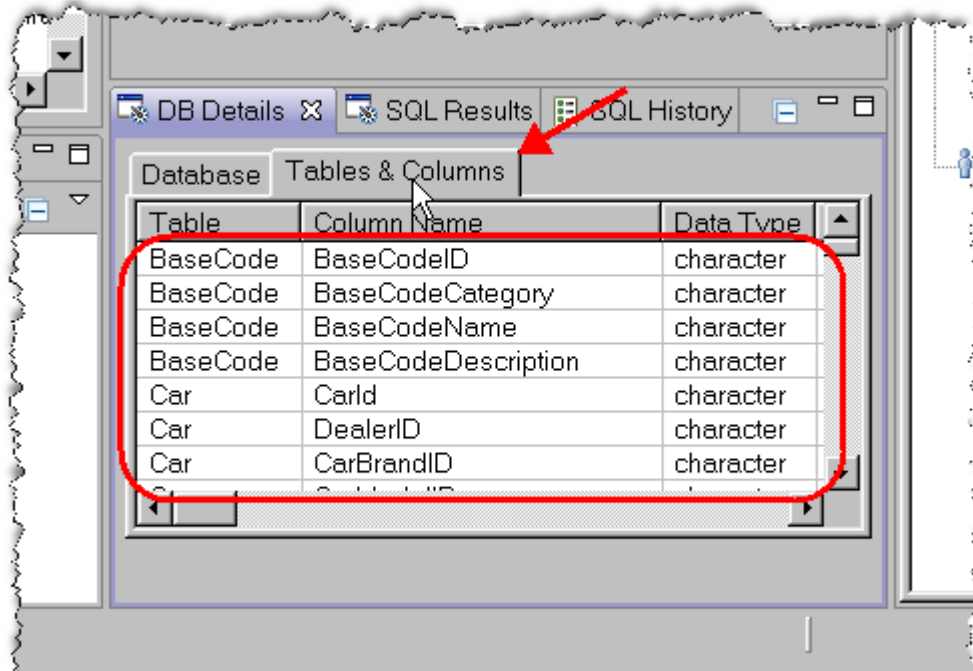



continued on next page

Lab 4 – Using the DB Navigator, continued

Working with the DB Structure View, continued

6. Select the **Tables & Columns** tab. This tab provides an alternative way to view column information for the database and includes all of the columns from all of the tables in the database, as opposed to just those in a single table.



 **Tip:** This tab is useful because it allows you to sort all of the columns in the entire database by any attribute such as data type and format. This makes it easy to confirm that attributes are consistent across similar fields in the database (such as key fields).

Lab 5 – Using DB Navigator’s SQL Editor

Overview

The goal of this lab is to acquaint you with OpenEdge Architect’s SQL Editor.

SQL Editor Basics

This section shows how to use the SQL Editor to write queries against a database.

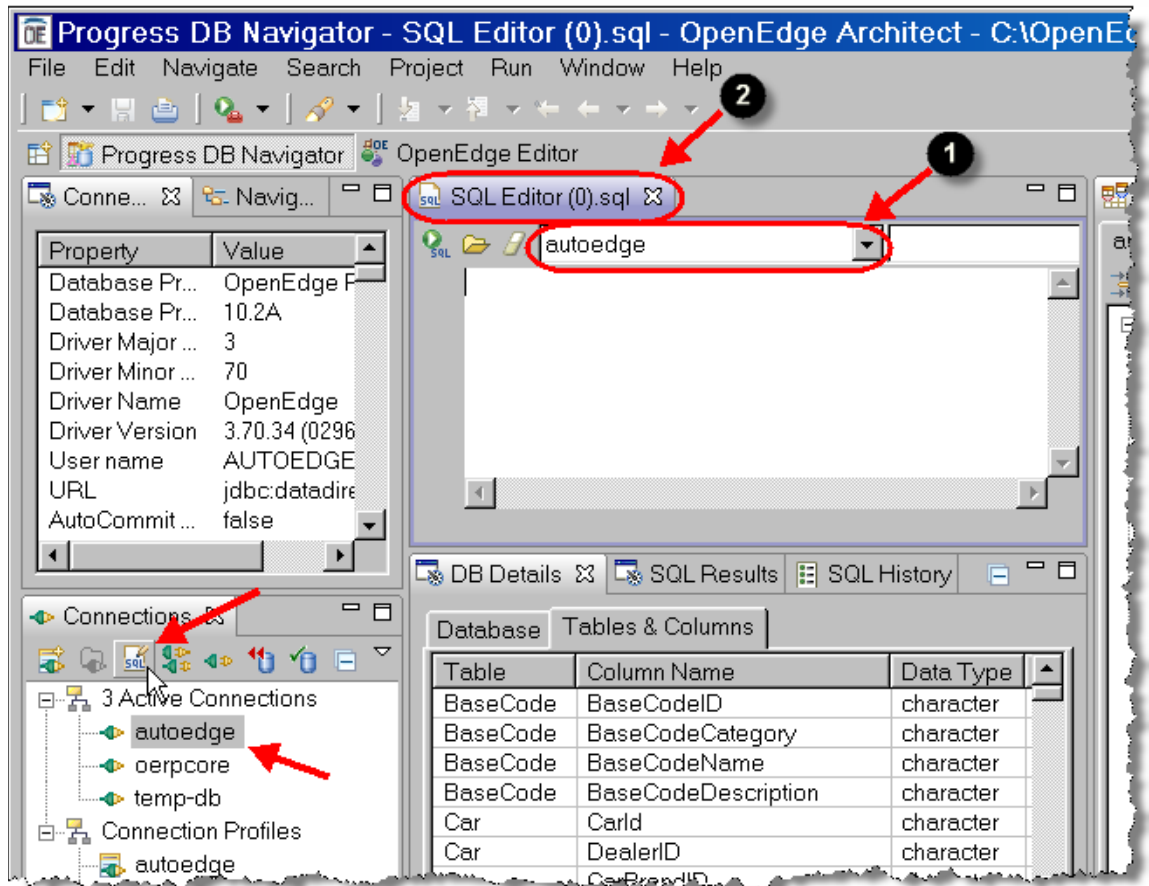
1. Make sure that you are in the DB Navigator perspective.
2. In the Connections view, select the **autoedge** database node under Active Connections.


continued on next page


Lab 5 – Using DB Navigator’s SQL Editor, continued

SQL Editor Basics, continued

3. Choose the **New SQL Editor** button in the Connections view toolbar. A new SQL editor is opened.



 **Tip:** When a new SQL editor is requested, it will default to run queries against the currently selected active database. You can change the database the queries are run against using the database drop-down (❶).

 **Note:** Since you are creating a new temporary SQL file that does not exist on a disk, the name of the file is generated with a number and a .sql extension. The number is unique within the session and is incremented each time a new SQL editor is created. The tab of the SQL Editor includes the generated name of the temporary SQL file (❷)

continued on next page


Lab 5 – Using DB Navigator’s SQL Editor, continued


SQL Editor Basics, continued

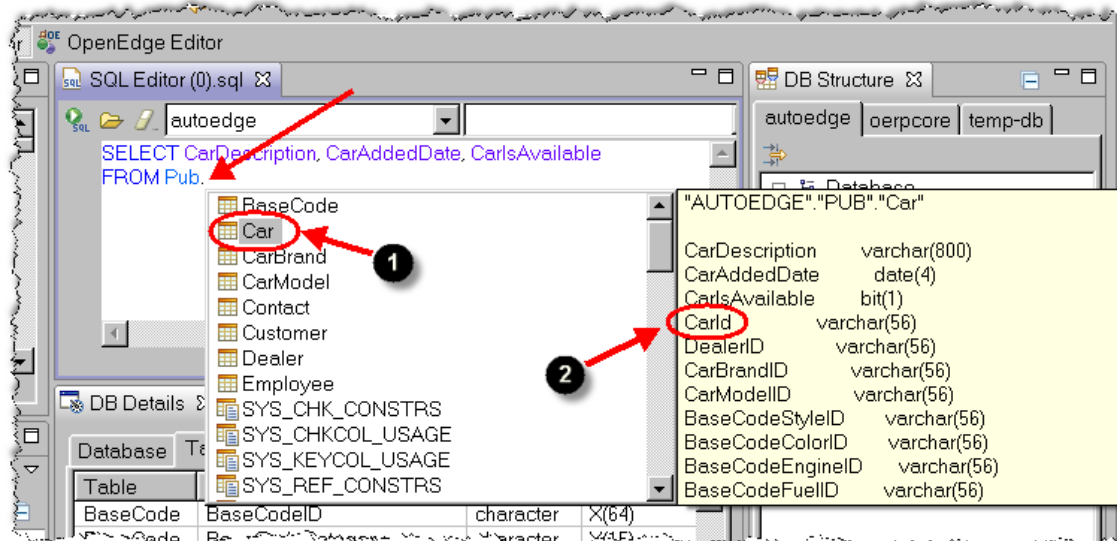
4. Enter the following code in the SQL Editor:

Code for the SQL Editor

```
SELECT CarDescription, CarAddedDate, CarIsAvailable  
FROM Pub.Car;
```

 **Note:** Tables must always be fully qualified in the FROM clause when accessing an OpenEdge database using the SQL interface. So in this case, the **PUB** qualifier is used for the Car table.


 **Note:** If you do not want to type the code, you can copy it from the above box and paste it directly in the editor. However, you might want to become comfortable with Architect’s code completion features before skipping directly typing code. For instance, the SQL editor will provide pop-up help when the dot (.) is typed after **Pub**, allowing you to select from a list of available tables (❶) and columns (❷).

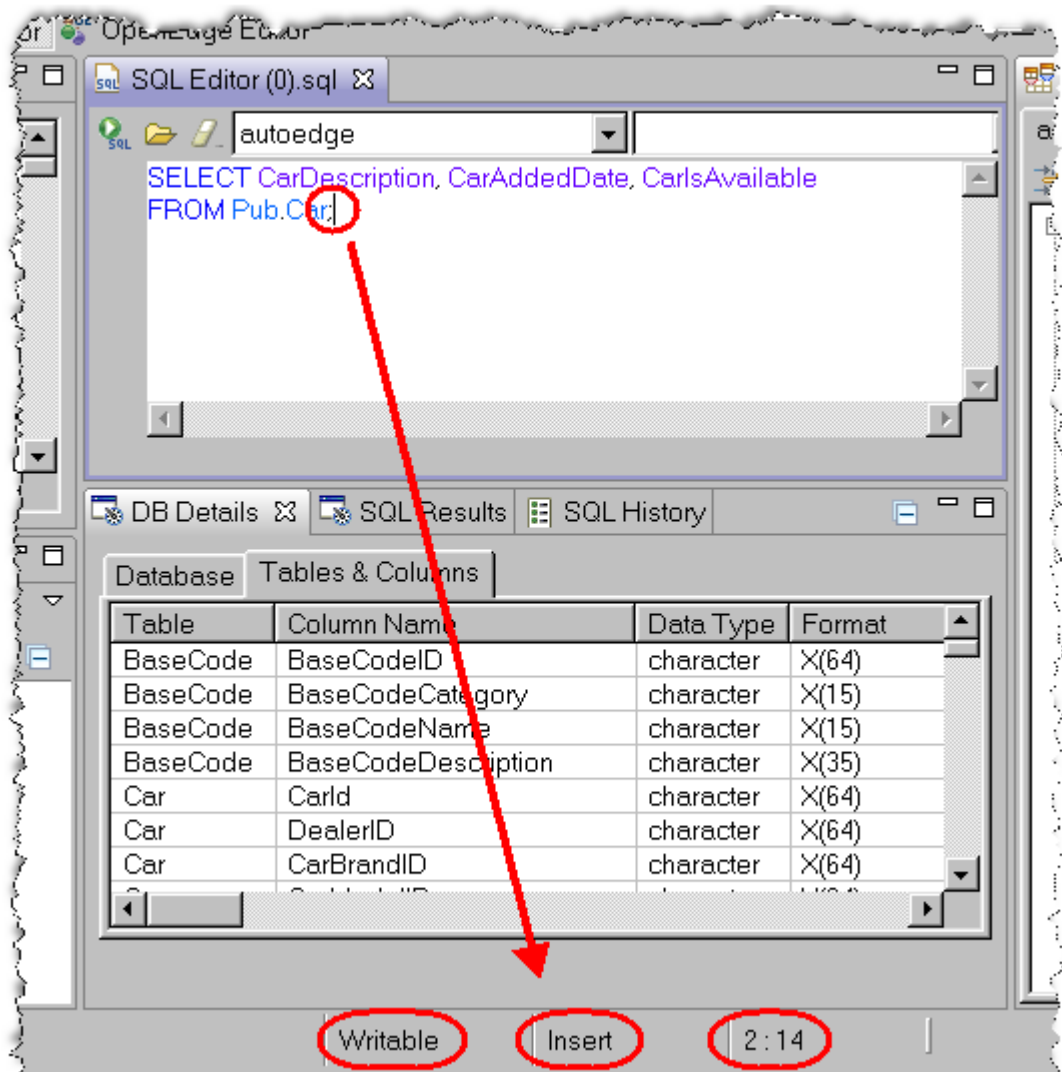


continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

SQL Editor Basics, continued

 **Note:** The SQL Editor shows information about the selected file and cursor location in the status bar area. As shown below, the status bar indicates that the current file is Writable, the current mode is Insert, and the location of the cursor is at row 2 and column 14. Reducing the size of Architect’s window so that it is too small will remove this information.

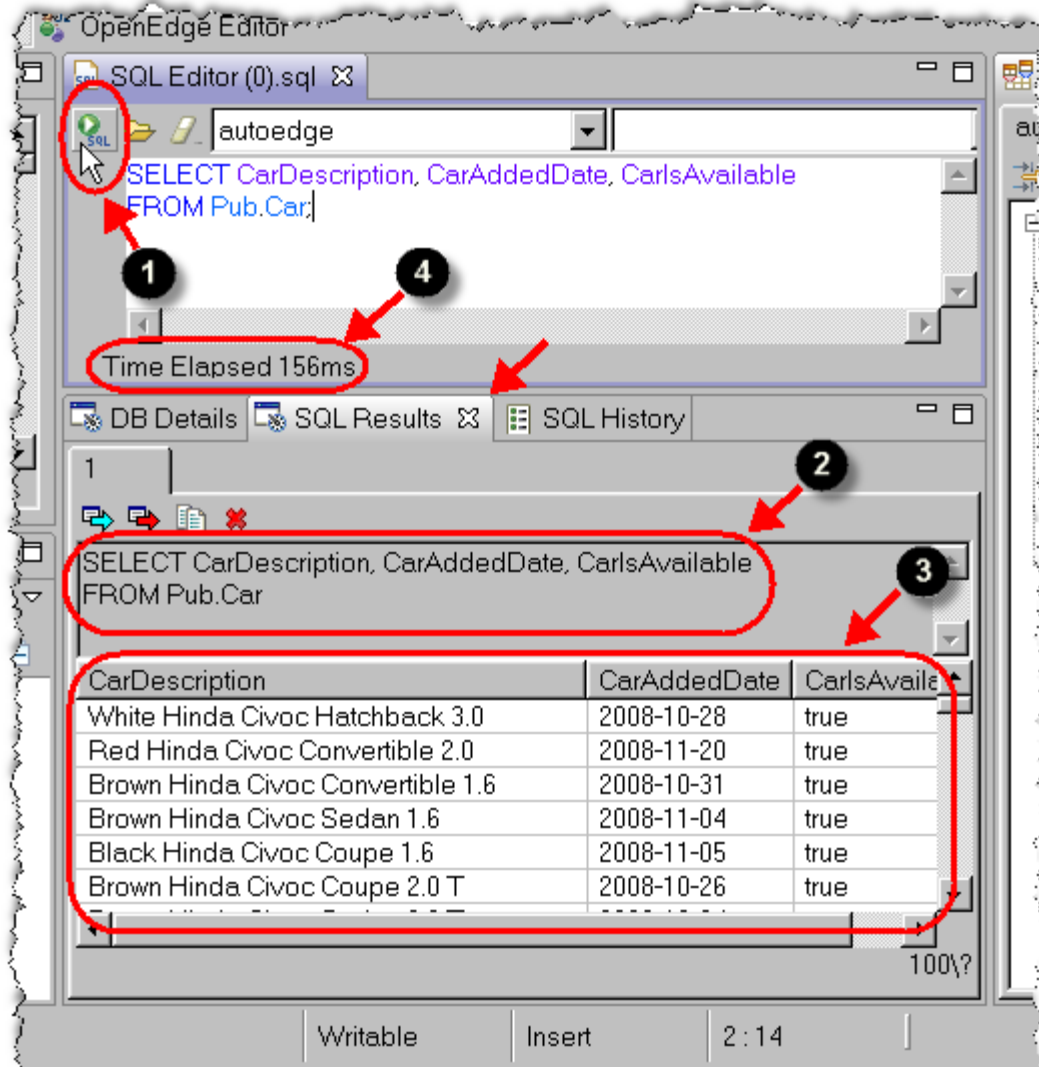


continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

SQL Editor Basics, continued

- Execute the query by pressing the **Execute SQL** (❶) button.




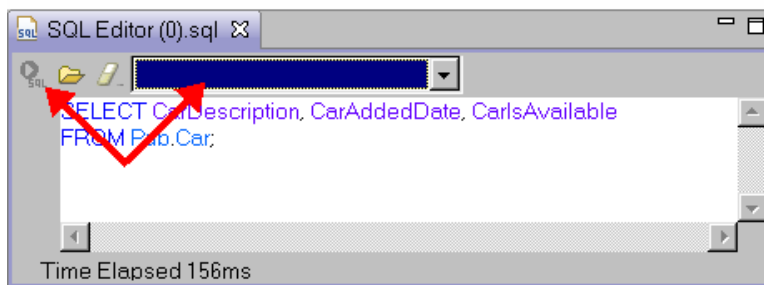
The result is shown in the **SQL Results** view. In the view, you will see the query that was performed (❷) along with the resulting rows from the database (❸), including the columns that were specified in the query. The time that it takes the SQL query to execute is shown in the lower portion of the editor (❹).

continued on next page

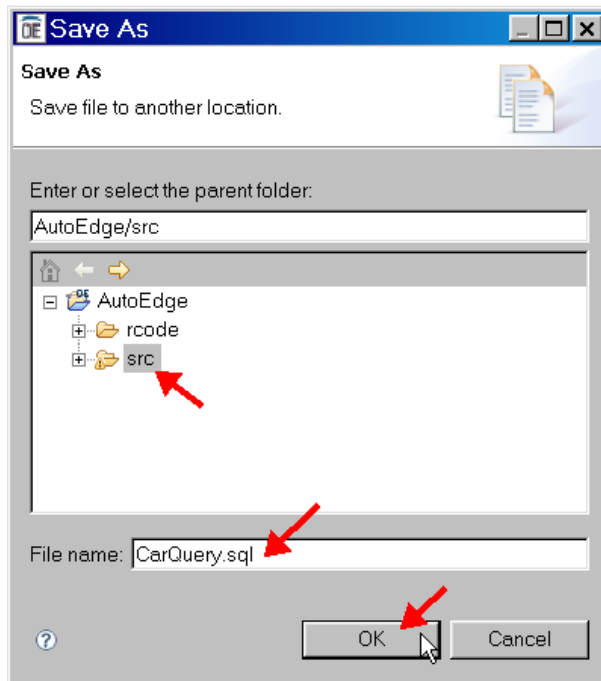
Lab 5 – Using DB Navigator’s SQL Editor, continued

SQL Editor Basics, continued

 **Tip:** Remember that since you selected the AutoEdge database in the Connections view, the SQL editor was opened with the correct database already set. However, if the Execute SQL button is disabled, the likely cause is that a database has not been selected. Select a database from the combo-box in the Editor if one is not specified. This is a common issue when reopening a query, since a SQL file does not retain information on the database that was connected.



6. From Architect’s menu, select **File**→**Save As**. The Save As dialog is displayed.
7. Expand the **AutoEdge** node and select the **src** folder. Enter **CarQuery.sql** for the File Name. Click **OK** to save the file.

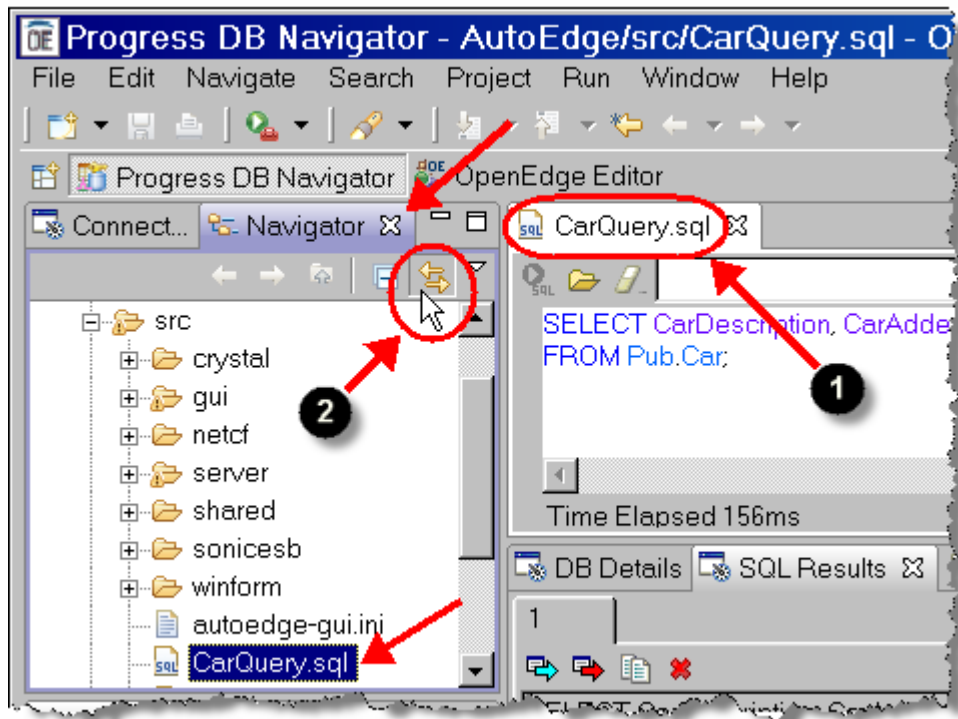



continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

SQL Editor Basics, continued

8. After saving the file, the SQL Editor’s tab changes to reflect the new name for the file (❶).



 **Tip:** The SQL program is now part of the project and can be seen in the Navigator view by clicking on the **Link with Editor** (❷) button.

continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Generating SQL code using the DB Structure view

You can use the DB Structure view to help with generating SQL code. This is especially useful because instead of manually typing table, column, and field names, you can copy the information directly from the DB Structure view. The following steps show how to use these features.

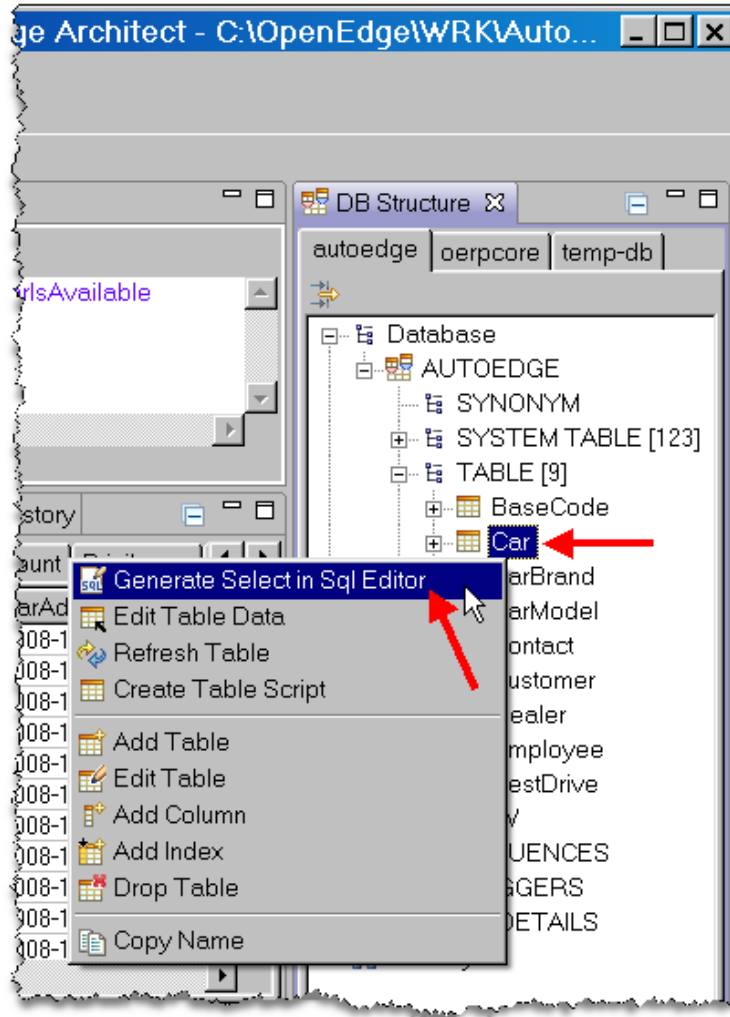
1. Select the **Database→AutoEdge→TABLE→Car** node in the **autoedge** tab of the DB Structure view.

continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Generating SQL code using the DB Structure view, continued

2. Right-click on the **Car** table and select **Generate Select in Sql Editor**.

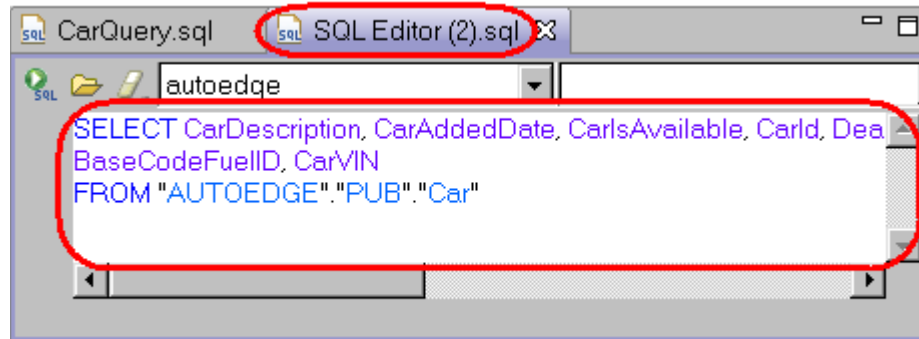


continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Generating SQL code using the DB Structure view, continued

A new SQL Editor is opened and a SQL SELECT statement is generated to access the records in the Car table. Not all fields are visible because by default generated SQL has 10 fields per line.

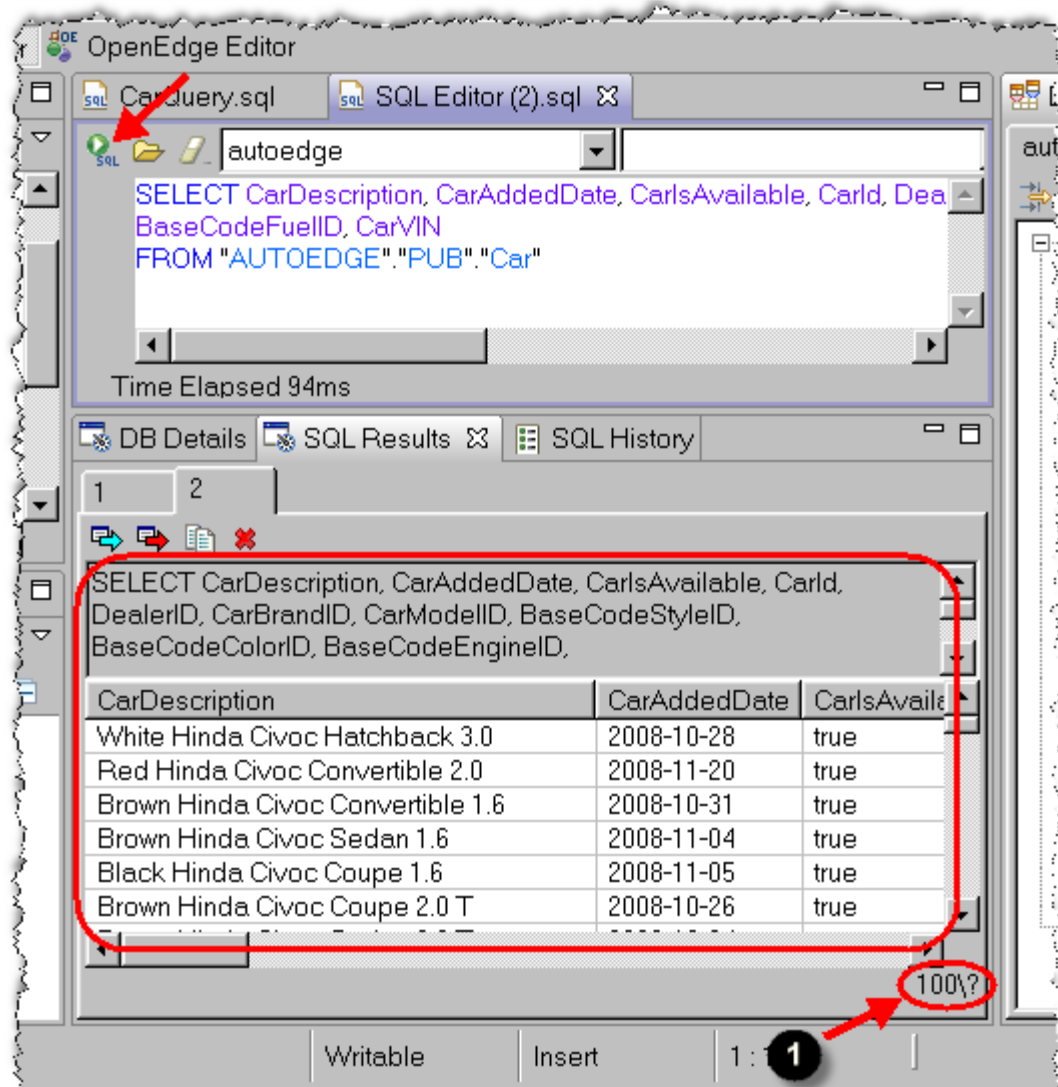


continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Generating SQL code using the DB Structure view, continued

3. Choose the **Execute SQL** button to run the code.



The **SQL Results** view shows the results of the executed SQL statement. The lower right corner shows the number of records returned by the query. In this case, only the first 100 records were returned.

4. If you wish, you can save this query as **GeneratedCarQuery.sql** in the AutoEdge project under the src directory.

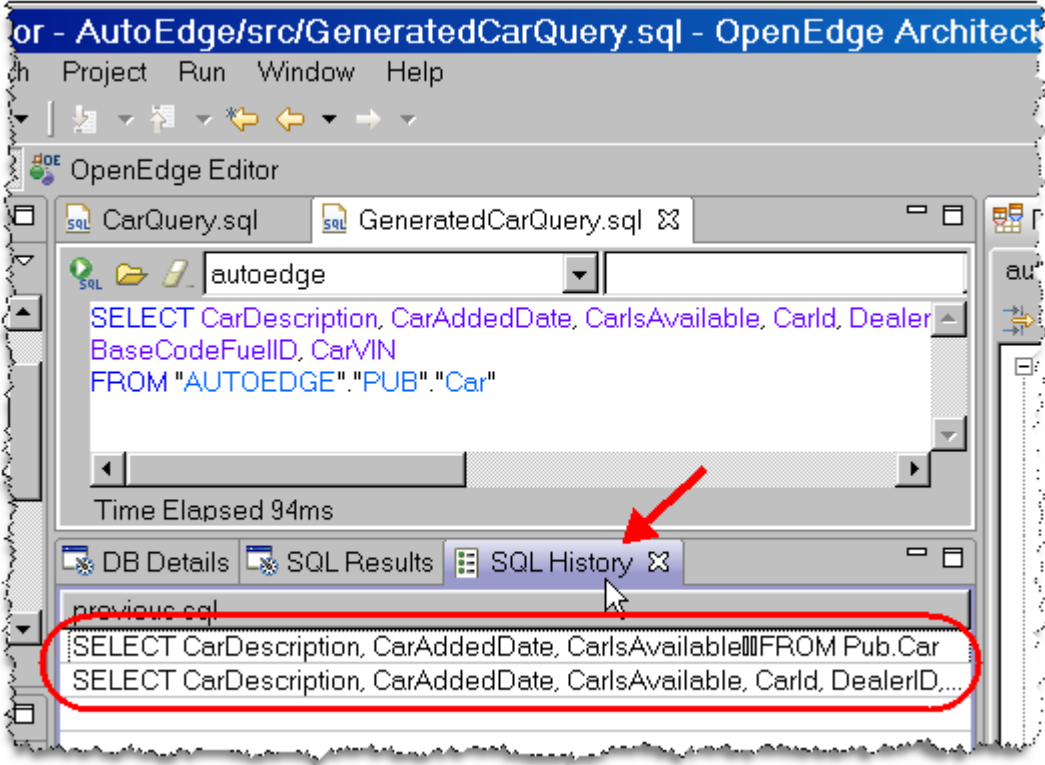
continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Using the SQL History view

The SQL History view contains one entry for each unique query that has been run during the session. This allows you to recall previous queries in case you need to run a query again but do not want to load a saved file.

1. Select the **SQL History** view.



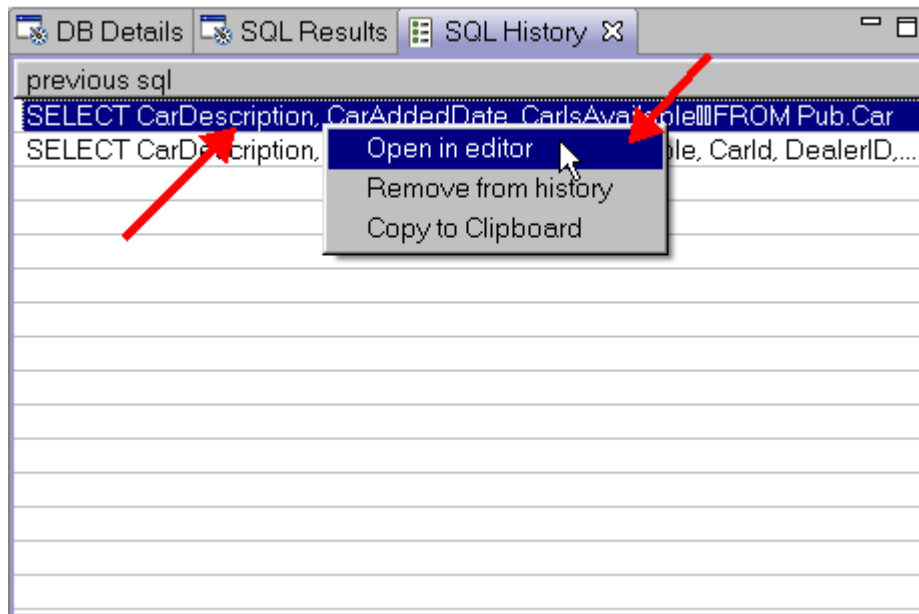
There should be at least two entries for the two queries that you have run in previous steps. Note that there will be one entry for each unique query, not entries for each time you have run a query.

continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Using the SQL History view, continued

2. Right-click on the entry that only has the three Car fields in the query and choose **Open in editor**.

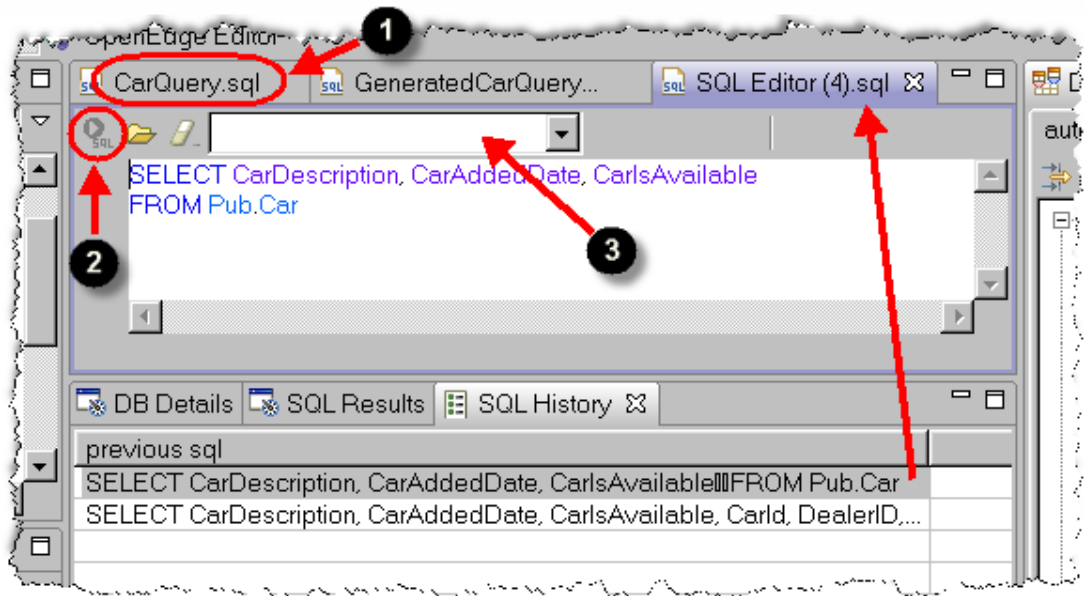


continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Using the SQL History view, continued

3. The query is opened in the SQL Editor. Notice that it didn't open the saved SQL program (❶), but instead the code was opened in a new SQL Editor tab. The Execute SQL button (❷), is disabled, since a database (❸) is not currently associated with the query.



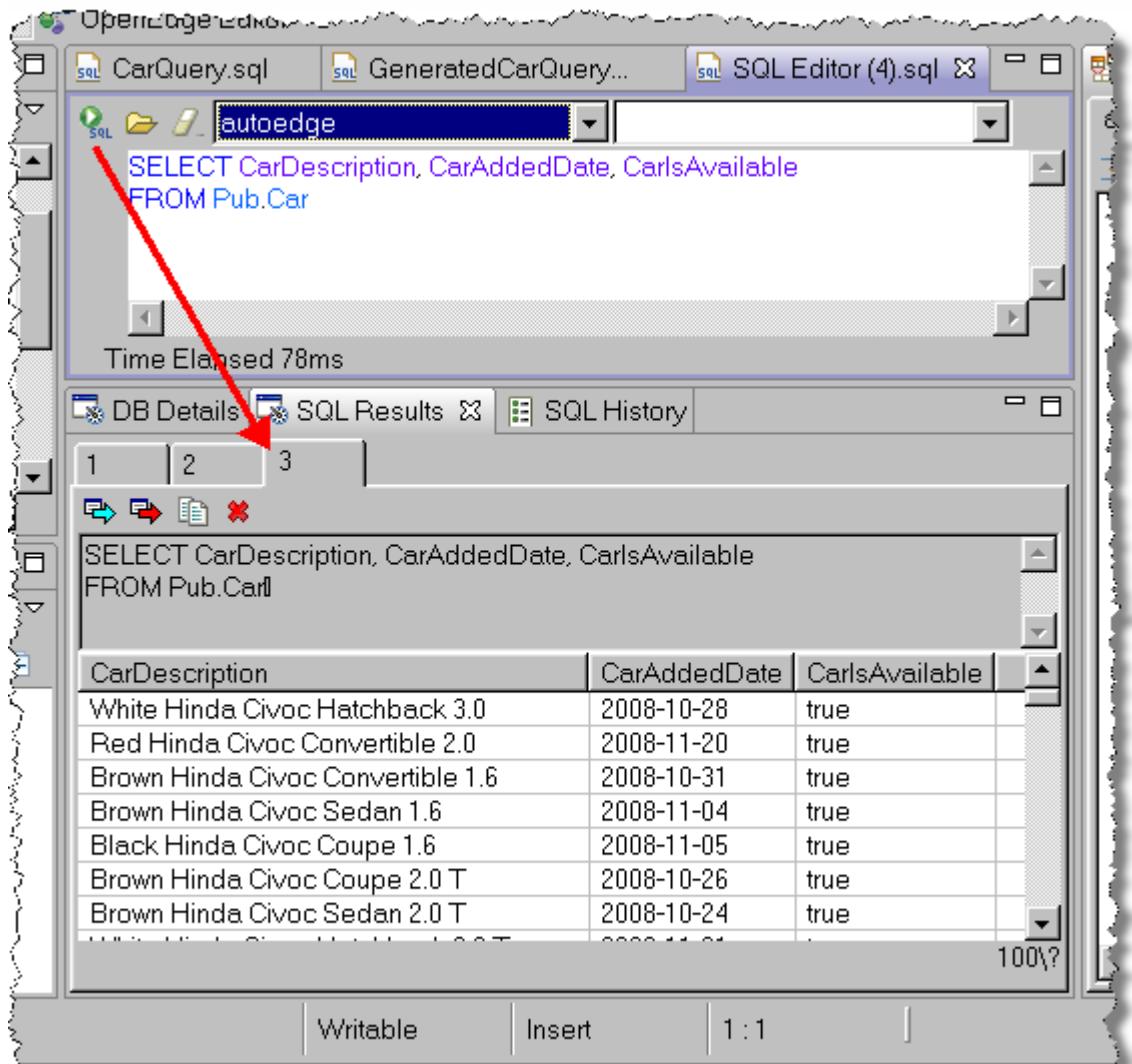
4. Associate the **autoedge** database with the query by selecting it from the Choose Connection drop-down.

continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Using the SQL History view, continued

5. Execute the query.



A new tab folder is created in the SQL Results view showing the results.

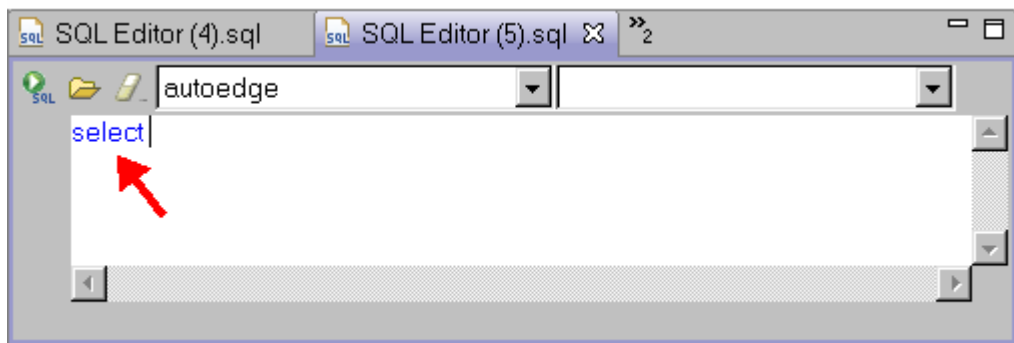
continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Copying object names from the DB Structure view

In addition to generating a complete select statement from the DB Structure view, you can copy individual database object names from the DB Structure view to the SQL Editor.

1. Open a new SQL editor by selecting the **New SQL Editor** button in the Connections view.
2. Type **select** in the editor.



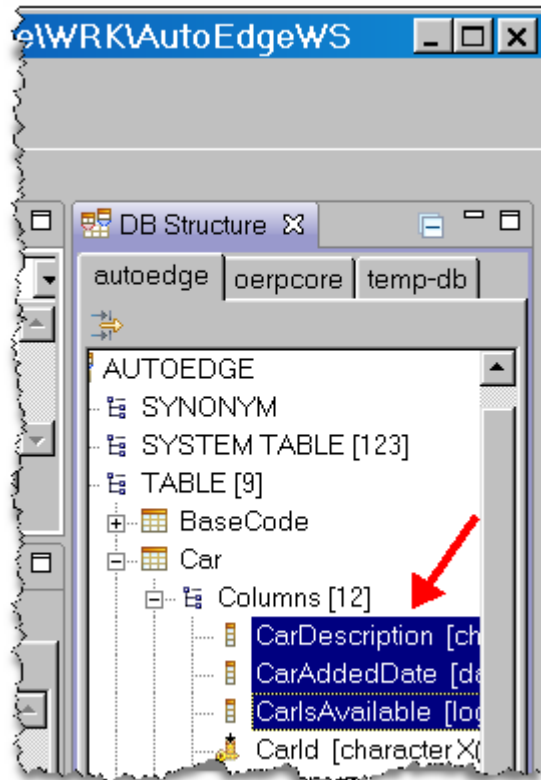
3. Expand the **Database**→**AUTOEDGE**→**TABLE**→**Car**→**Columns** nodes in the DB Structure view.

continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Copying object names from the DB Structure view, continued

4. Perform multiple selections by holding down the **Ctrl** key and selecting the **CarDescription**, **CarAddedDate**, and **CarIsAvailable** columns.

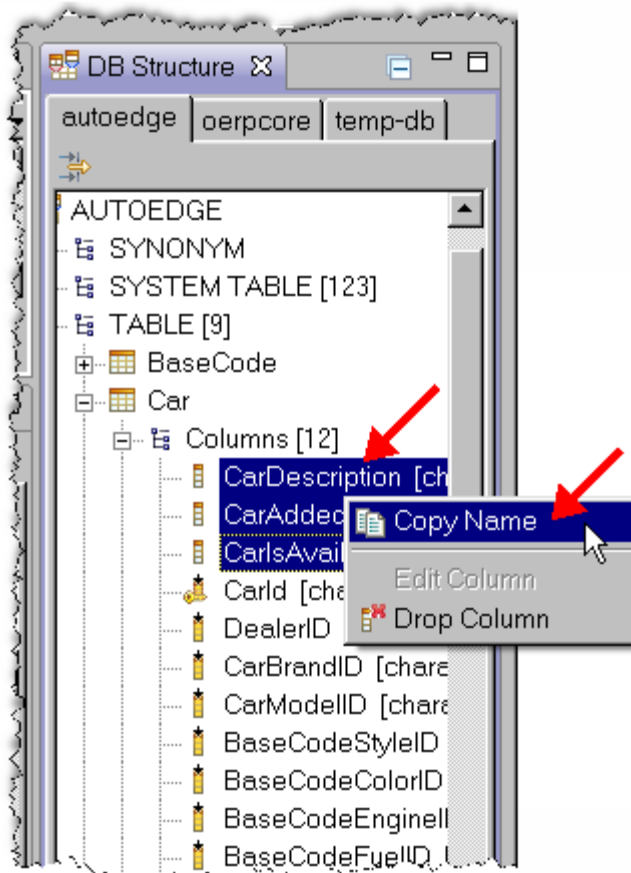


continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Copying object names from the DB Structure view, continued

5. Right-click on any of the highlighted columns and select **Copy Name**.

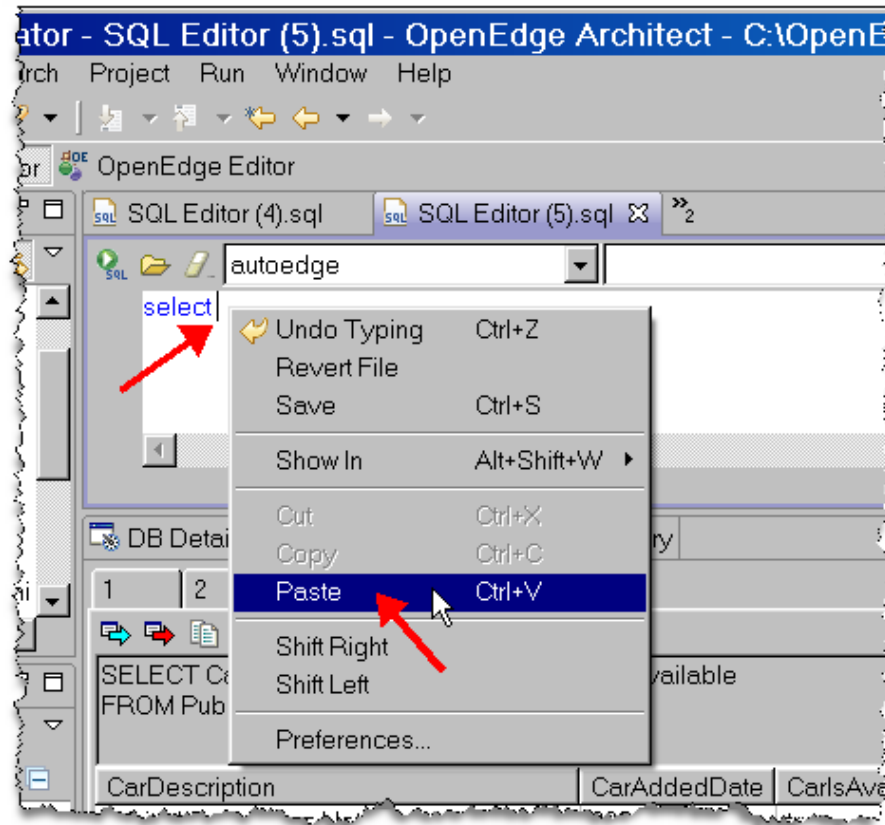


continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

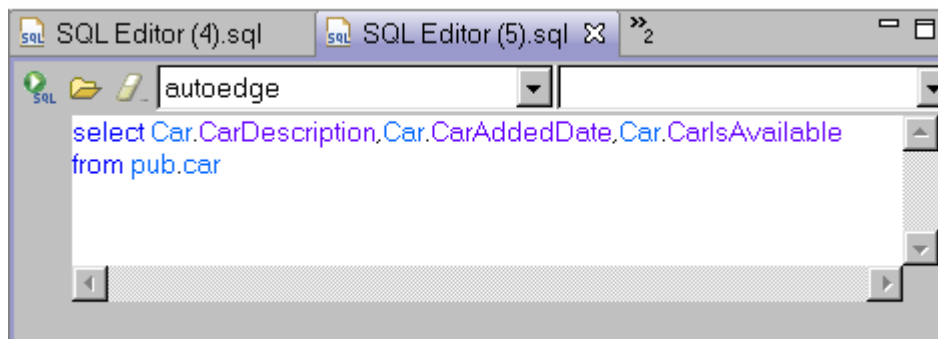
Copying object names from the DB Structure view, continued

6. Place the cursor a space away from the **select** statement in the SQL Editor, right-click, and select **paste**.



The selected fields are added to the SQL Editor.

7. Press the **Enter** key and type **from pub.car**.



continued on next page

Lab 5 – Using DB Navigator’s SQL Editor, continued

Copying object names from the DB Structure view, continued

8. Execute the query. The results are displayed in the SQL Result view.
 9. Close all the open SQL Editors.
-

Lab 6 – Modifying Schema using the DB Navigator

Overview

This lab shows how to modify OpenEdge RDBMS schema using the DB Navigator.

Adding a new table

In this section, steps are provided that show how to add a new table to the database.

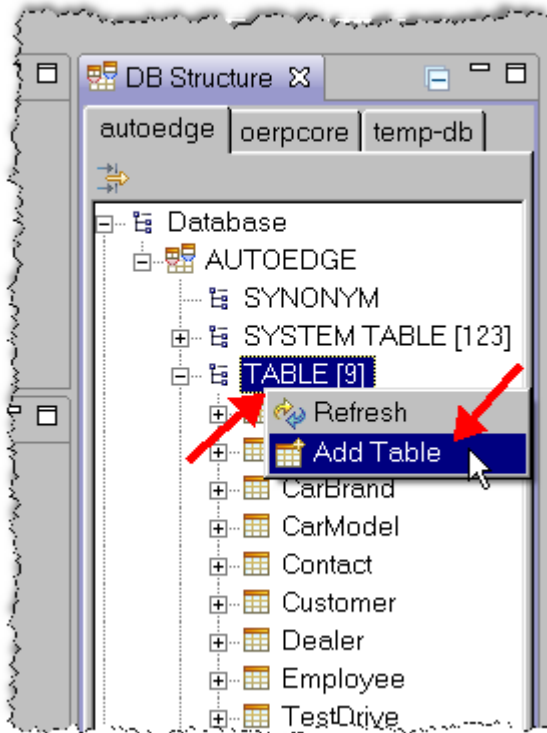
1. Make sure the **autoEdge** entry is selecting in the Active Connections list in the Connections view.
2. In the DB Structure view, select the **autoedge** tab.
3. Expand the **Database→AUTOEDGE→TABLE** nodes.


continued on next page

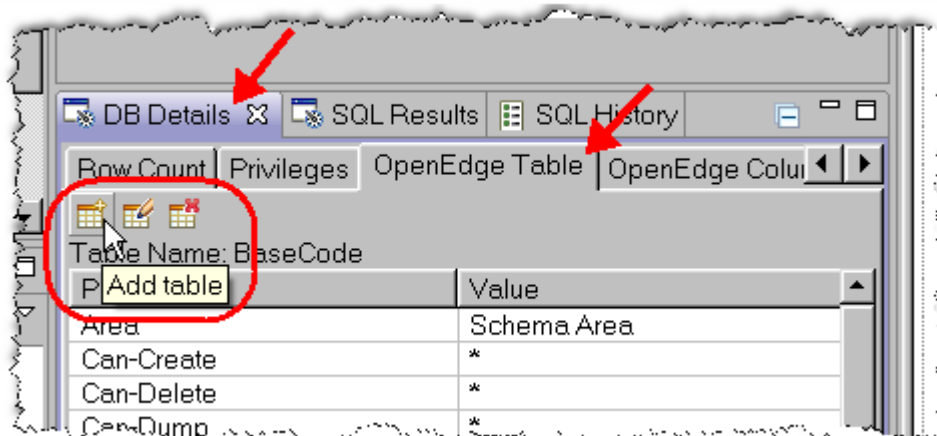
Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new table, continued

4. Right-click on the **TABLE** node, and then select **Add Table**.



 **Tip:** You can also use the toolbar in the OpenEdge Table tab of the DB Details view to add a table.



continued on next page

Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new table, continued

5. In the Add Table dialog enter the following (be sure to maintain case sensitivity):
 - Table Name: **NewCar**
 - Area: **Auditing Area**
 - Description: **Table created for lab.**

The screenshot shows the 'Add Table' dialog box. The 'Table Name' field contains 'NewCar', the 'Area' dropdown is set to 'Auditing Area', and the 'Description' text area contains 'Table created for Lab'. The 'Next >' button is highlighted with a mouse cursor and a red arrow.

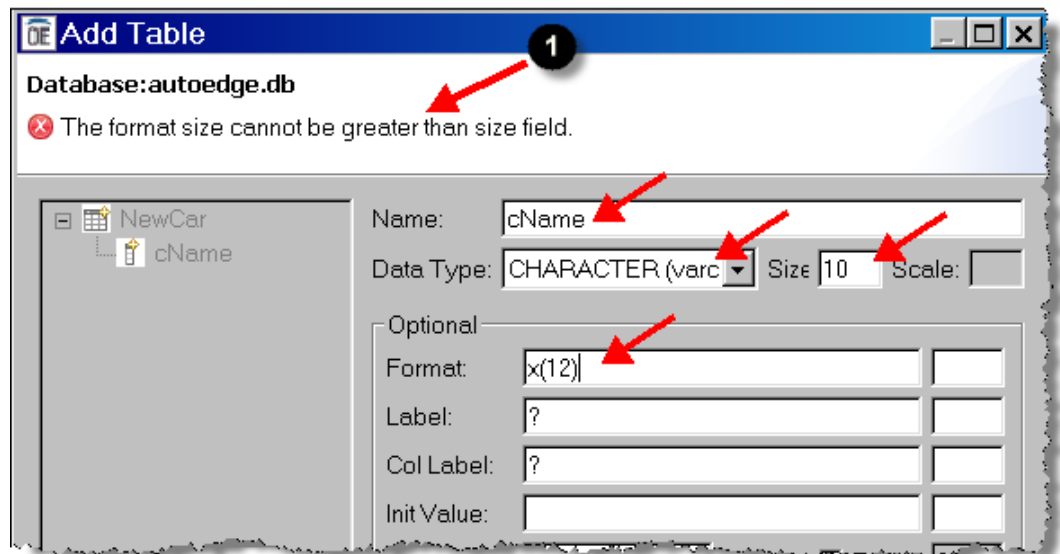
6. Click **Next**. The next page of the Add Table wizard is displayed which allows you to enter columns for the table.

continued on next page

Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new table, continued

7. Add the following column information:
 - Name: **cName**
 - Data Type: **Character**
 - Size: **10**
 - Format: **x(12)**



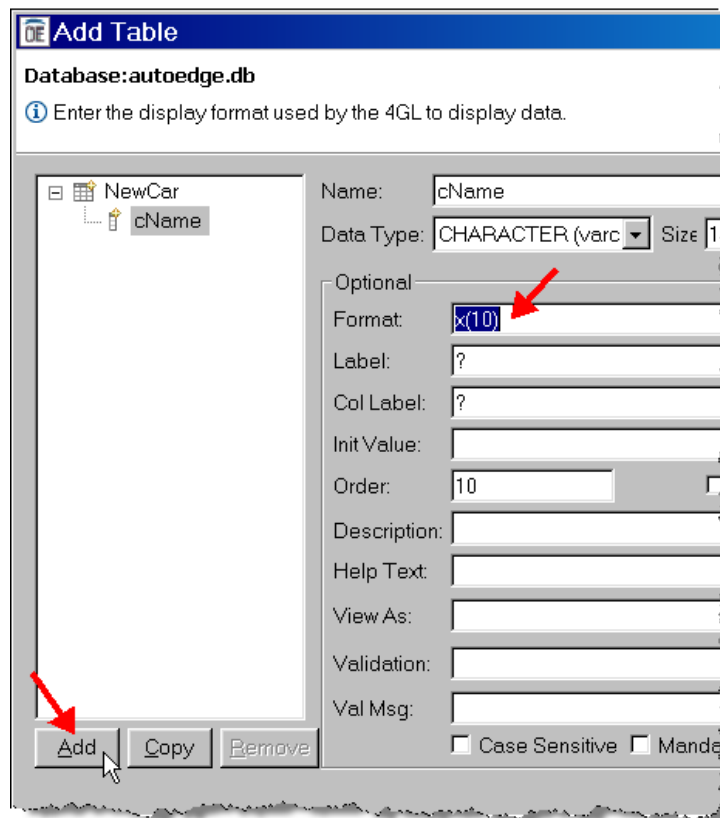
An error message will be displayed in the message area (❶). This message area is a common feature among some of Architect's wizards and either shows context help for the field that the cursor is in or an error/warning message, depending on the information that is entered for a field. This error message (**The format size cannot be greater than the size field**) indicates that the format field size is greater than the fixed SQL field size that you set (currently set to 10). This check helps to insure that the SQL field is large enough to hold the number of characters that can be entered for a field as specified by its format.

continued on next page

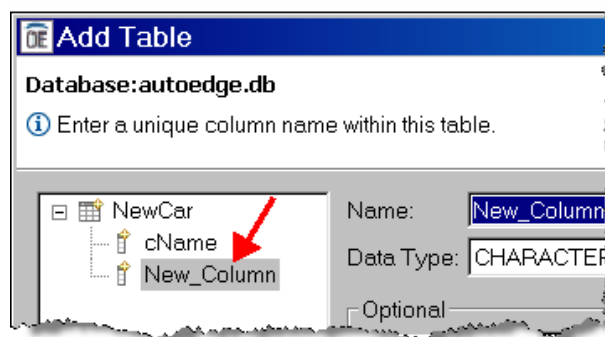
Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new table, continued

8. Change the Format to **x(10)**. The error message will disappear.
9. Click the **Add** button to add another column to the table.



A New_Column label will be placed under the newly added cName column under the NewCar table node. This indicates that the wizard is ready for you to add a new field.



Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new table, continued

10. Add the following two columns. Make sure to click the **Add** button as each column is entered.

- Name: **cDescription**
- Data Type: **Character**
- Size: **35**
- Format: **x(35)**

- Name: **dAdded**
- Data Type: **Date**
- Format: **99/99/9999**

continued on next page

Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new table, continued

11. Add a column with the following information:

- Name: **aClob**
- Data Type: **CLOB**
- Area: **Auditing Area**
- Code Page: **UTF-8**
- Collation: **BASIC**

The screenshot shows the 'Add Table' wizard interface. The 'Name' field contains 'aClob'. The 'Data Type' dropdown is set to 'CLOB (clob)'. The 'Area' dropdown is set to 'Auditing Area'. The 'Code Page' dropdown is set to 'UTF-8'. The 'Collation' dropdown is set to 'BASIC'. The 'Next >' button is highlighted with a red arrow.

12. Click **Next**. The next page of the Add Table wizard is displayed which allows you to enter indexes for the new table.

continued on next page

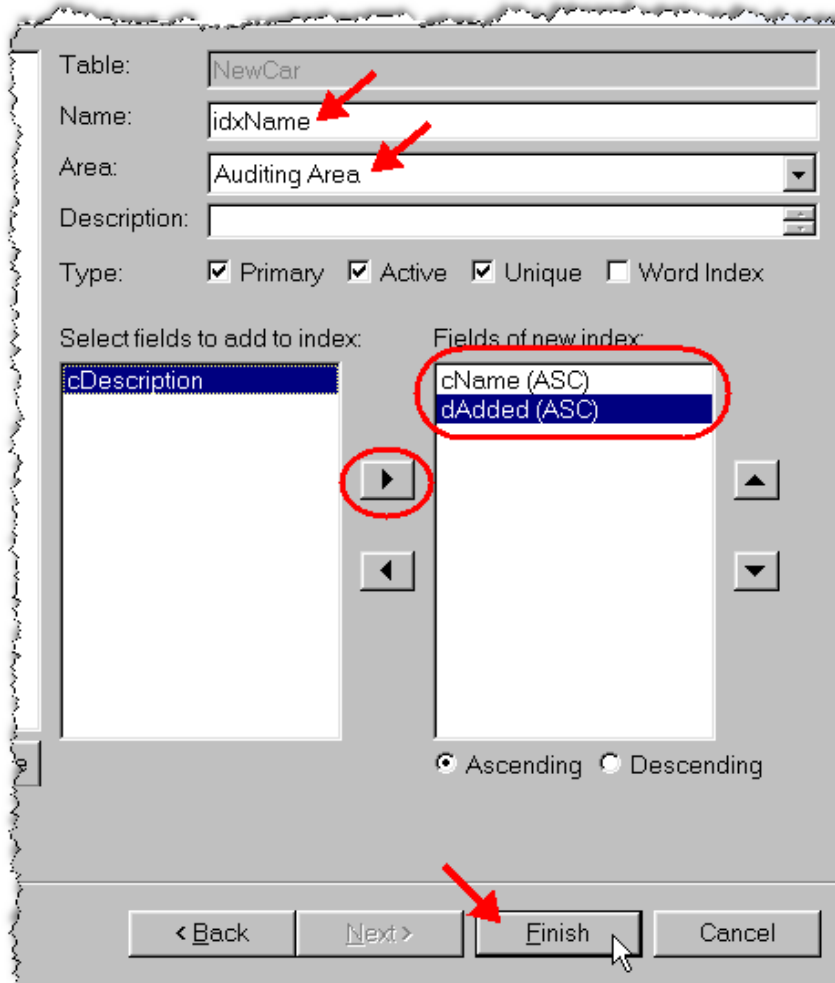
Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new table, continued

13. Enter the following for the new index:

- Name: **idxName**
- Area: **Auditing Area**
- Primary: **Checked**
- Active: **Checked**
- Unique: **Checked**

14. Select the **cName** field in the Select fields list and click the Arrow button. Repeat the process for the **dAdded** field.

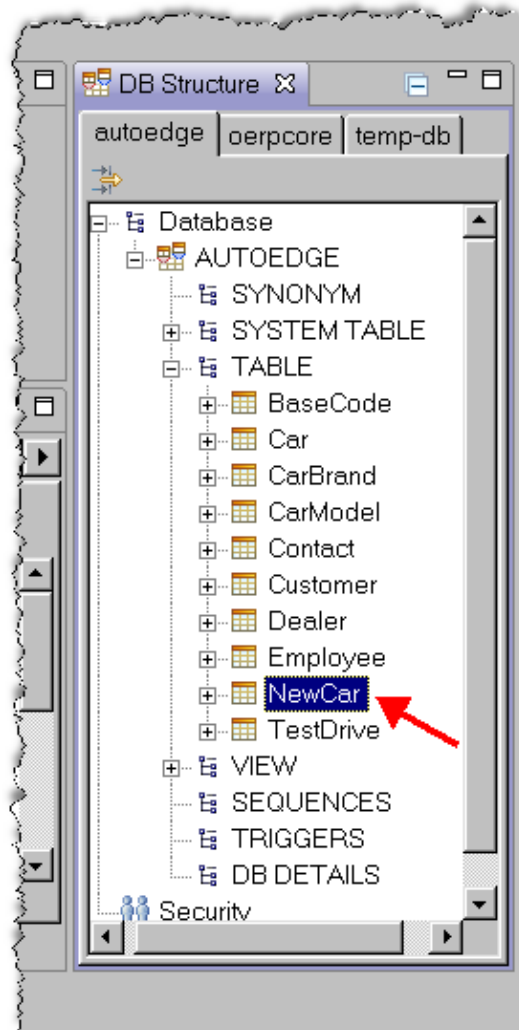



continued on next page


Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new table, continued

15. Choose the **Finish** button to complete the Add Table wizard. The table is created and is displayed in the DB Structure view.



 **Note:** The DB Structure view is sorted in ASCII order.

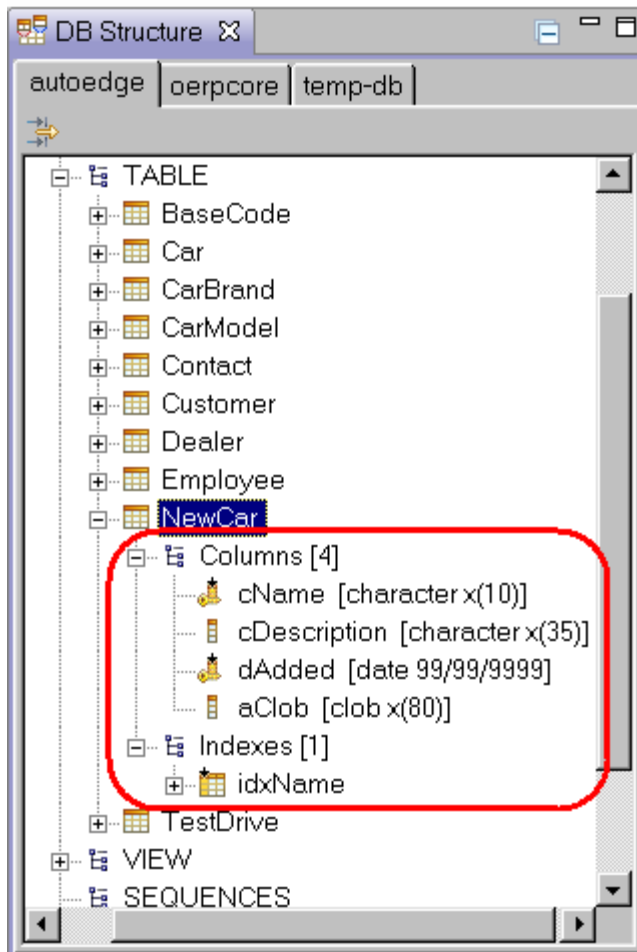
 **Tip:** The DB Structure view is generally refreshed automatically when making schema changes. However, if you think that the view has not been refreshed, you can right-click on the Database node and select Refresh to refresh the nodes.

continued on next page

Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new table, continued

16. Expand the **NewCar** node to see the columns and indexes that you defined.



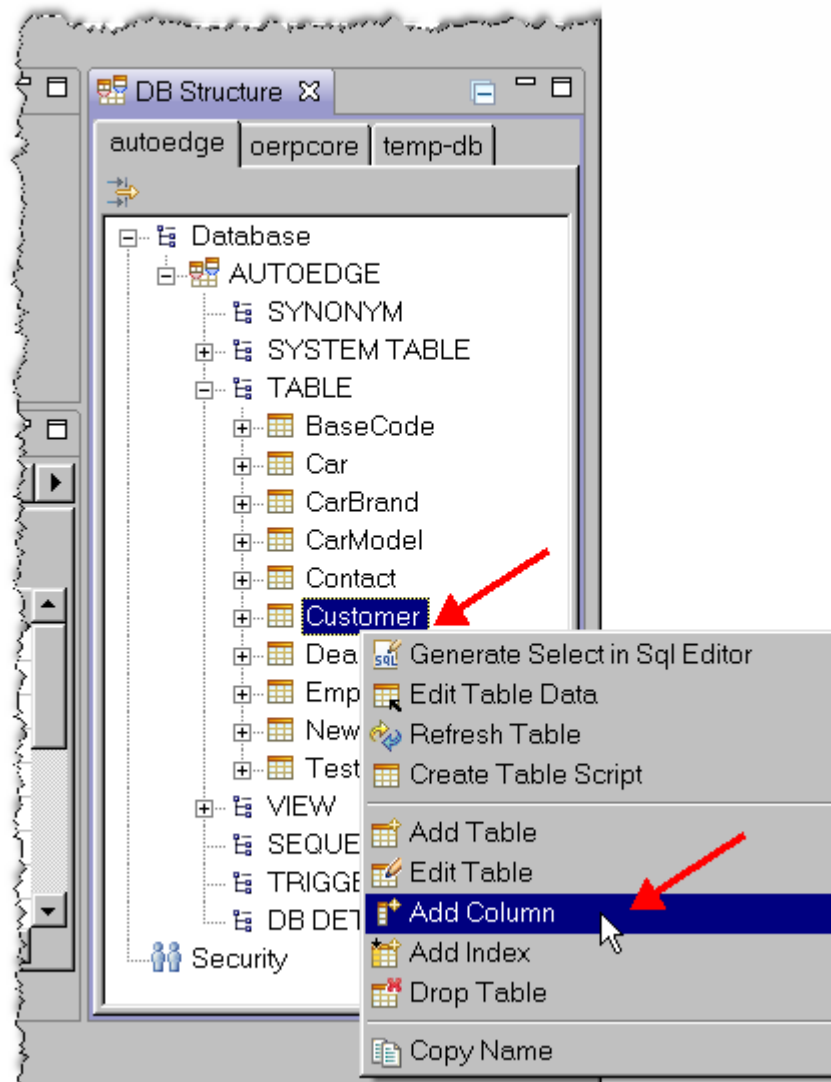
continued on next page

Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new column to the existing Table

The steps in this section show how to add a new column to an existing table in the database.

1. In the DB Structure view, select the **autoedge** tab.
2. Expand the **Database**→**AUTOEDGE**→**TABLE** nodes.
3. Right-click on the **Customer** table and select **Add Column**.



continued on next page

Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new column to the existing Table, continued

- In the Add Column window, enter the following details:
 - Name: **EmailAddress**
 - Data Type: **CHARACTER(varchar)**
 - Size: **100**
 - Format: **x(50)**
 - Label: **Email**
 - Help Text: **Please enter your full Email address**

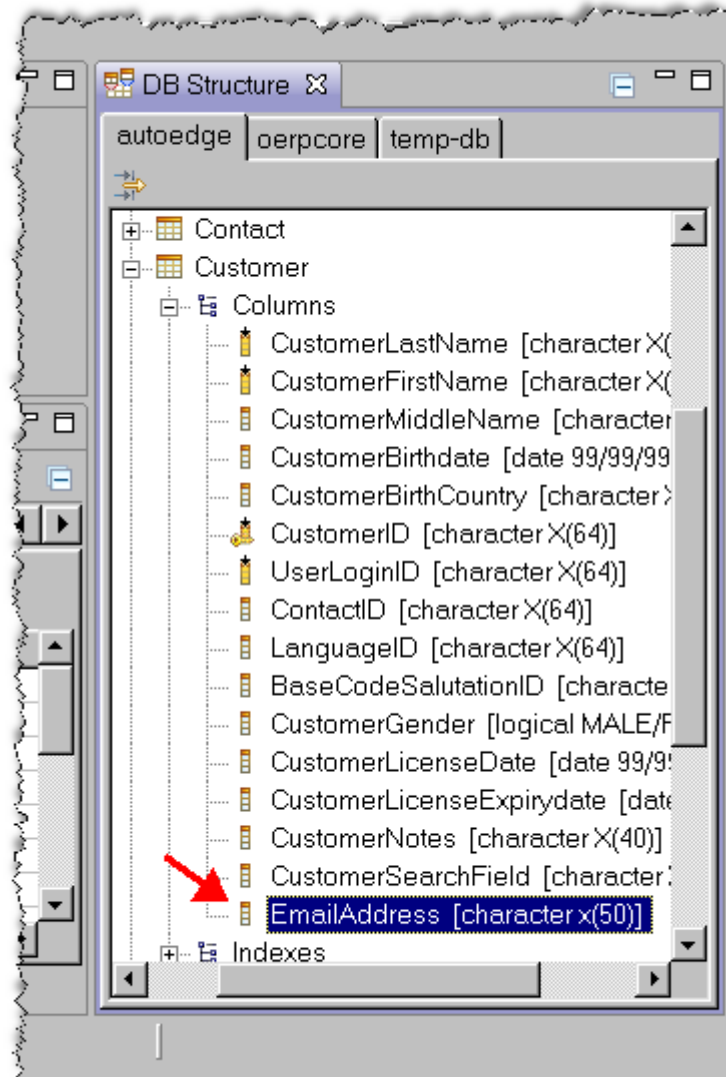
The screenshot shows the 'Add Column' dialog box for a database named 'autoedge.db'. The dialog is divided into two main sections. On the left is a list of existing columns in the table, including 'CustomerID', 'CustomerFirstName', 'CustomerMiddleName', 'CustomerLastName', 'UserLoginID', 'ContactID', 'CustomerGender', 'CustomerBirthdate', 'CustomerBirthCountry', 'CustomerLicenseDate', 'CustomerLicenseExpirydate', 'CustomerNotes', 'CustomerSearchField', 'LanguageID', 'BaseCodeSalutationID', and 'EmailAddress'. On the right, the configuration for the new column is shown. The 'Name' field is 'EmailAddress', 'Data Type' is 'CHARACTER (varchar)', 'Size' is '100', and 'Format' is 'x(50)'. The 'Label' is 'Email'. The 'Help Text' is 'Please enter your full Email address'. The 'Finish' button is highlighted with a red arrow.

continued on next page

Lab 6 – Modifying Schema using the DB Navigator, continued

Adding a new column to the existing Table, continued

5. Click **Finish**. The DB Structure view will display the new column.



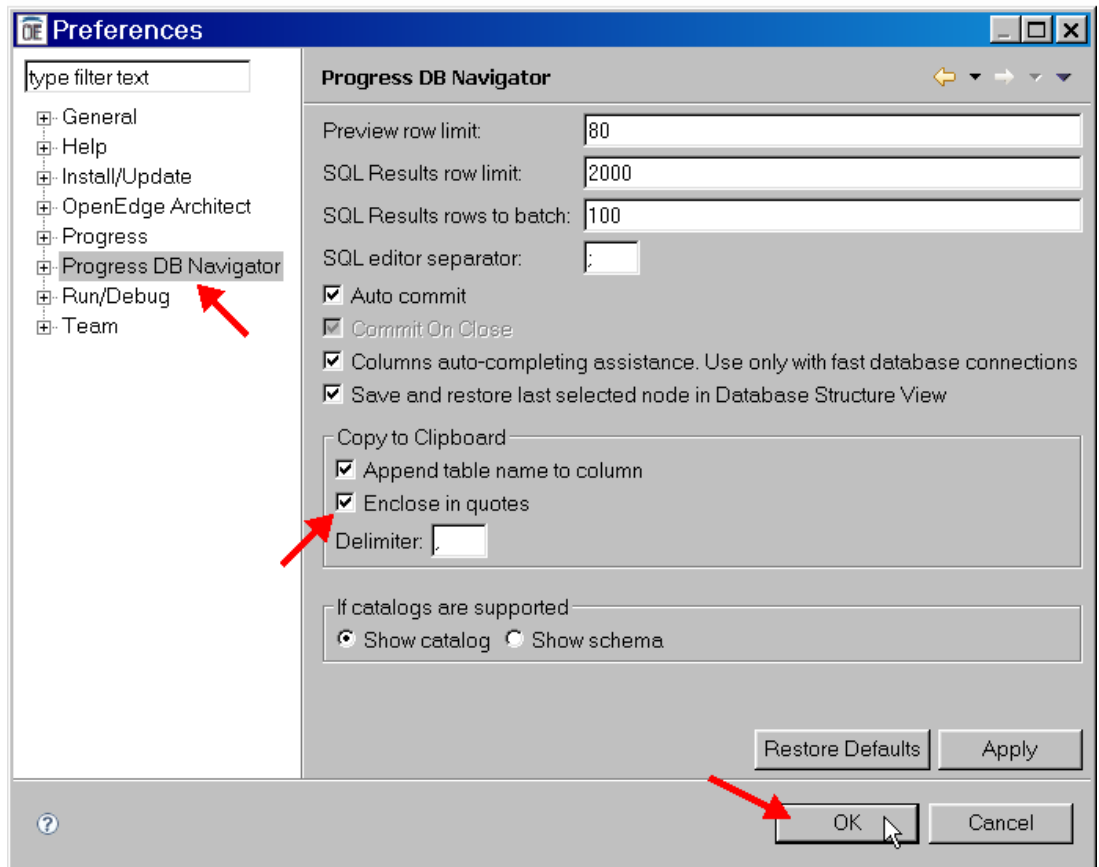
continued on next page

Lab 6 – Modifying Schema using the DB Navigator, continued

Generating a Create Table script

This section covers creating a SQL Create Table script for a table in the database. Once the SQL is generated, you can use the scripts for deployment purposes or to create a variation of the table by editing the script.

1. From the Architect menu, select **Window→Preferences**.
2. Select **Progress DB Navigator** in the left pane.
3. Check the **Enclose in quotes** box. This insures case sensitivity when generating the Create Table script.



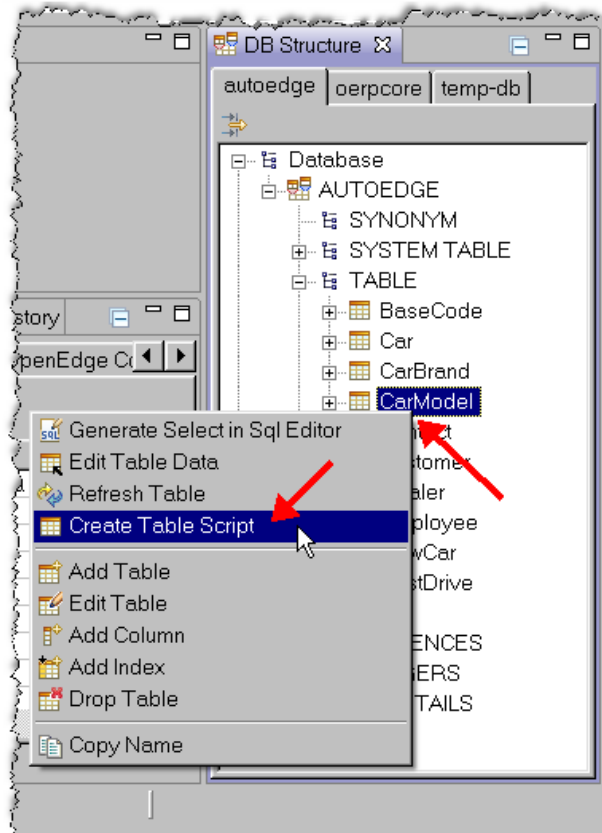
4. Choose **OK** to save and close the Preferences dialog.
5. In the DB Structure view, select the **autoedge** tab and expand the **Database→AUTOEDGE→TABLE** nodes.

continued on next page

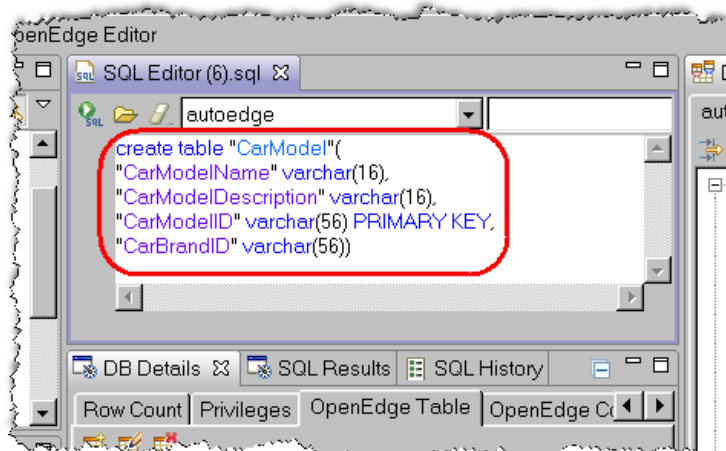
Lab 6 – Modifying Schema using the DB Navigator, continued

Generating a Create Table script, continued

6. Right-click on the **CarModel** table and select **Create Table Script**.



The generated table script is created and displayed in a SQL Editor.

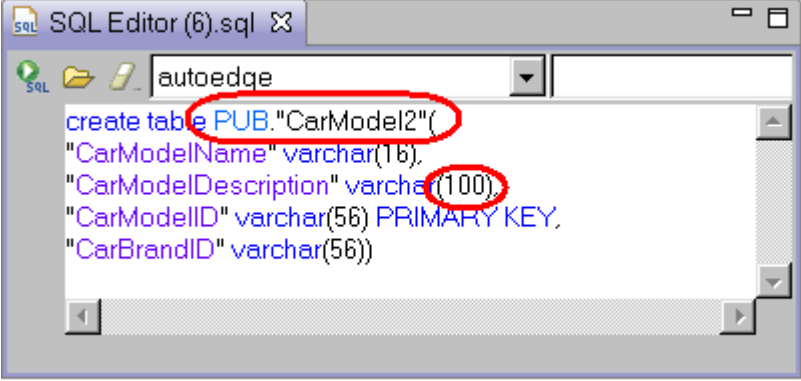


continued on next page


Lab 6 – Modifying Schema using the DB Navigator, continued


Generating a Create Table script, continued

7. Change the table name from “CarModel” to PUB.”CarModel2” and set CarModelDescription to varchar(100).



```
SQL Editor (6).sql
autoedge
create table PUB."CarModel2"(
  "CarModelName" varchar(16),
  "CarModelDescription" varchar(100),
  "CarModelID" varchar(56) PRIMARY KEY,
  "CarBrandID" varchar(56))
```

 **Tip:** PUB must be included in the script to include it in the proper schema area. For OpenEdge, the default schema area is always PUB.

 **Tip:** To maintain case sensitivity for table and column names, the names must be surrounded by quotes. For instance, if you don't quote CarModel2, then the table name will be added as CARMODEL2 when the script is executed.

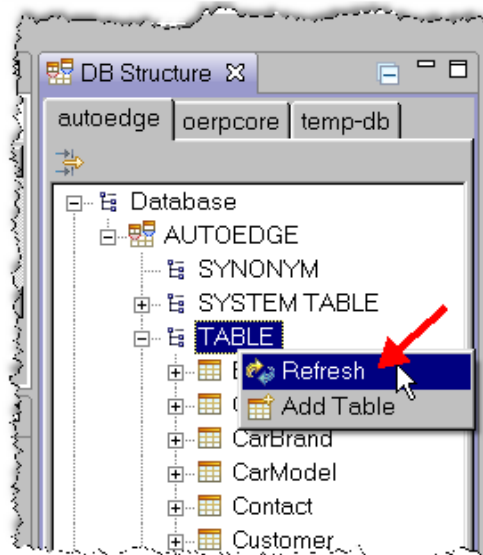
8. Execute the script.

continued on next page

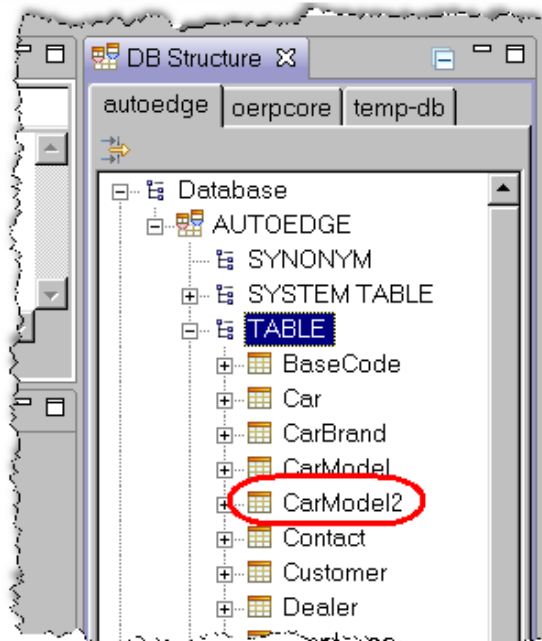
Lab 6 – Modifying Schema using the DB Navigator, continued

Generating a Create Table script, continued

9. Right-click on the **Table** node in the DB Structure view and select **Refresh**.



The newly created table is shown.



continued on next page

Lab 6 – Modifying Schema using the DB Navigator, continued

Generating a Create Table script, continued

10. Save the file as **carModel2CreateTable.sql** in the AutoEdge project in the src directory.
 11. Close the file.
-

Lab 7 – Using the OpenEdge Editor

Overview

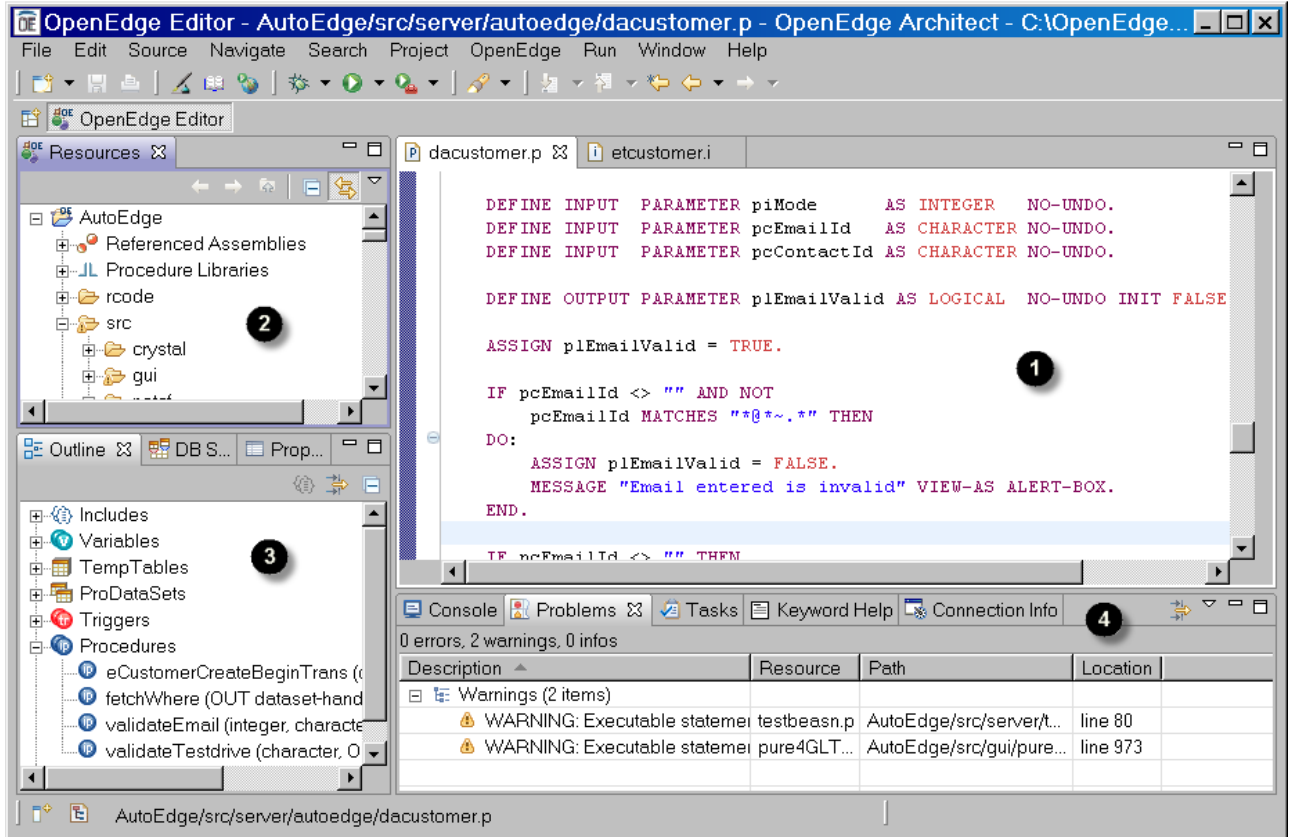
The OpenEdge Editor provides many features to facilitate in the development of OpenEdge applications. This lab is designed to demonstrate the functionality of the OpenEdge Editor and the OpenEdge Editor perspective.

The lab consists of creating a new ABL program and displaying some data from the database. Many of the available usability features for the Editor are covered through the process.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

OpenEdge Editor Perspective



Notes:

- 1 ABL Editor
- 2 Resources View
- 3 Outline, DB Structure, and Properties Views
- 4 Console, Problems, Tasks, Keyword Help, and Connection Info Views

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Creating a new ABL program

The section shows how to add a new ABL procedure file to an existing project.

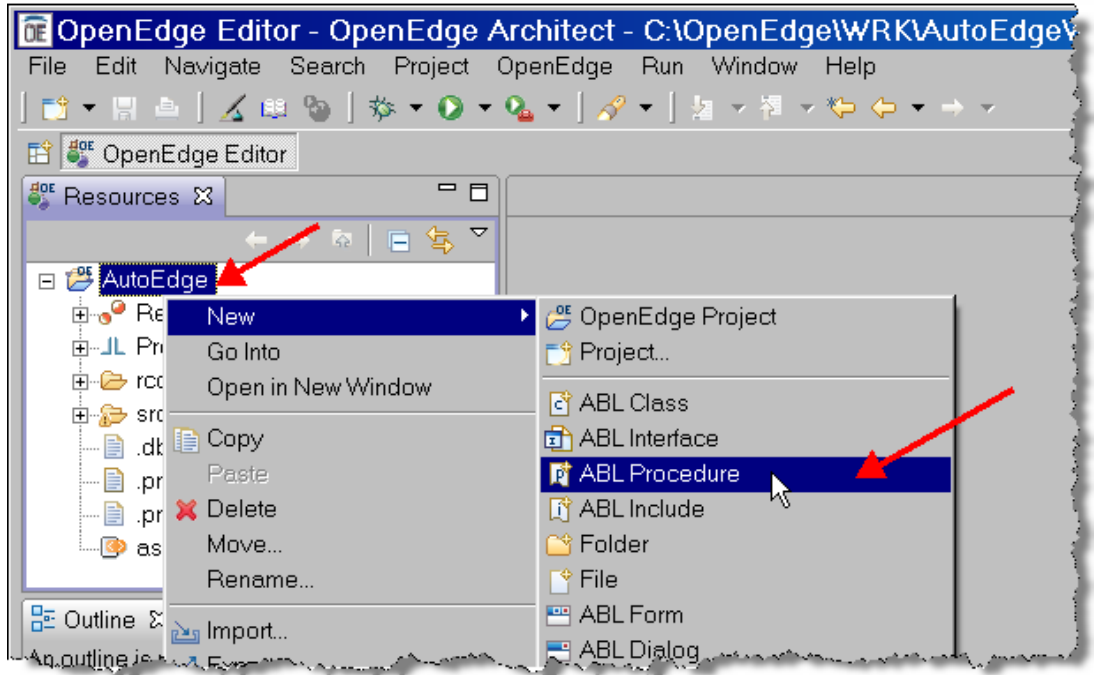
1. Open the **OpenEdge Editor** perspective or switch to the perspective if it is already open. Close other perspectives, and close any resources that are open in the editor.
2. Expand the **AutoEdge** node in the Resources view.


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Creating a new ABL program, continued

3. Right-click on the **AutoEdge** project and select **New→ABL Procedure**.



 **Note:** As previously discussed, you can also use the New button or the Architect menu File→New menu option to create new resources or projects. Just make sure that the project in which you want to create the resource is selected in the Resources view.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Creating a new ABL program, continued

4. Enter the following in the wizard:
 - Container: `\AutoEdge\src`
 - File name: `testDBConnection.p`
 - Description: **Example ABL file which includes accessing a database**
 - Author: `<Enter your name>` (defaults to current user name)

New ABL Procedure

Create an ABL procedure file

Optionally identify the author of the procedure. This text will appear in the file header.

Container: Browse...

File name:

Description:

Purpose:

Author:

Add routine level error handling

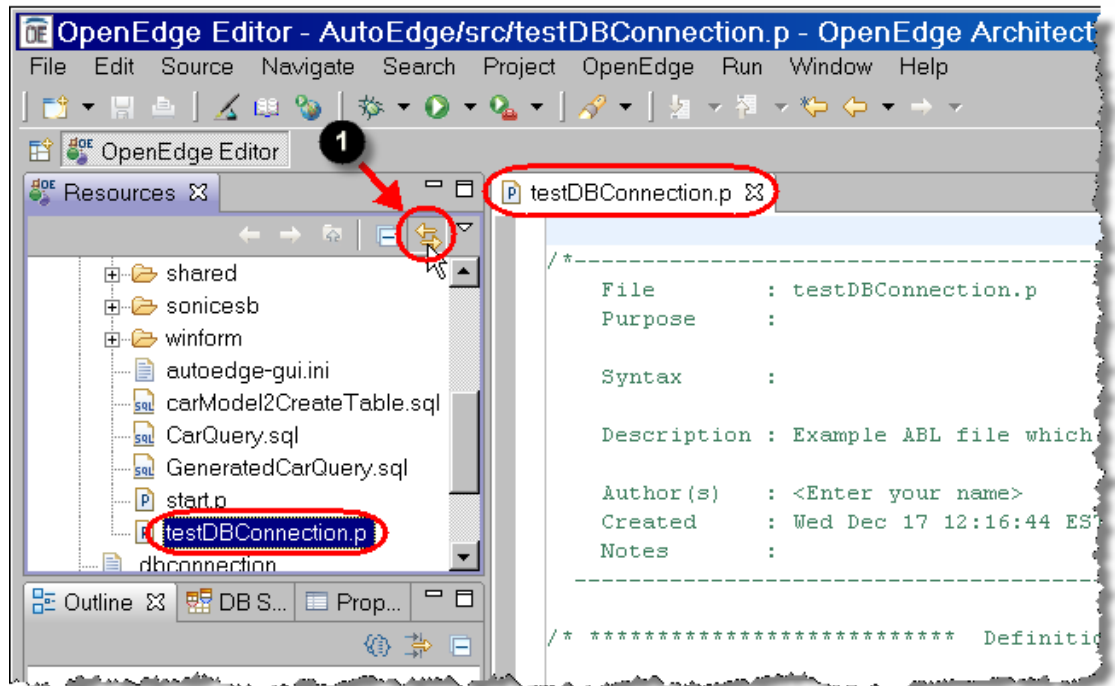
Finish Cancel


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Creating a new ABL program, continued

5. Click **Finish**. The new procedure is added to the project in the **Resources** view in the ABL folder and the file is opened in the Editor.



 **Tip:** A fast way to synch (expand and select the file) the open file resource in the editor with the Resources view is to click the **Link with Editor** button (⚡) in the Resources view's toolbar.

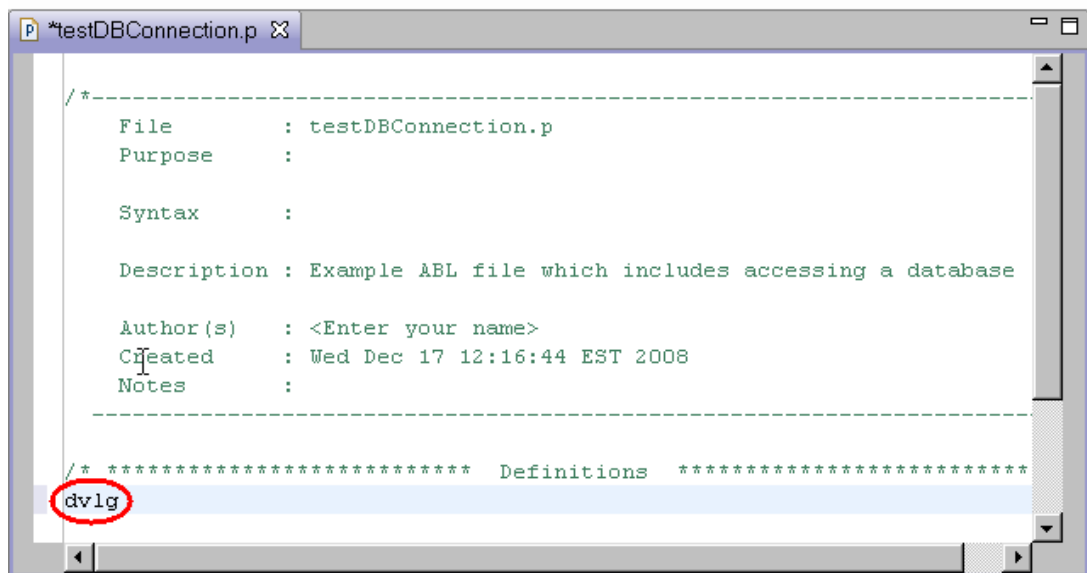
continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor

This section covers writing ABL code using the OpenEdge Editor and will focus on highlighting some of the features of the Editor.

1. In the new file, after the Definitions comment, type **dvlg**. These characters represent a macro that expands to a logical variable definition.



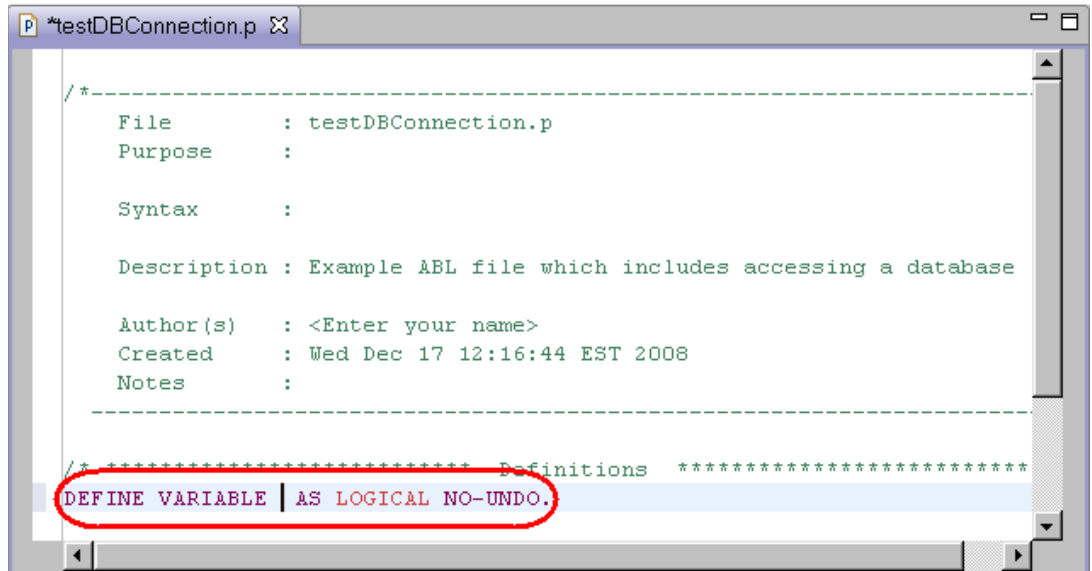
```
/*-----  
File      : testDBConnection.p  
Purpose   :  
  
Syntax    :  
  
Description : Example ABL file which includes accessing a database  
  
Author(s)  : <Enter your name>  
Created   : Wed Dec 17 12:16:44 EST 2008  
Notes     :  
-----  
/* ***** Definitions *****  
dvlg
```

continued on next page


Lab 7 – Using the OpenEdge Editor, continued


Entering code in the ABL Editor, continued

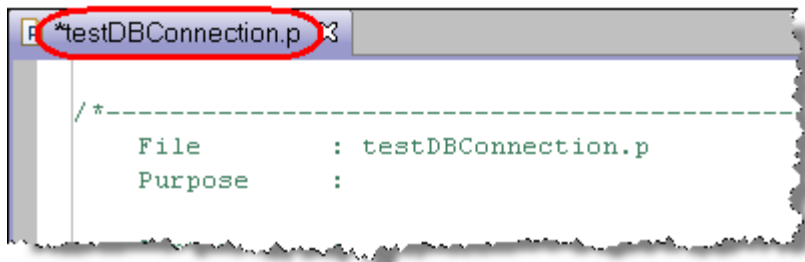
2. Press the **Spacebar**. The macro is expanded.



```
/*-----  
File      : testDBConnection.p  
Purpose   :  
  
Syntax    :  
  
Description : Example ABL file which includes accessing a database  
  
Author(s)  : <Enter your name>  
Created    : Wed Dec 17 12:16:44 EST 2008  
Notes     :  
-----  
/* ***** Definitions *****  
DEFINE VARIABLE AS LOGICAL NO-UNDO.
```

 **Tip:** Architect supports a number of pre-defined macros and even allows you to create your own. Available macros can be found in Workspace Preferences (**Windows**→**Preferences**) in the **OpenEdge Architect**→**Editor**→**Templates(Macros)** section. See online help for more information on macros.

 **Note:** An asterisk (*) next to a program name in the Editor tab indicates that the file has been modified in the editor since the last time it was saved.



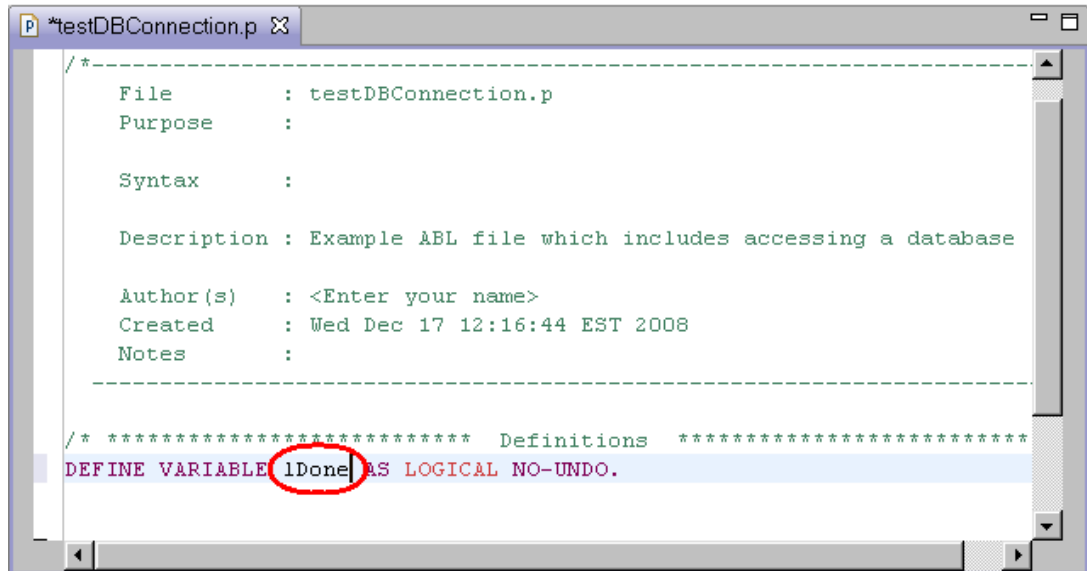
```
*testDBConnection.p  
/*-----  
File      : testDBConnection.p  
Purpose   :
```

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

3. After the VARIABLE keyword enter the variable name **lDone**.



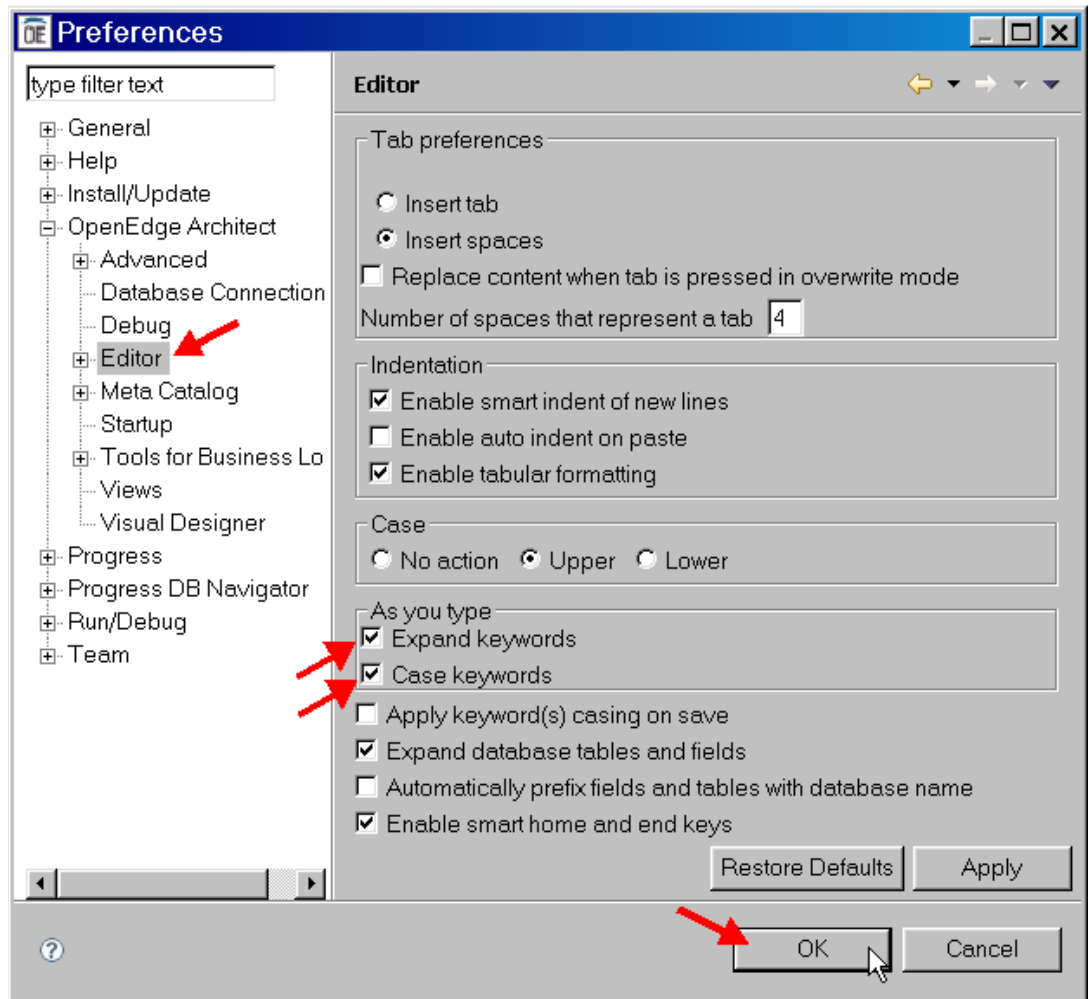
```
/*-----  
File       : testDBConnection.p  
Purpose    :  
  
Syntax     :  
  
Description: Example ABL file which includes accessing a database  
  
Author(s)  : <Enter your name>  
Created    : Wed Dec 17 12:16:44 EST 2008  
Notes      :  
-----  
/* ***** Definitions *****  
DEFINE VARIABLE lDone AS LOGICAL NO-UNDO.
```


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

4. Open the Workspace Preference dialog by selecting **Window**→**Preferences**. Expand the **OpenEdge Architect** node and select the **Editor** entry. Check the **Expand keywords** and **Case keywords** boxes. Click **Ok**. This will instruct the editor to fully expand partially typed keywords and to automatically capitalize recognized keywords.



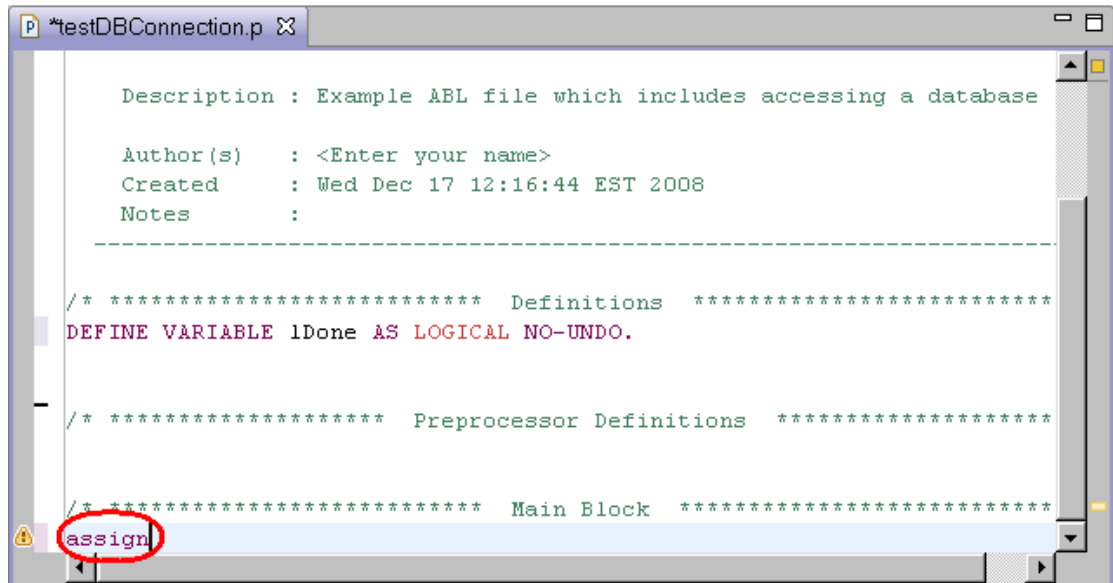
 **Tip:** You can control many of the Editor's behaviors by modifying preferences. View online help for information about the options.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

5. Back in the editor, type **assign** after the Main Block comment.



```
*testDBConnection.p

Description : Example ABL file which includes accessing a database

Author(s)   : <Enter your name>
Created    : Wed Dec 17 12:16:44 EST 2008
Notes      :

-----

/* ***** Definitions *****
DEFINE VARIABLE lDone AS LOGICAL NO-UNDO.

/* ***** Preprocessor Definitions *****

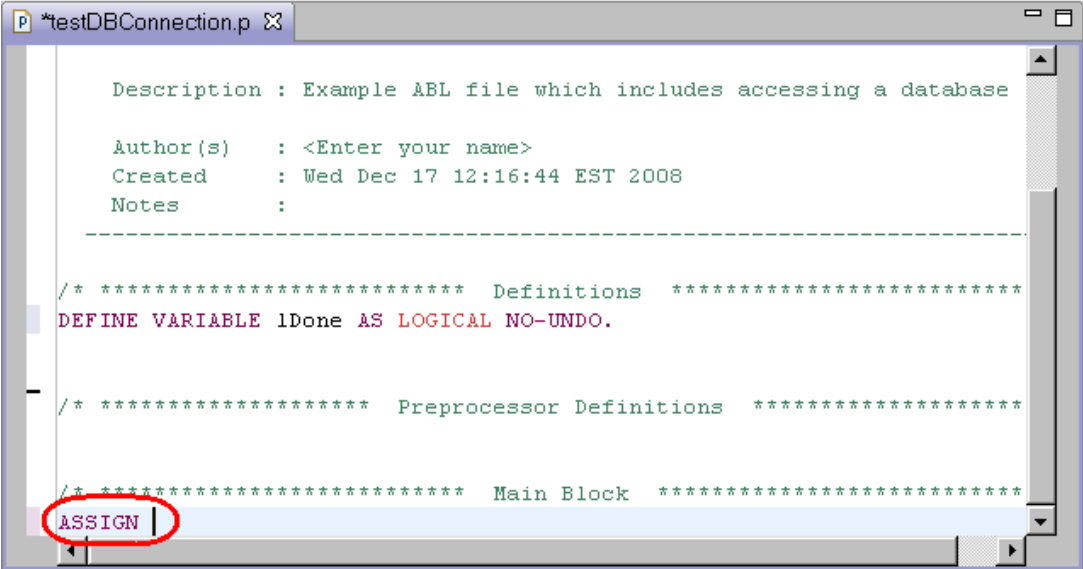
/* ***** Main Block *****
assign
```

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

- 6. Press the **Spacebar**. The ASSIGN keyword is automatically capitalized.

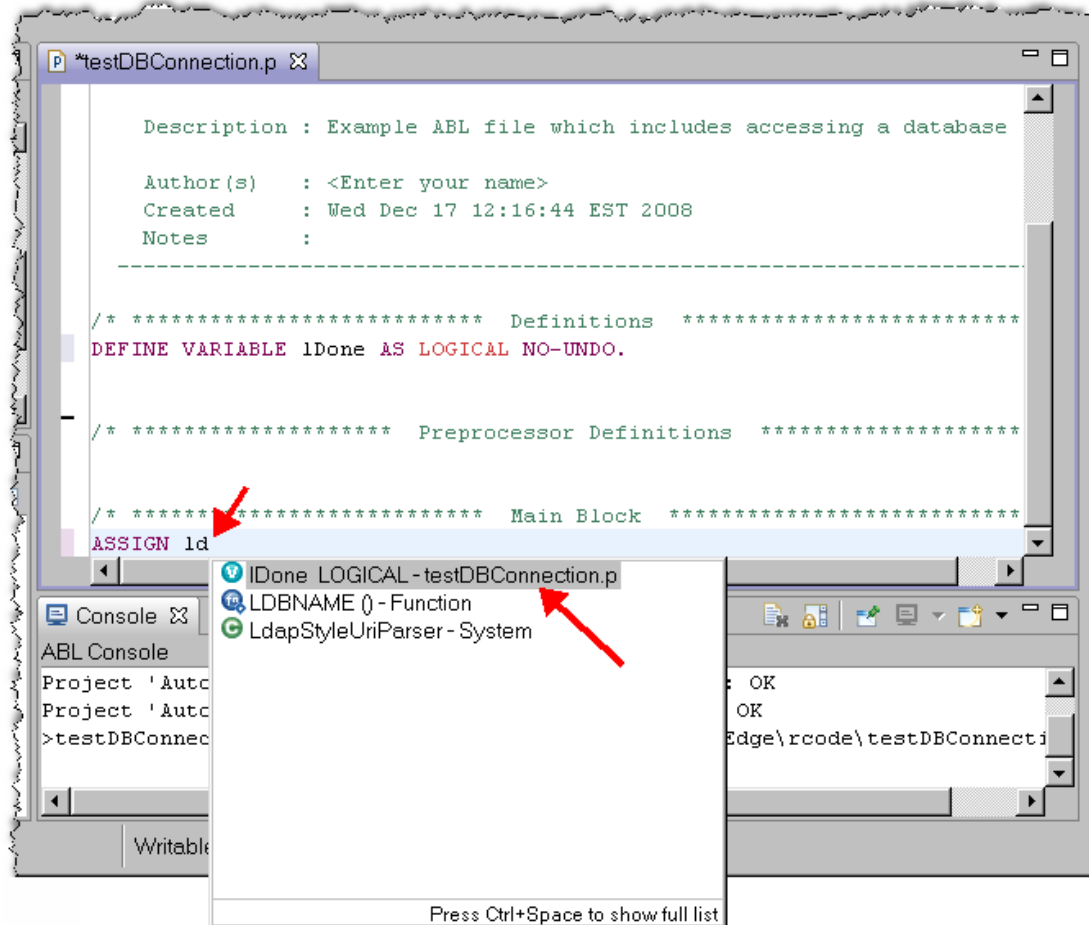


continued on next page




Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

7. Type **ID** on the same line and then press the Ctrl and Spacebar keys.



Upon pressing the key sequence Ctrl-Space, a list of options is displayed. These options are valid object names (such as variables, keywords, or database names) beginning with the characters **ld** that can be selected to complete the typed code. On top of the list is the variable **IDone** that was coded earlier in this lab. You can scroll through the list of options by using the arrow keys.

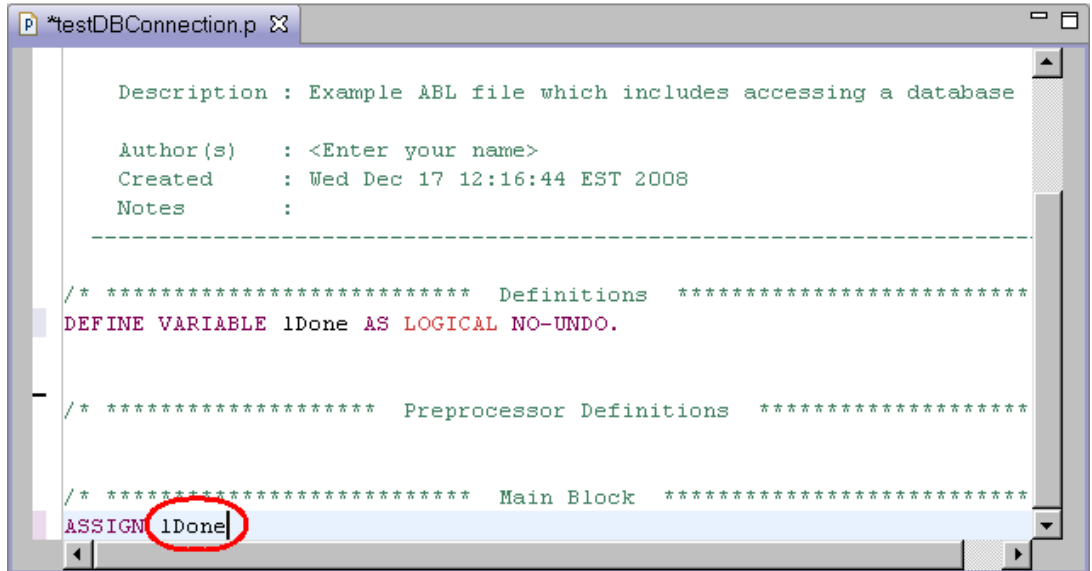
 **Tip:** It is useful to become familiar with the icons that are used throughout Architect. Knowing the meaning of the icons can help you quickly understand what is being presented to you. For example, in the list of options that was presented, the  icon is used to identify variables and  icon is used to indicate that an item is a function.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

- To select the `!Done` option, press the **Enter** key (you can also double-click on the option with your mouse). The code will be completed with that option and the pop-up list will disappear.



```

Description : Example ABL file which includes accessing a database

Author(s)   : <Enter your name>
Created    : Wed Dec 17 12:16:44 EST 2008
Notes     :

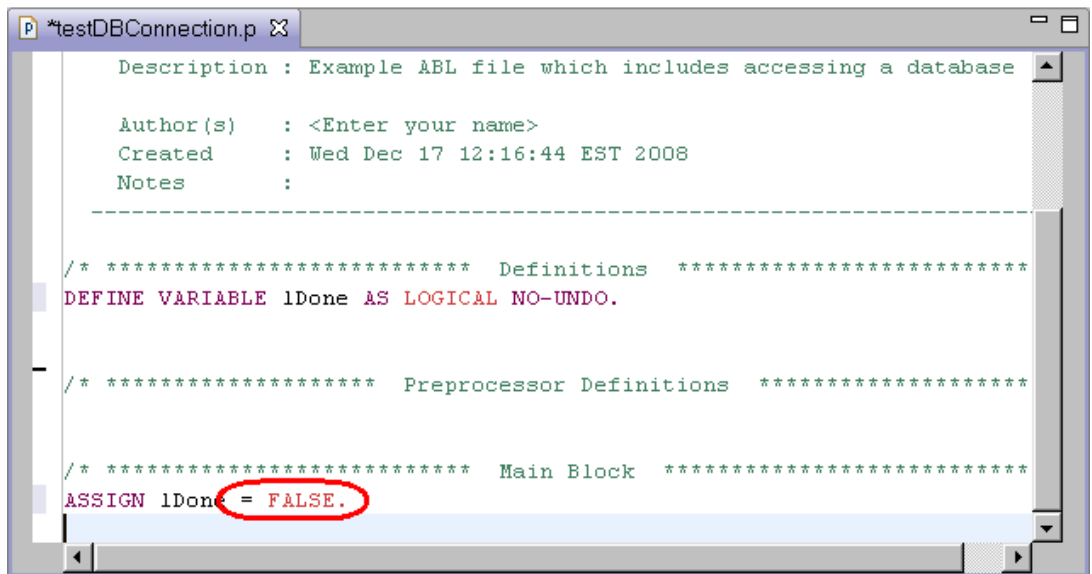
-----

/* ***** Definitions *****
DEFINE VARIABLE !Done AS LOGICAL NO-UNDO.

/* ***** Preprocessor Definitions *****

/* ***** Main Block *****
ASSIGN !Done|
```

- Complete the line by typing `= false`. Press **Enter** to start a new line.



```

Description : Example ABL file which includes accessing a database

Author(s)   : <Enter your name>
Created    : Wed Dec 17 12:16:44 EST 2008
Notes     :

-----

/* ***** Definitions *****
DEFINE VARIABLE !Done AS LOGICAL NO-UNDO.

/* ***** Preprocessor Definitions *****

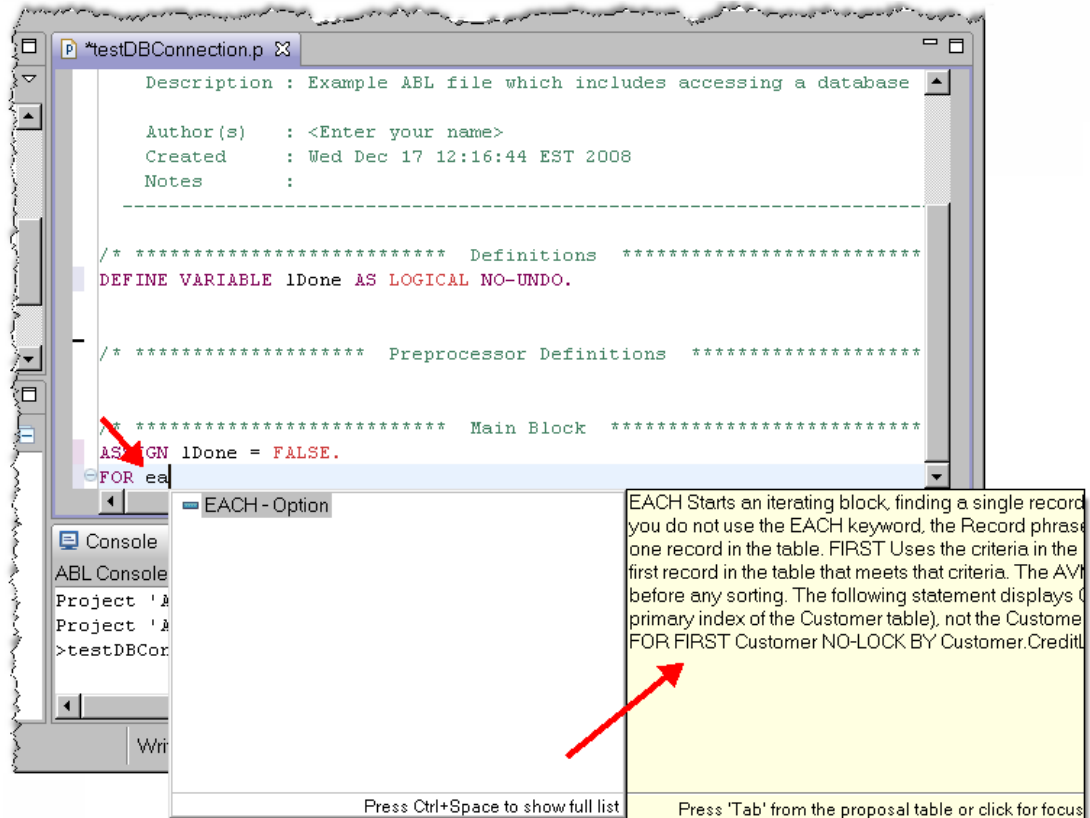
/* ***** Main Block *****
ASSIGN !Done = FALSE.
```

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

10. Type **for ea** and press the Spacebar. The EACH option is presented along with a pop-up help box that offers information about the keyword. You can give the box focus and scroll to read the help. Press **Enter** to select the option.

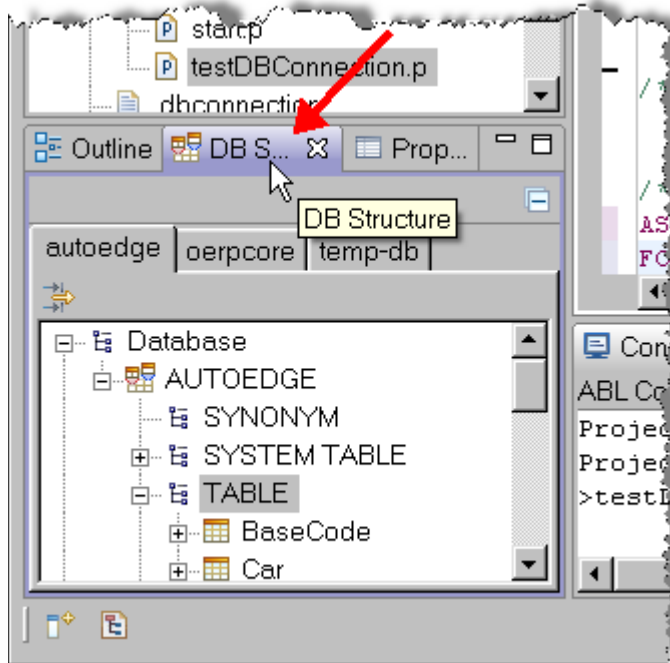



continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

11. Select the **DB Structure** view (by default it stacked below the Outline view). This is the same DB Structure view that is part of the DB Navigator perspective. A tab for each database associated with the project is displayed.



 **Note:** OpenEdge Architect makes it much easier and faster to work with database schema than was possible when using OpenEdge Studio. One reason for this is the functionality of the DB Structure view in the OpenEdge Editor perspective. Having the DB Structure view available at the same time that you are coding allows modeless searching of a database, the ability to quickly see data types of fields, and the ability to copy schema names including tables and columns directly into the Editor.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued



Tip: If the autoedge tab is not displayed in the DB Structure view, then the most likely cause is that a SQL connection to the database has not been made. Go to the Connections view in the OpenEdge DB Navigator perspective (or alternatively open that view in the OpenEdge editor perspective). In the Connections view, connect to the database if it is not connected. Select the active connection for the database. The database should now be available in the DB Structure view.



Tip: What if the previous tip didn't resolve the problem? Another possible reason that the database is not listed in the DB Structure view is that the database server is not started. Architect contains some clues that let you know that this is the problem. One is that the color coding for database references in the code is black (coloring in the Editor will be discussed later). Also check the Console view to see if there are any connection failure messages.

If a connection failure exists, restart the database server. Once it is started, restart the AVM for each project that needs to connect to the database by selecting **Project→Restart OpenEdge Runtime** from the main menu. Confirm that everything was started correctly by checking the messages in the Console view.



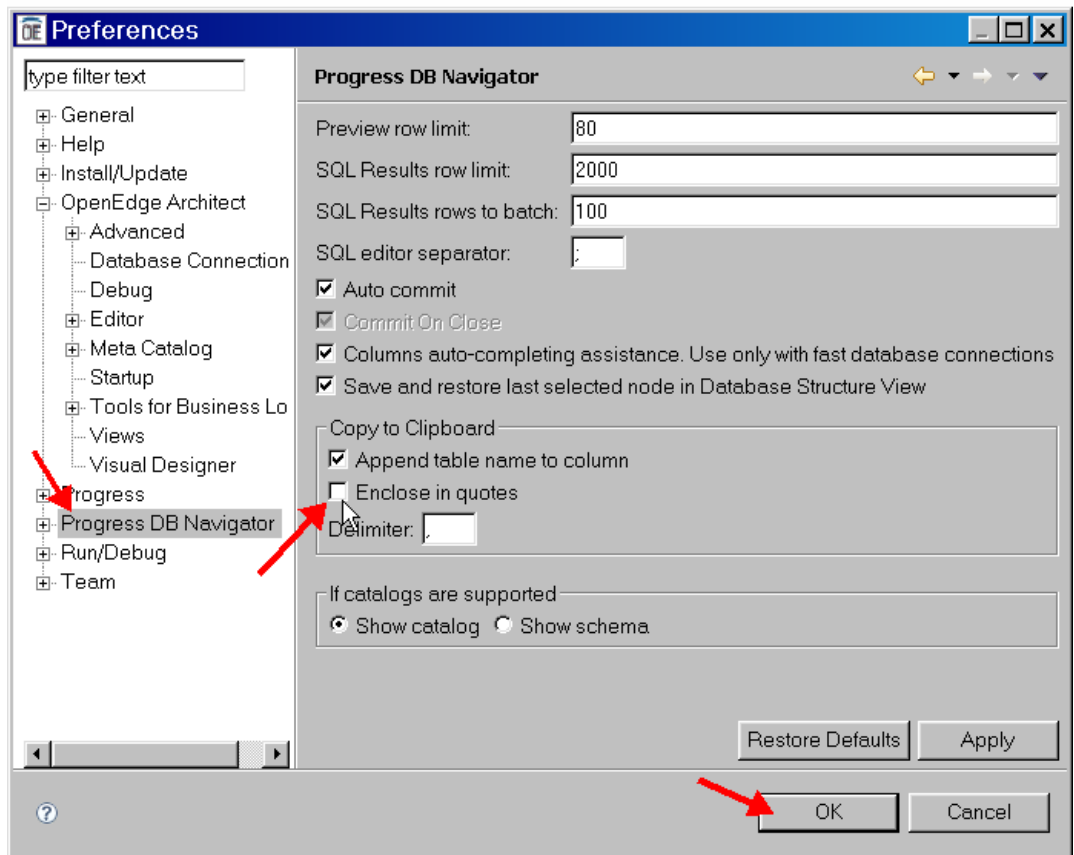
Tip: What if you have many projects that need to have their AVM's restarted to connect to the recently restarted database? Instead of selecting Restart OpenEdge Runtime for each project, you can restart the workspace by selecting **File→Switch Workspace** from the Workbench menu. The current workspace is supplied in the Workspace Launcher dialog. Instead of selecting a different workspace, just click OK and the current workspace and all associated AVM's will be restarted.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

- Table and field quoting was turned on in a previous lab to maintain case when working with the SQL Editor. For ABL code, you do not want the table and field names to be in quotes. Select **Window→Preferences** to open the Preferences window. Select the **Progress DB Navigator** entry in the left pane and uncheck the **Enclose in quotes** option. Select the **OK** button to close the window.

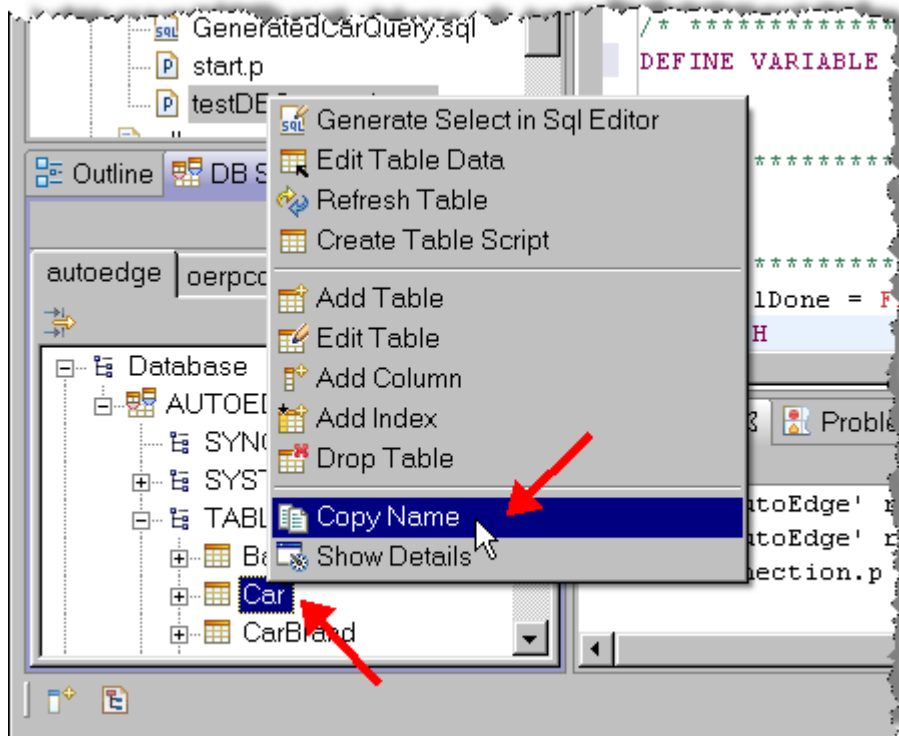


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

13. Make sure the **autoedge** tab is selected in the DB Structure view. Expand the **AUTOEDGE→TABLE** nodes.
14. Right-click on the **Car** table and select **Copy Name**.

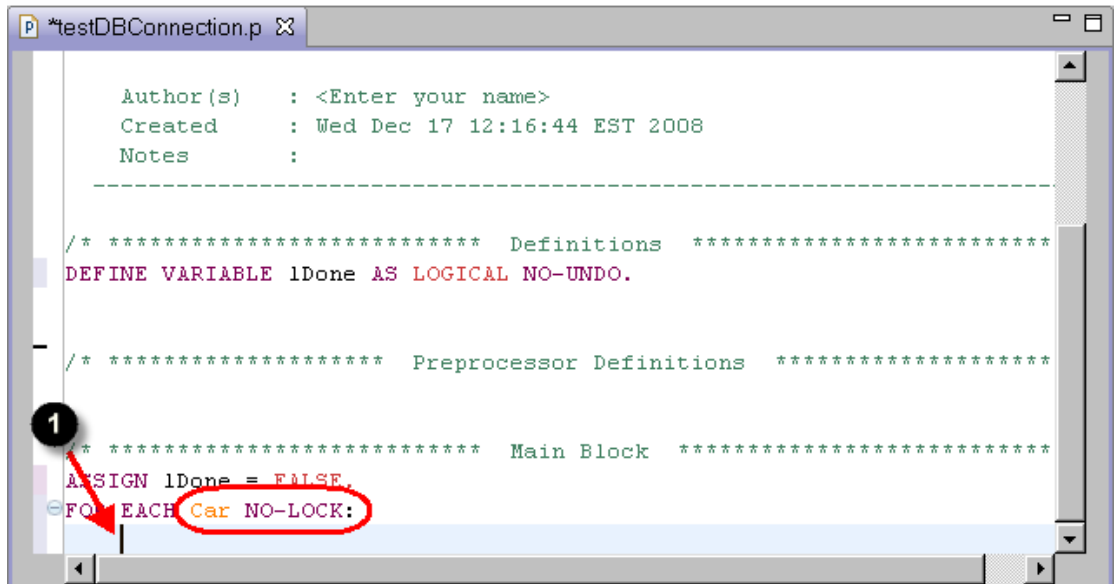


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

15. Position the cursor after **EACH** in the editor. Right-click and select paste (or press CTRL-V) to paste the table name. Finish the line by typing **NO-LOCK:** and press **Enter** to start a new line.





```
Author(s) : <Enter your name>
Created   : Wed Dec 17 12:16:44 EST 2008
Notes    :
-----
/* ***** Definitions *****
DEFINE VARIABLE lDone AS LOGICAL NO-UNDO.

/* ***** Preprocessor Definitions *****

/* ***** Main Block *****
ASSIGN lDone = FALSE.
FOR EACH Car NO-LOCK:

```

 **Note:** When starting a new line in a coding block (in this case, a FOR EACH block), the editor will automatically start the next line as an indented line. This option can be turned off in the Preferences window under the Editor section.

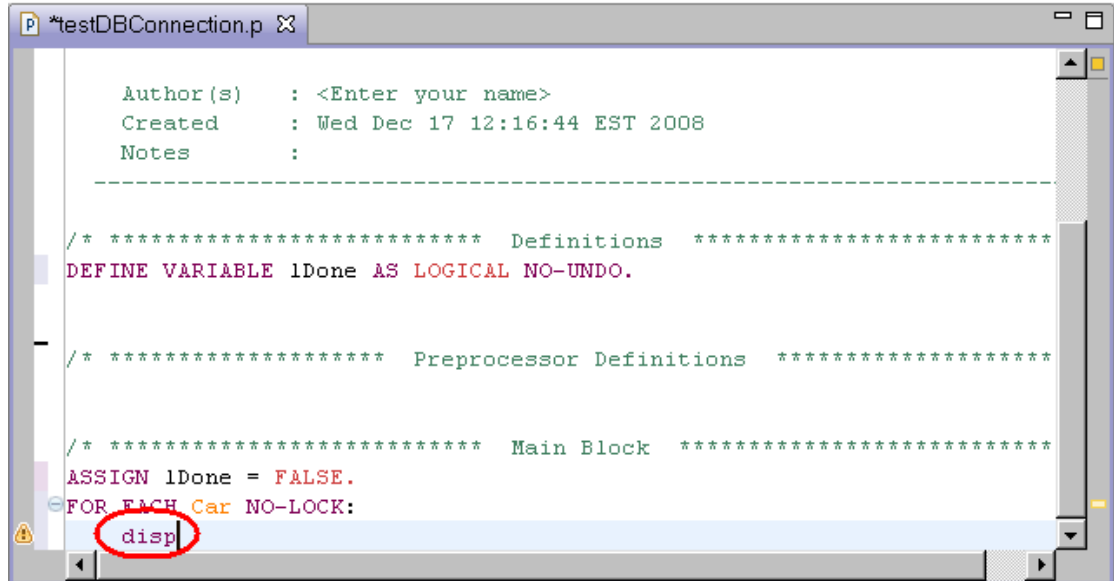
 **Note:** The screen shot above shows some of the varied syntax coloring that is possible in the OpenEdge Editor. The coloring options in the OpenEdge Editor are much more sophisticated than those previously available in OpenEdge Studio. This sophistication is reflected in the colors shown in the Editor such as keywords in purple, data types in red, and database tables and fields in orange. For a full list of the coloring options, and the ability to change specific colors, go to Workspace Preferences and select **OpenEdge Architect**→**Editor**→**Colors** in the left pane. While the coloring is nice for quickly recognizing syntax structures, it is only the beginning since Architect uses the knowledge of the types of code to provide numerous advanced editing features, which will be shown throughout this lab.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

16. Type **disp** on the new line.



```
Author(s) : <Enter your name>
Created   : Wed Dec 17 12:16:44 EST 2008
Notes    :
-----

/* ***** Definitions *****
DEFINE VARIABLE lDone AS LOGICAL NO-UNDO.

/* ***** Preprocessor Definitions *****

/* ***** Main Block *****
ASSIGN lDone = FALSE.
FOR EACH Car NO-LOCK:
  disp
```

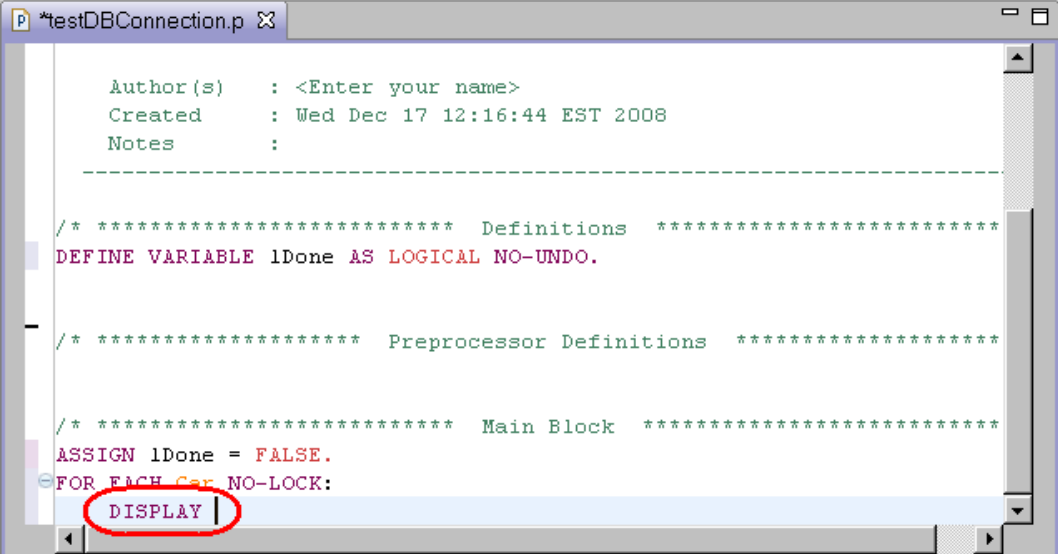
Notice that while you are typing the characters change colors. This lets you know that the editor is trying to match the characters to a corresponding syntax element. After typing the **p** in **disp**, the characters turn purple. This means that the editor has recognized the characters as a match for a keyword. Because we have previously set the preference to expand partial keywords, pressing the Spacebar will complete the keyword.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

17. Press the **Spacebar**. The keyword will be completed for you.



The screenshot shows a window titled '*testDBConnection.p'. The code is as follows:

```
Author(s) : <Enter your name>
Created   : Wed Dec 17 12:16:44 EST 2008
Notes    :
-----
/* ***** Definitions *****
DEFINE VARIABLE lDone AS LOGICAL NO-UNDO.

/* ***** Preprocessor Definitions *****

/* ***** Main Block *****
ASSIGN lDone = FALSE.
FOR EACH Car NO-LOCK:
    DISPLAY |
```

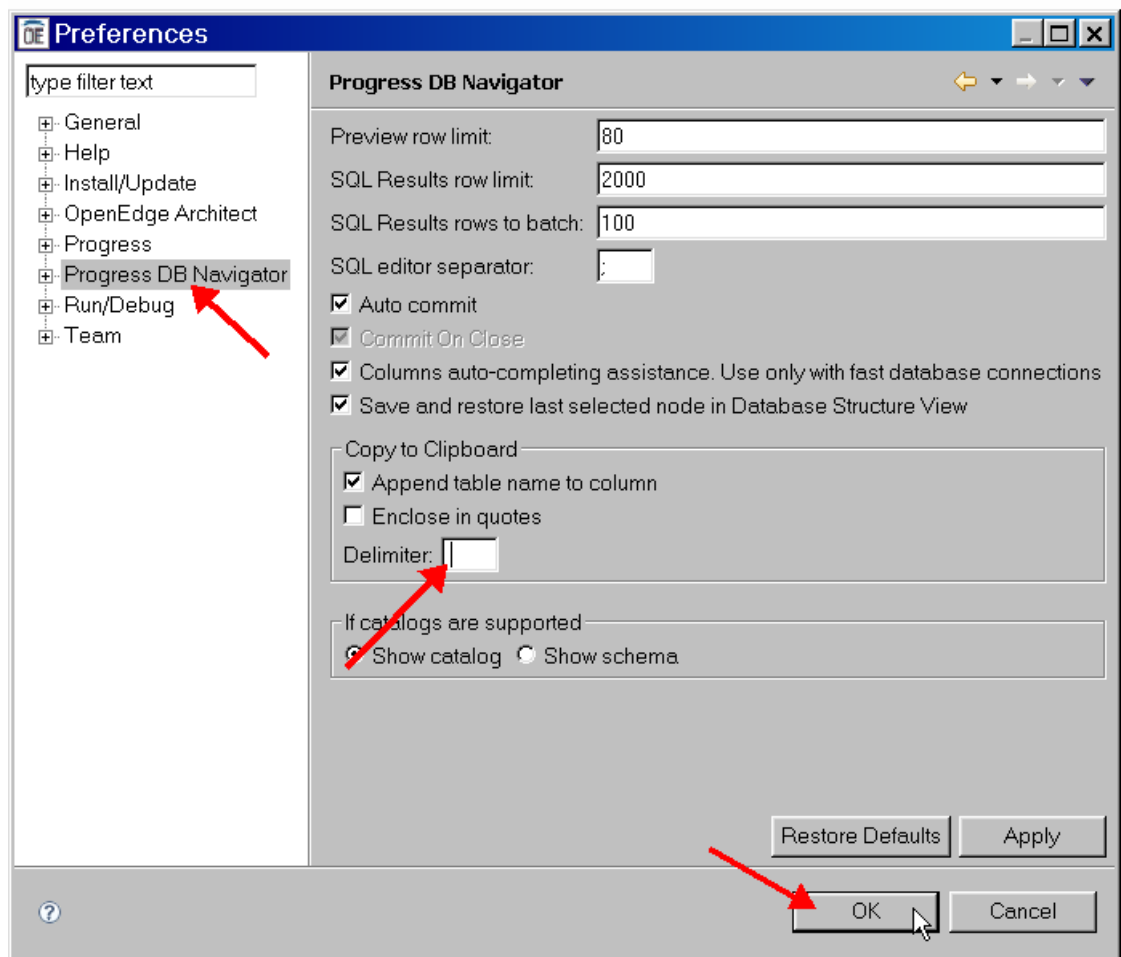
The word 'Car' in the 'FOR EACH' statement is highlighted in blue, and a red circle highlights the word 'DISPLAY' on the line below it, indicating that the editor is suggesting the completion of the keyword.


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

18. In the DB Structure view, expand the **Car**→**Columns** nodes.
19. A modification needs to be made to the DB Navigator preferences for copying multiple field names from the DB Structure view to the editor. Open the Preferences dialog and select **Progress DB Navigator**. Change the Delimiter from a comma to a space and close the Preferences dialog. Changing this option will separate multiple copied fields by a space instead of a comma.



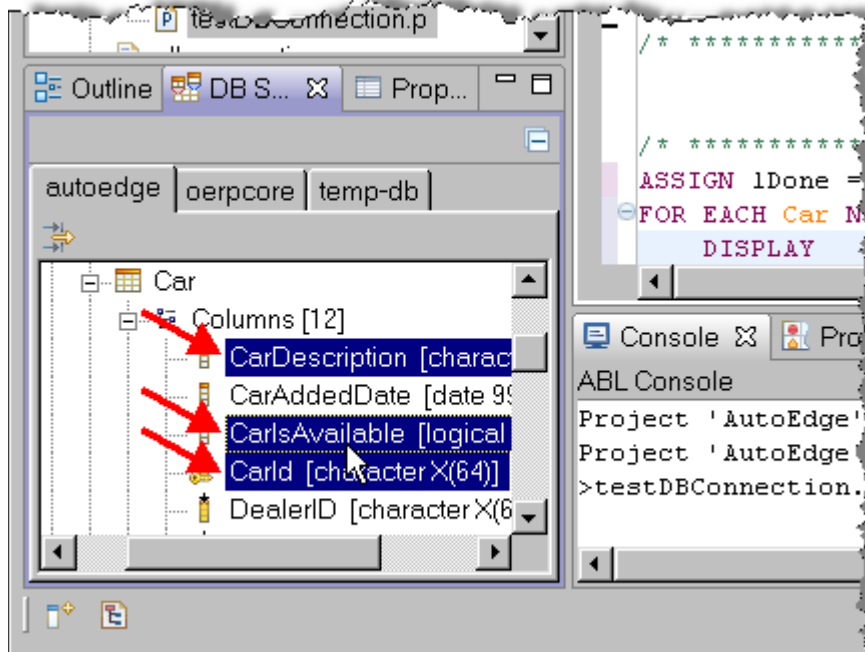
 **Tip:** A comma delimiter is what you would likely want if you are working with the SQL Editor.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

20. Click on the **CarID** column. Hold down the Ctrl key and click on the **CarDescription** and **CarIsAvailable** columns.

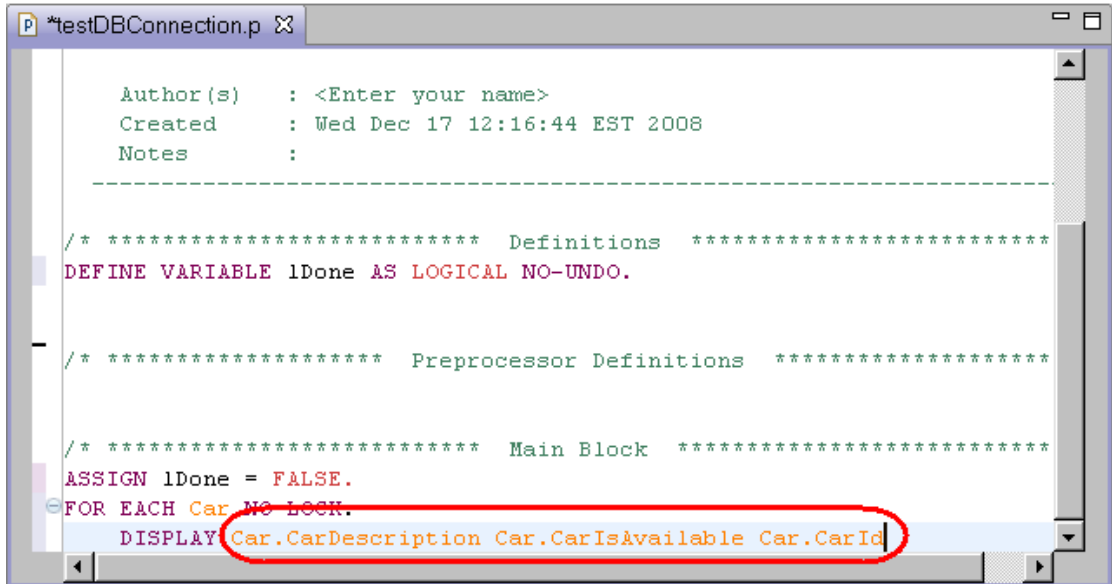


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

21. With the three fields selected, right-click on one of the selections and select **Copy Name**. In the editor after the DISPLAY statement, paste the field names. This adds the three fields qualified with their table name to the editor.

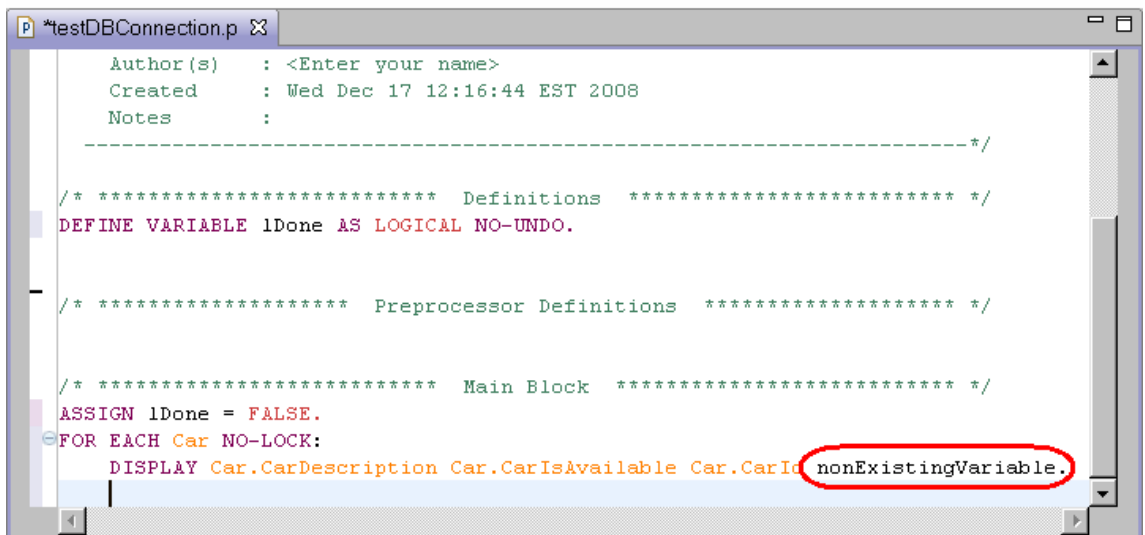


```
Author(s) : <Enter your name>
Created   : Wed Dec 17 12:16:44 EST 2008
Notes    :
-----
/* ***** Definitions *****
DEFINE VARIABLE lDone AS LOGICAL NO-UNDO.

/* ***** Preprocessor Definitions *****

/* ***** Main Block *****
ASSIGN lDone = FALSE.
FOR EACH Car NO-LOCK:
    DISPLAY Car.CarDescription Car.CarIsAvailable Car.CarId
```

22. Finish the display statement by typing **nonExistingVariable.** and press **Enter** to start a new line.



```
Author(s) : <Enter your name>
Created   : Wed Dec 17 12:16:44 EST 2008
Notes    :
-----
/* ***** Definitions *****
DEFINE VARIABLE lDone AS LOGICAL NO-UNDO.

/* ***** Preprocessor Definitions *****

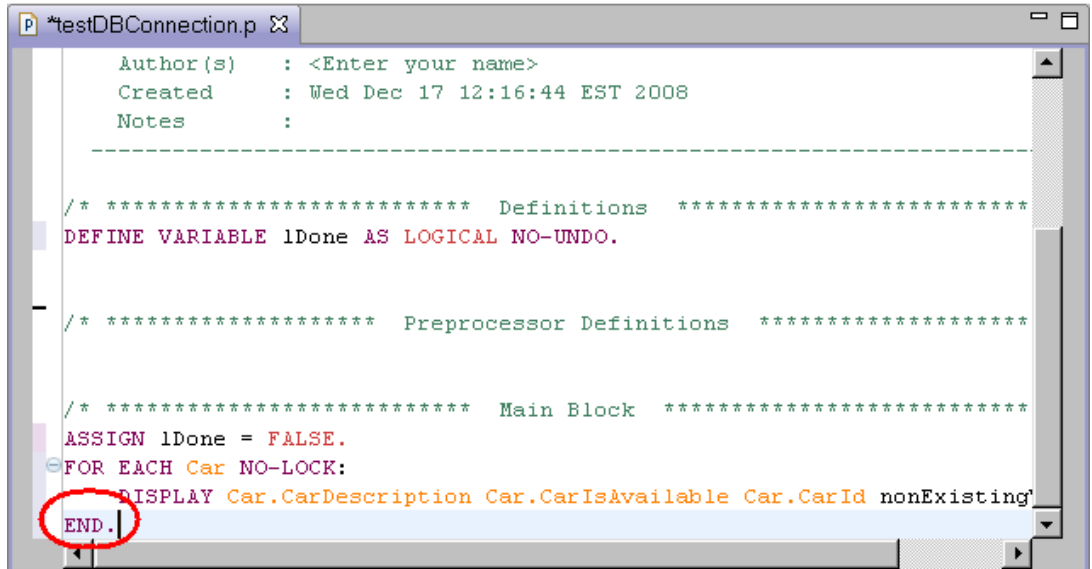
/* ***** Main Block *****
ASSIGN lDone = FALSE.
FOR EACH Car NO-LOCK:
    DISPLAY Car.CarDescription Car.CarIsAvailable Car.CarId nonExistingVariable.
```

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Entering code in the ABL Editor, continued

23. Press Shift-Tab to remove the indent from the new line. Type **end**.




```
Author(s) : <Enter your name>
Created   : Wed Dec 17 12:16:44 EST 2008
Notes    :

-----

/* ***** Definitions *****
DEFINE VARIABLE lDone AS LOGICAL NO-UNDO.

/* ***** Preprocessor Definitions *****

/* ***** Main Block *****
ASSIGN lDone = FALSE.
FOR EACH Car NO-LOCK:
    DISPLAY Car.CarDescription Car.CarIsAvailable Car.CarId nonExisting
END.
```

 **Tip:** If code indentation gets out of order at any time, you can press **CTRL+I** to automatically correct indentation levels. You can also use the **Source→Correct Indentation** menu option from Architect’s menu.

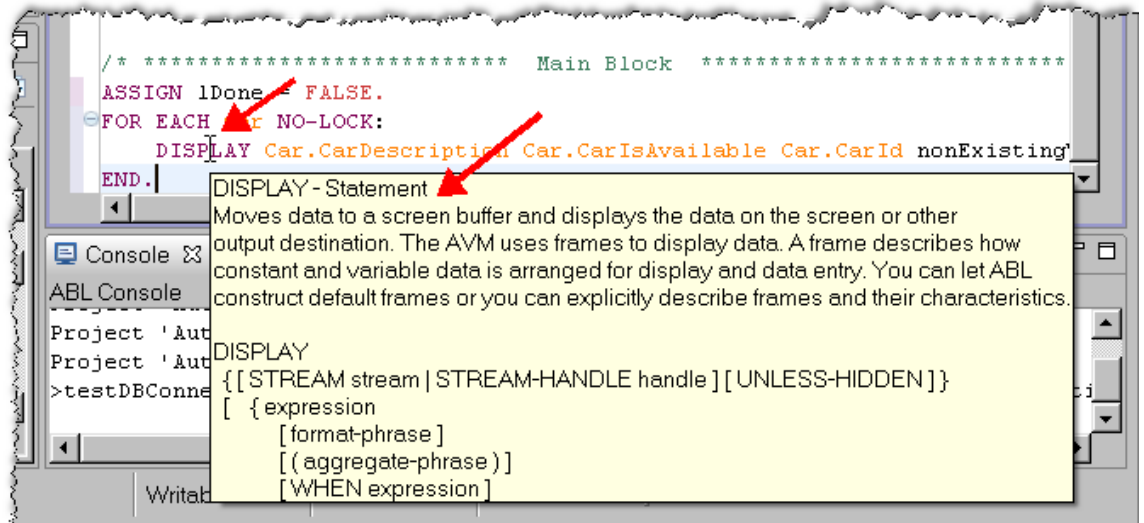
continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Using Coding Aids

There are a number of coding aids available in Architect that can assist in the coding process. The following steps demonstrate some of those coding aid features.

1. Move the mouse and hover over the **DISPLAY** keyword in the editor. Context help for the keyword pops up. This is a quick way to look up information about an ABL keyword.

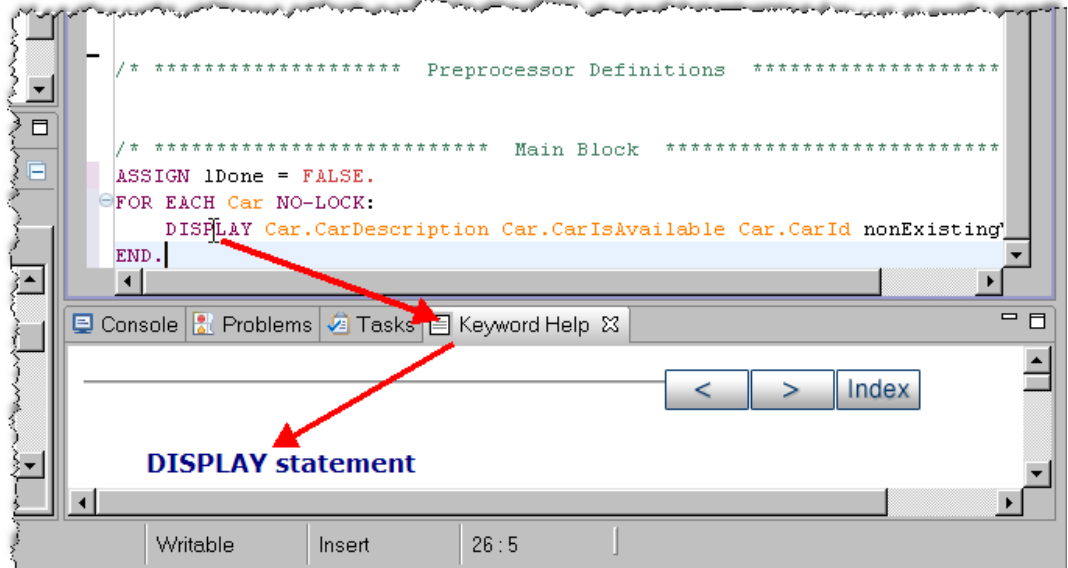



continued on next page


Lab 7 – Using the OpenEdge Editor, continued

Using Coding Aids, continued

2. If you need more information on the keyword, press the **Shift-F2** keys while still hovering over DISPLAY. The Keyword Help view is opened and displays the ABL on-line help page for the DISPLAY statement.



 **Note:** If Shift-F2 does not show keyword help, the likely cause is that the tab for the editor does not have focus. Select the tab folder and try again.

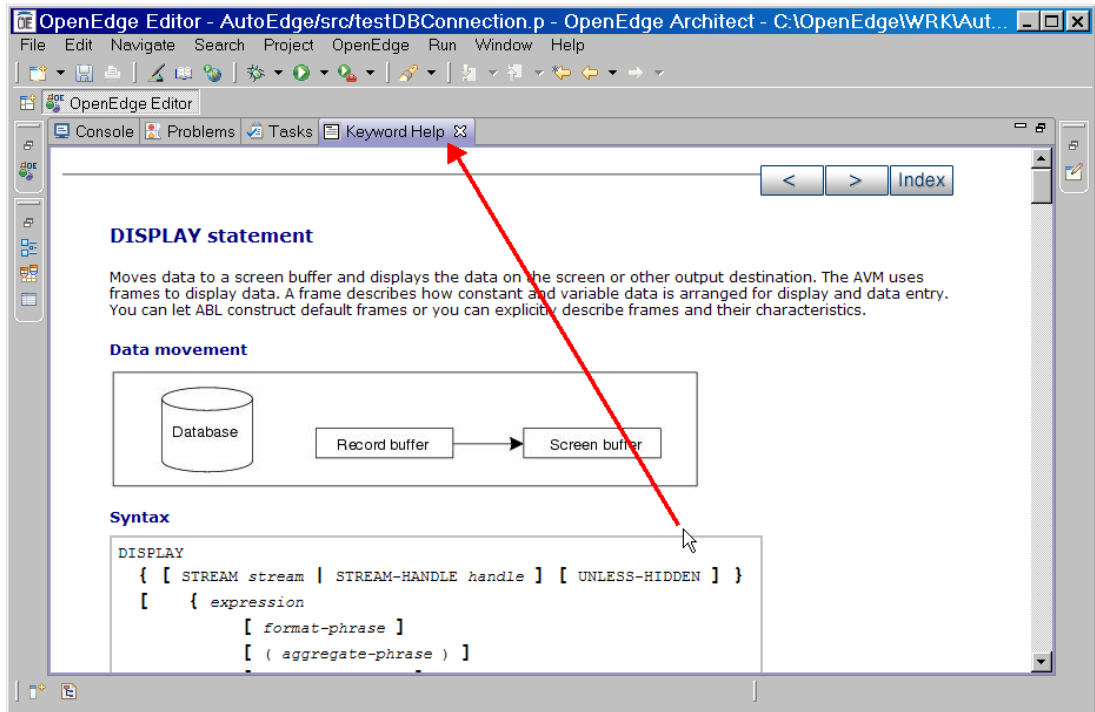
 **Tip:** You can also open the Keyword Help by right-clicking in the editor and then selecting the Keyword Help option.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Using Coding Aids, continued

3. Double-click on the Keyword Help View's tab. This will maximize the view so that its content is easier to read. This functionality is particularly useful when working the ABL Editor.



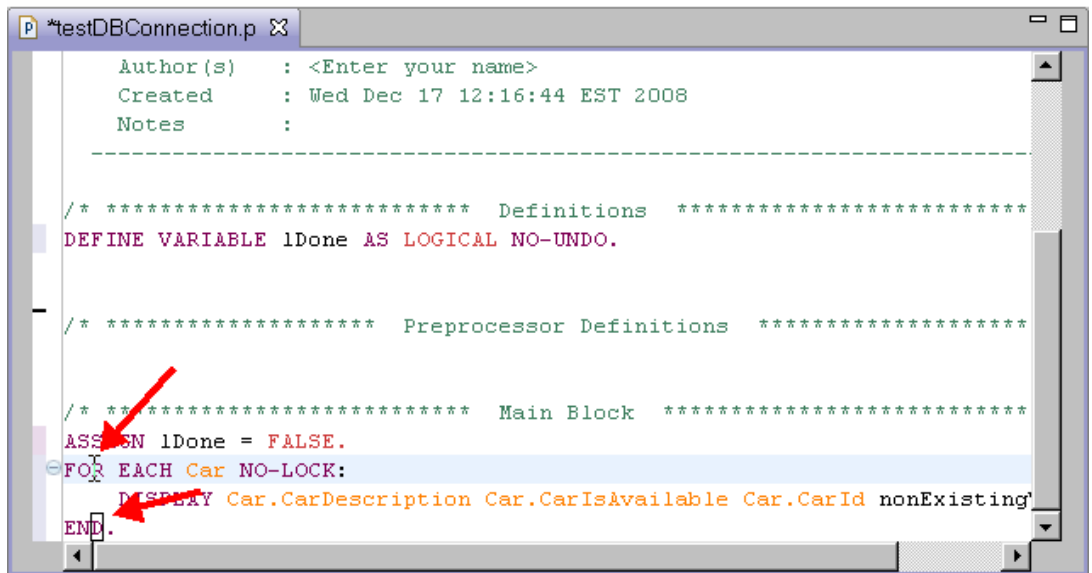
4. Double-click on the Keyword Help View's tab again to return it to its normal size.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Using Coding Aids, continued

5. Click on the **FOR** keyword. A black rectangle is placed at the closing keyword (**END**) of the block. This feature, which identifies where blocks are terminated, comes in very handy in a poorly indented program where it is difficult to locate the end of a block.




```
Author(s) : <Enter your name>
Created   : Wed Dec 17 12:16:44 EST 2008
Notes    :


-----

/* ***** Definitions *****
DEFINE VARIABLE lDone AS LOGICAL NO-UNDO.

/* ***** Preprocessor Definitions *****

/* ***** Main Block *****
ASSIGN lDone = FALSE.
FOR EACH Car NO-LOCK:
    DISPLAY Car.CarDescription Car.CarIsAvailable Car.CarId nonExisting!
END.
```

 **Tip:** The opposite also works. You can place the cursor on the **END** keyword and the editor will place a block where the code block begins.

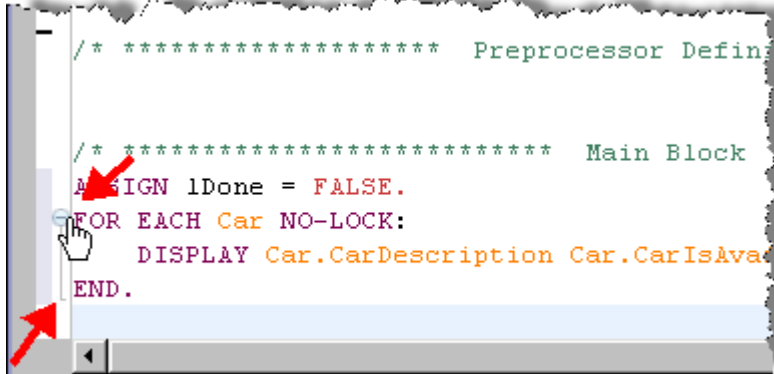
 **Note:** This blocking functionality is common in Architect, and works not only for code blocks, but also for quotation marks and brackets. Also, with quotes and brackets, once you type in the starting quote or bracket, Architect adds a matching quote or bracket respectively. This option can be configured in Preferences under the OpenEdge Architect→Editor→Assistance option.

continued on next page

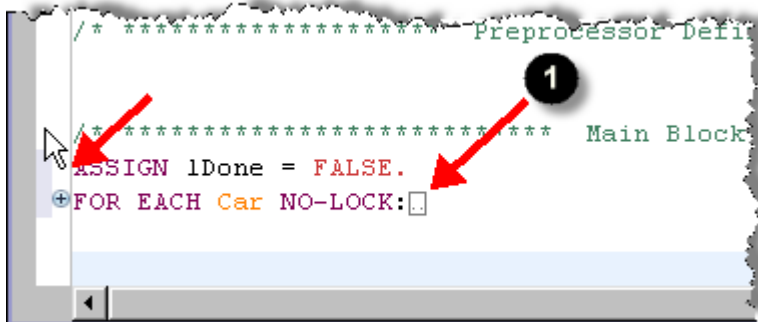
Lab 7 – Using the OpenEdge Editor, continued

Using Coding Aids, continued

6. Another way to determine a blocks scope is to use the marker bar. Place the cursor over the minus (-) sign in the marker bar next to the FOR keyword. A line is displayed from the minus sign to the last line in the block.



7. The minus sign can also be used for code folding to hide blocks of code. Click on minus sign. This collapses (folds) the block so that only one line is visible. The indicator in the marker bar changes to a Plus (+) sign.



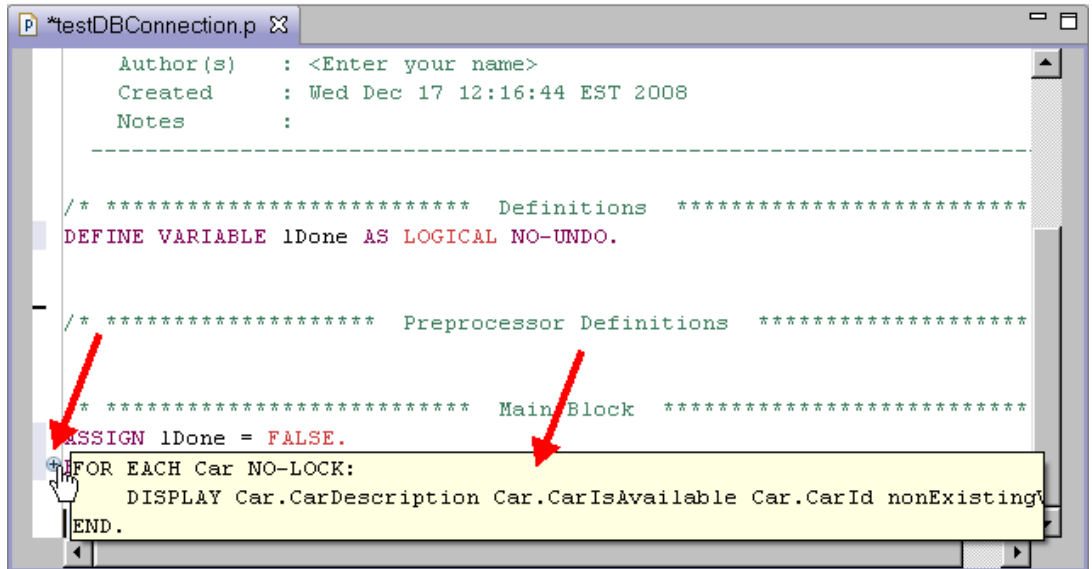
This feature is very useful in a large program. By collapsing large blocks of code that you are not currently working with, you can focus on the areas of code that need attention. The ellipse indicator (1) shows that the current line has been collapsed and that more code exists than that which is shown.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Using Coding Aids, continued

- Place the cursor over the Plus sign. The first few lines of hidden code are displayed. This is useful if you need to quickly view the code contained in a hidden block.



The screenshot shows a window titled '*testDBConnection.p'. The code is as follows:

```
Author(s) : <Enter your name>
Created   : Wed Dec 17 12:16:44 EST 2008
Notes     :
-----

/* ***** Definitions *****
DEFINE VARIABLE lDone AS LOGICAL NO-UNDO.

/* ***** Preprocessor Definitions *****

/* ***** Main Block *****
ASSIGN lDone = FALSE.
FOR EACH Car NO-LOCK:
    DISPLAY Car.CarDescription Car.CarIsAvailable Car.CarId nonExisting
END.
```

Two red arrows point to the plus sign on the left margin and the yellow highlight box around the 'FOR EACH' block. A mouse cursor is positioned over the plus sign.

- Un-collapse the block by clicking on the Plus sign.

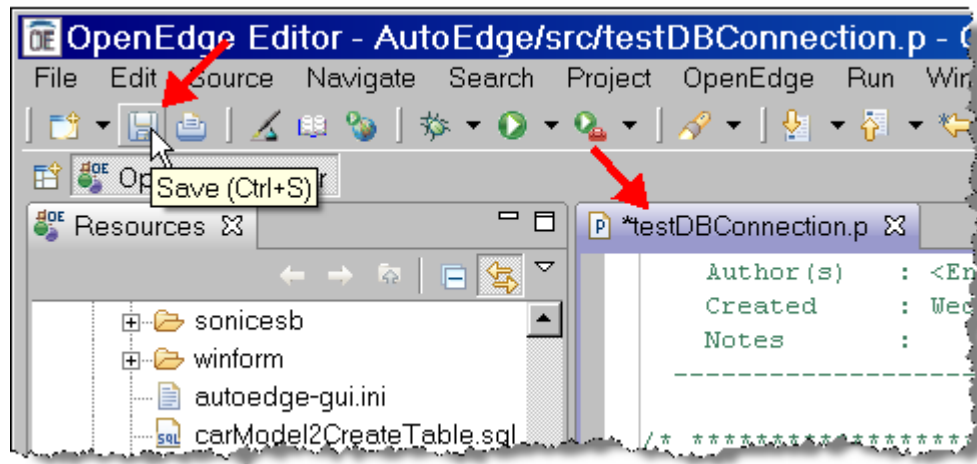
continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Viewing Errors

This section contains an overview of the error reporting features of the ABL Editor and associated views.

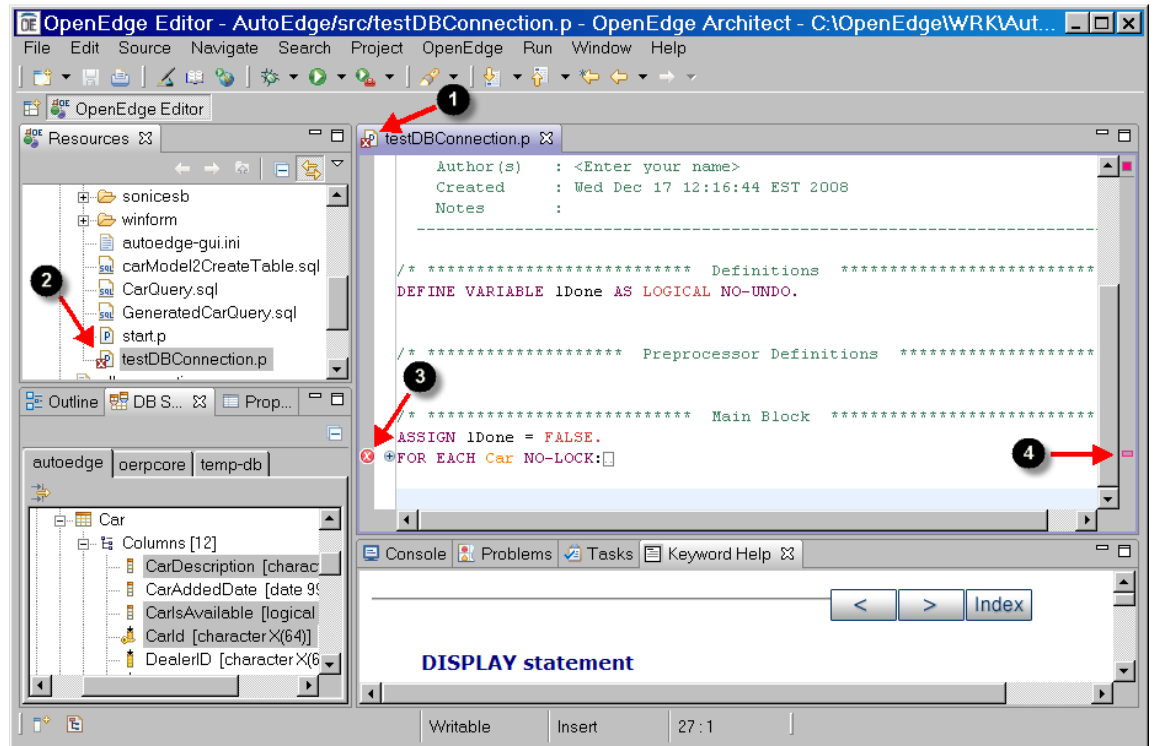
1. Make sure the editor's tab is selected and select the **Save** button. The asterisk next to the program name in the Editor tab is cleared.




continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Viewing Errors, continued




The program will be saved and an attempt will be made to compile the program. If the compile was unsuccessful, error markers will be displayed next to the file's name in the editor tab (❶), next to the file's name in the Resources view and the directory where the resource resides (❷), in the marker bar next to the line where the error(s) occurred (❸), and in the overview ruler (❹).

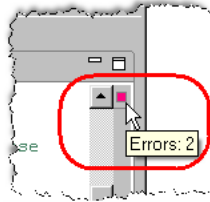
 **Tip:** The marker in the overview ruler (❹) represents the location of the error relative to the entire file. This is useful when you are working in a large file since it allows you to position the viewing area to the location of each error even if the lines of code with the error are not currently visible in the editor.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Viewing Errors, continued

 **Note:** Hovering over the red box in the upper right-hand corner of the editor will display the total number of errors in the program.

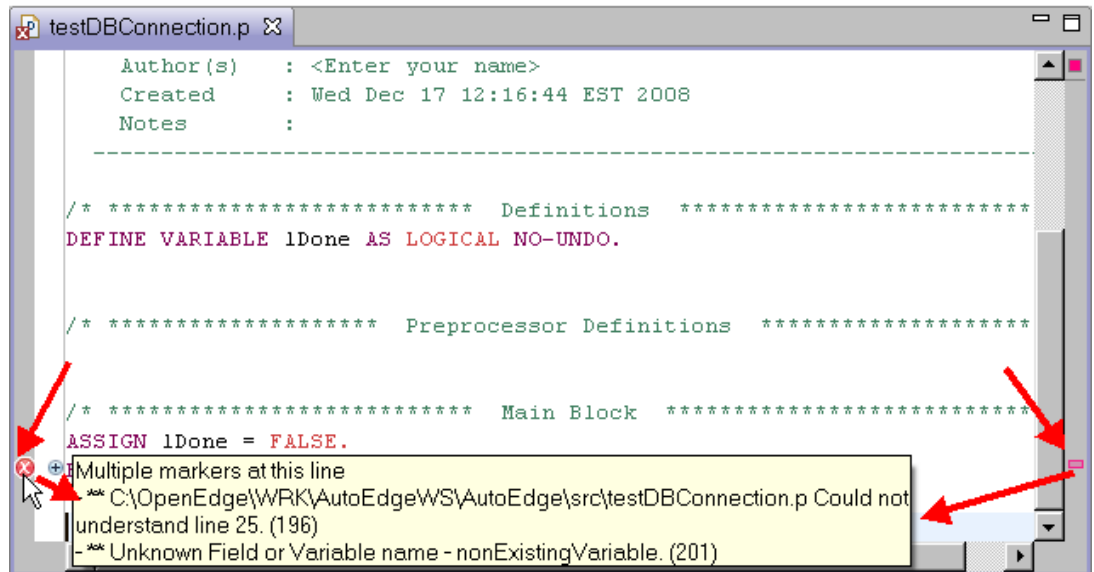



continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Viewing Errors, continued

2. Hover over the error marker in the marker bar or error bar in the overview ruler. The error message for the error will be displayed.



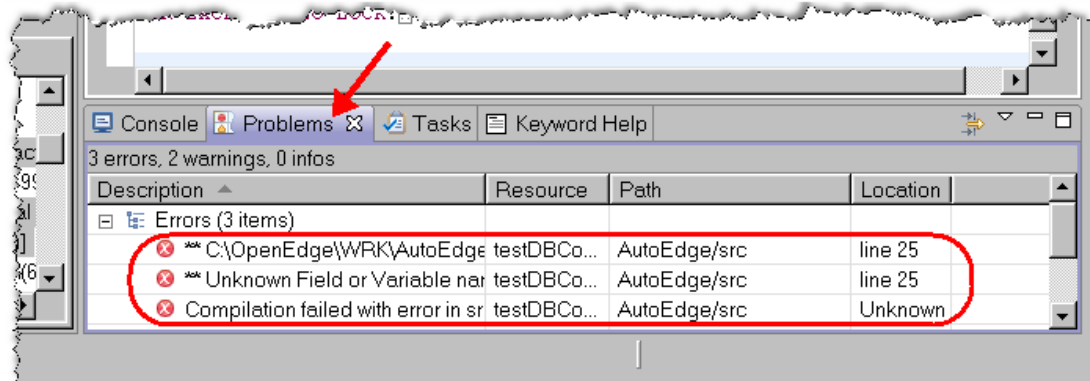
 **Tip:** In many cases it is useful to see the line numbers in the editor. To toggle the line numbers on/off right-click in the Marker bar and select **Show Line Numbers**. A check mark will appear next to that option indicating that line numbering has been turned on. You can also set this option in the Workspace Preferences dialog in the **General→Editors→Text Editors** section.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Viewing Errors, continued

3. Click on the **Problems** view. The Problems view shows a list of all the detected errors and warning messages for the resources contained in the workspace.

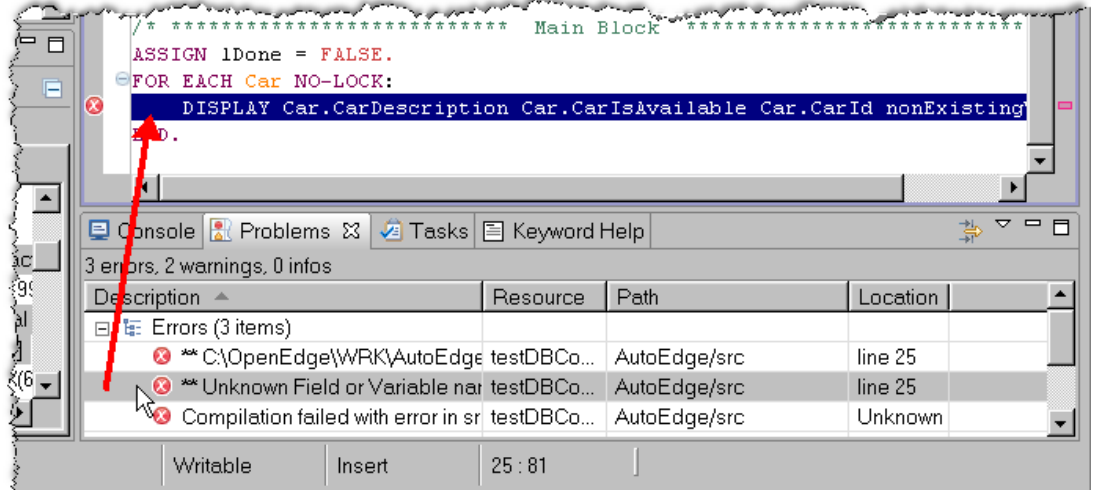


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Viewing Errors, continued

4. Double-click on the **Unknown Field or Variable name** error in the grid. This repositions the cursor to the line of code in the editor with the error. If you click on an error for a resource that is not already opened, the file will also be opened.



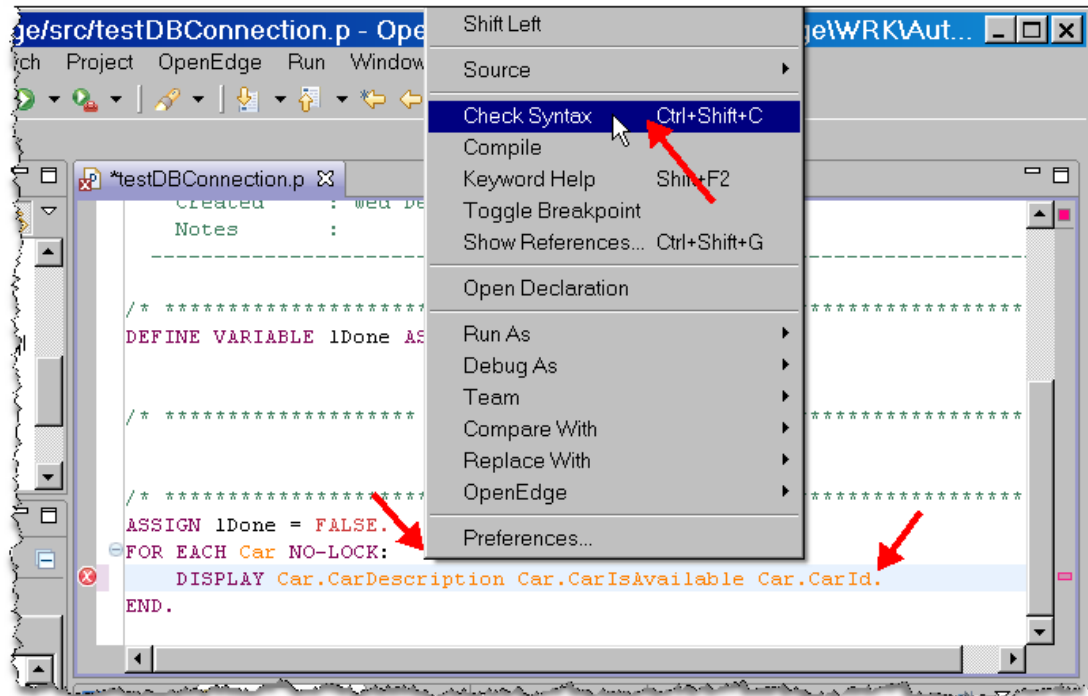
5. The error in testDBConnection.p is the unknown variable named NonExistingVariable in the DISPLAY statement. Delete the variable from the DISPLAY statement.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Viewing Errors, continued

6. Right-click in the editor area and select **Check Syntax** to verify that the syntax was fixed. A message appears showing that the syntax is correct. Click **OK** to continue.

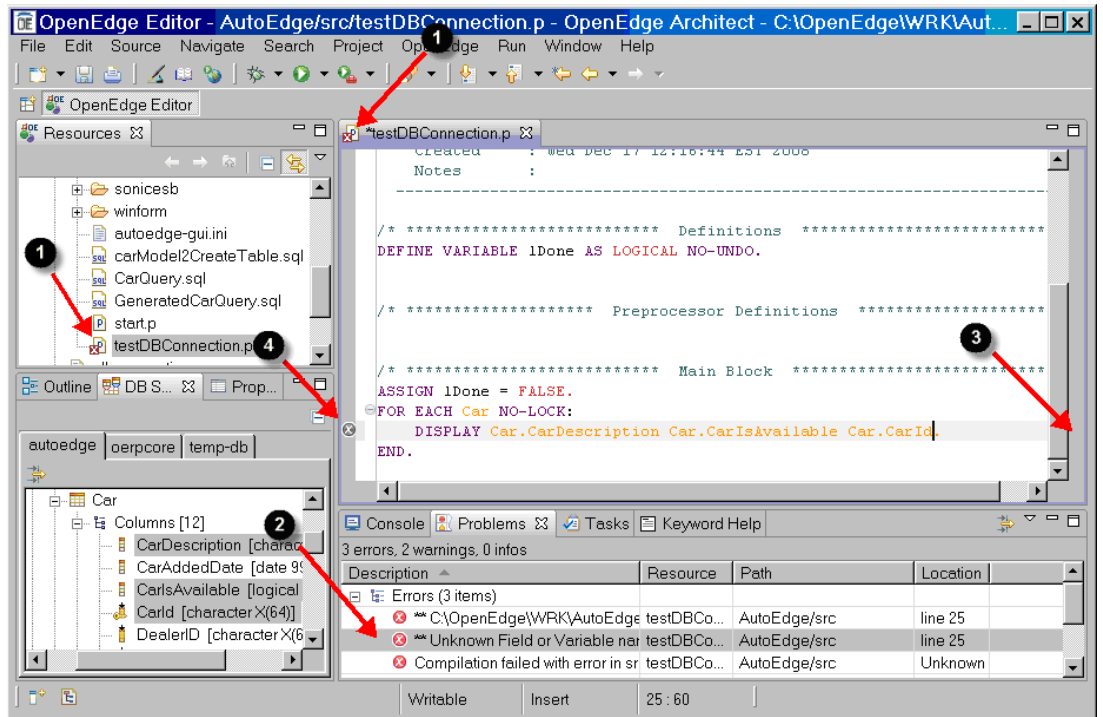


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Viewing Errors, continued

7. After performing the syntax check, the error screen decorator for the file (❶) and the error in the Problems view (❷) will still be shown. This is because these markers show the status of the file that is saved to disk and this file has not been saved yet. However, the error bar (❸) will be removed and the error marker (❹) in the marker bar will be grayed out.



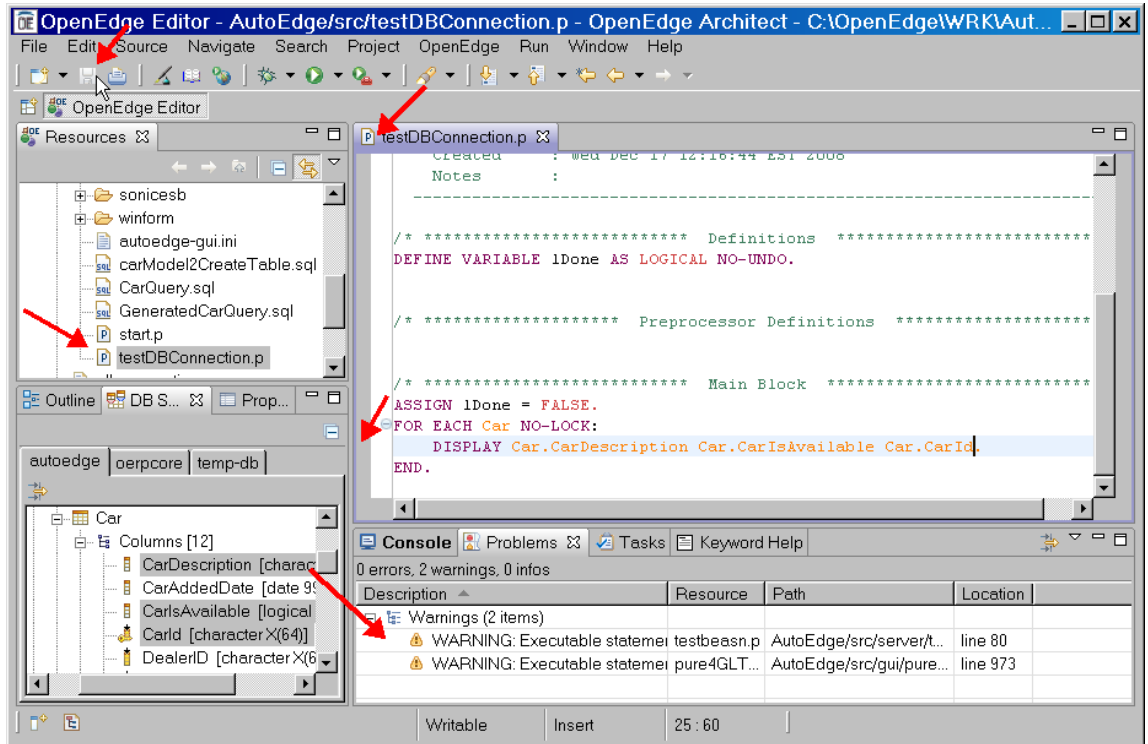
Note: Architect also only runs files that are saved on the disk, not from the in-memory copy that is visible in the Editor. The saved procedure on disk still contains the error and would fail if executed.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Viewing Errors, continued

8. Save the file.



All of the error markers and errors in the Problems view will be removed.

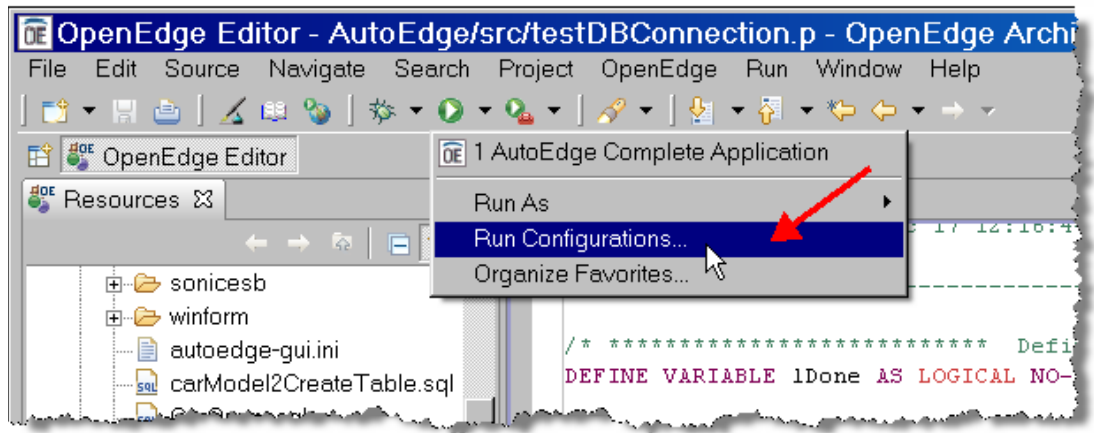
continued on next page

Lab 7 – Using the OpenEdge Editor, continued

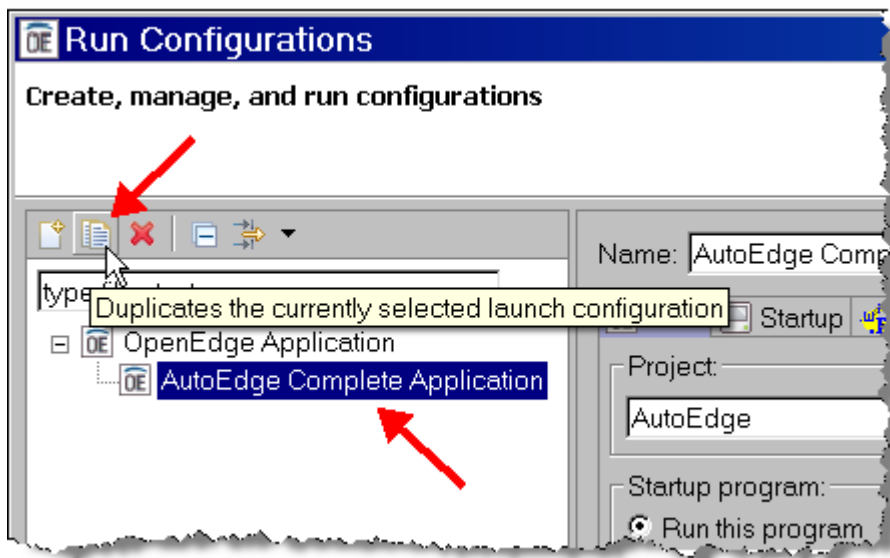
Running code

In a previous lab, you set up a Run Configuration that started the entire AutoEdge application. This section shows how to set up a configuration that executes just the procedure you are working on.

1. Select the down arrow next to the Run button and select the **Run Configurations** option. The Run Configurations window will open.



2. Select the existing **AutoEdge Complete Application** entry that you created in a previous lab. Click the Copy button to copy the configuration. This will duplicate all of the previous settings without having to set them manually in the new configuration.

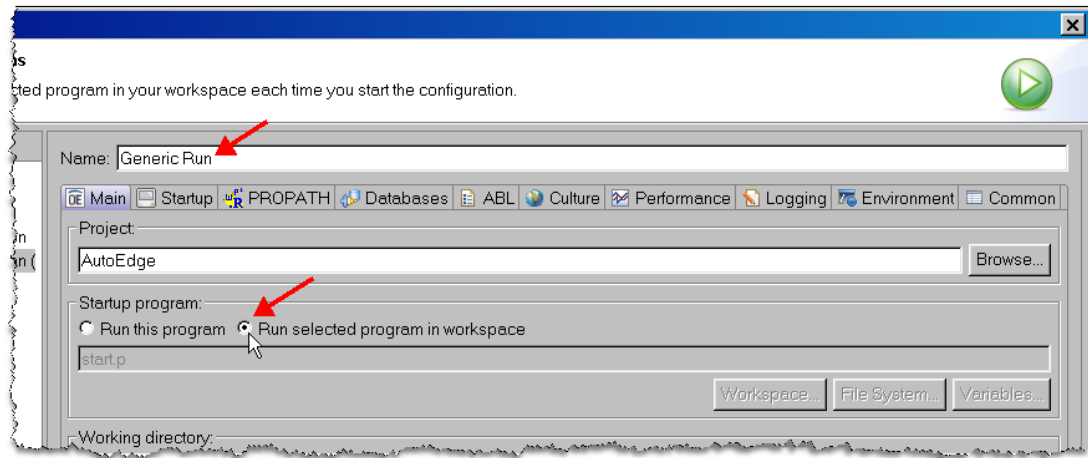


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Running code, continued

3. In the Main tab, change the new configuration name to **Generic Run**. Select the **Run selected program in workspace** option in the Startup program group.

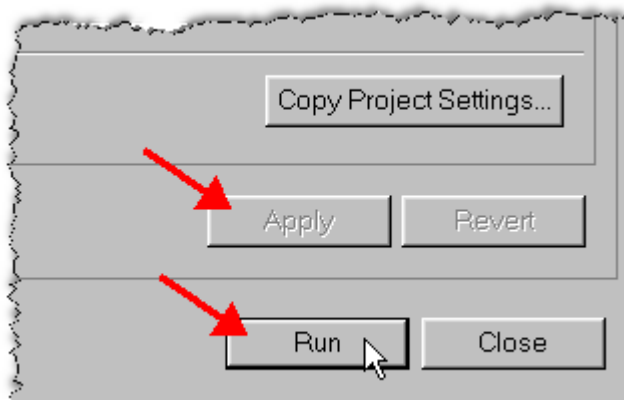


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Running code, continued

4. The rest of the settings, such as the PROPATH, were copied over from the other configuration, so click the **Apply** button to save the configuration.
5. Click the Run button.

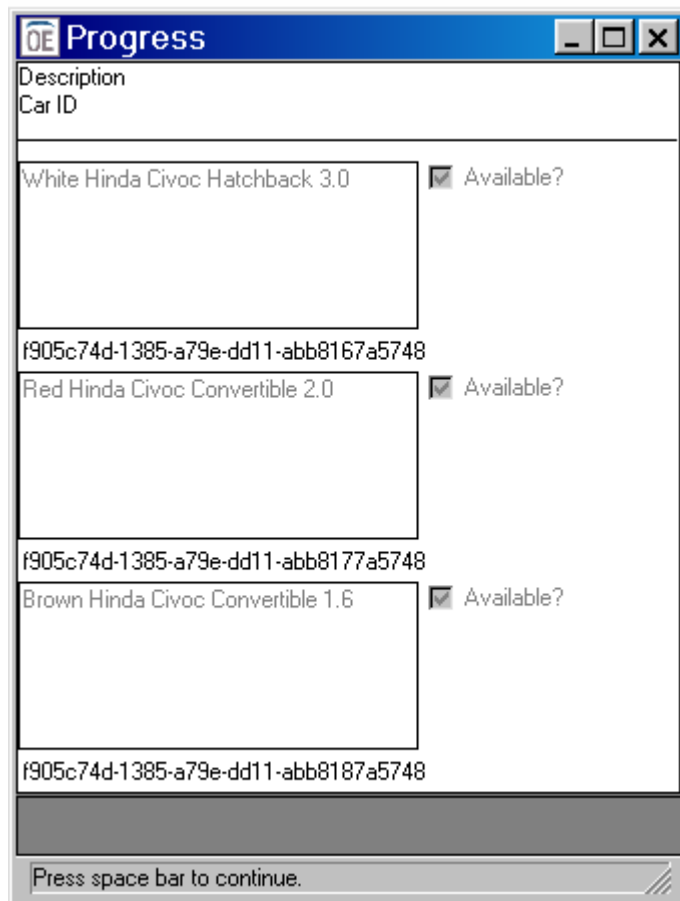


continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Running code, continued

6. The program runs and data from the database is displayed. If you encounter an error, review the Problems and Console views to diagnose the problem.



7. Press **Esc** and then the **Spacebar** at any time to close the running window.
8. Close the source file in the editor.

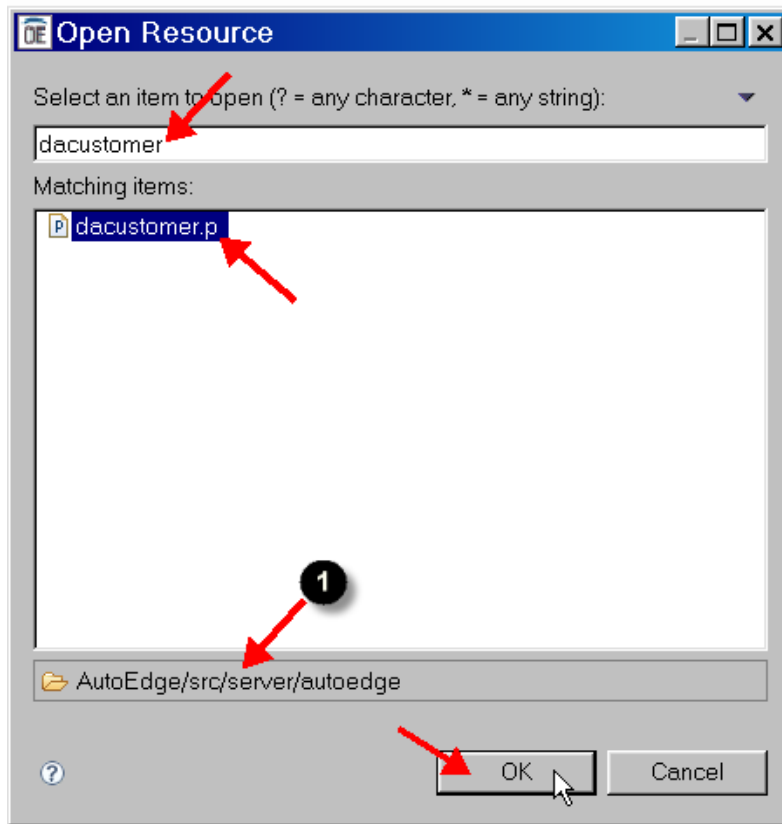
continued on next page

Lab 7 – Using the OpenEdge Editor, continued


Modifying and existing resource

This section will add validation logic to the Contact Email address column and adds the Customer Email field (added to the database in an earlier lab) to the code for future use.

1. Click on Resources view and either press **Ctrl+Shift+R** or select **Navigate→Open Resource** from the Architect menu.
2. In the Open Resource dialog, enter **dacustom** in the text field. As you are typing the characters, resource files will be filtered to match the characters you are typing.
3. Select the **dacustomer.p** file and click **OK**. The file will be opened in an editor.



 **Note:** The In folders area (❶) will display the location of the selected file.

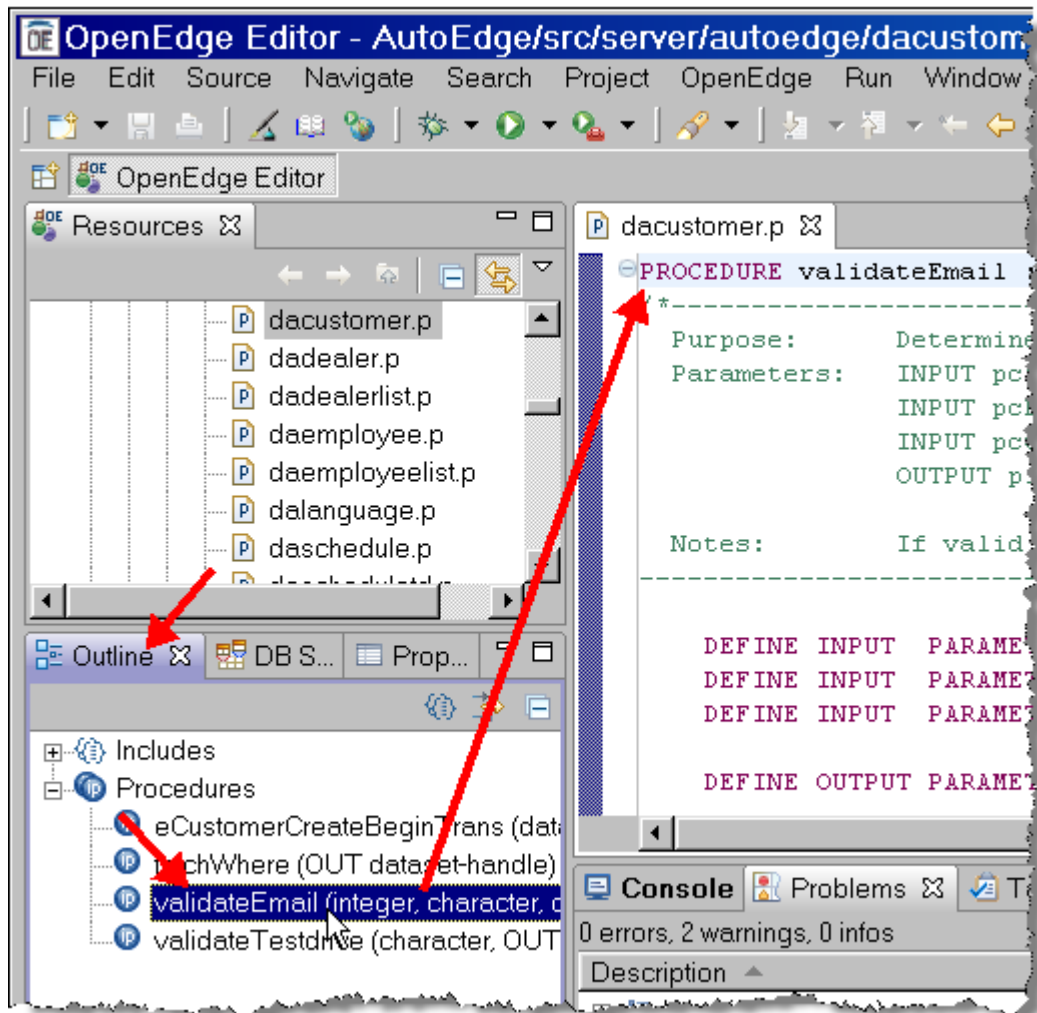
 **Tip:** This method of opening a resource file is faster than searching through the Resources view's tree-view for the resource—as long as you know the name of the file you are looking for.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Modifying and existing resource, continued

4. Click on the **Outline** view. Expand the **Procedures** node and click on the **validateEmail** entry. Doing will reposition the editor to that procedure.



continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Modifying and existing resource, continued

5. Add the code that is highlighted below after the first ASSIGN statement (you can copy and paste the code from the following document):

Procedure: validateEmail	Program: dacustomer.p
...	
ASSIGN plEmailValid = TRUE.	
<pre>IF pcEmailId <> "" AND NOT pcEmailId MATCHES "*@*~.*" THEN DO: ASSIGN plEmailValid = FALSE. MESSAGE "Email entered is invalid" VIEW-AS ALERT-BOX. END.</pre>	
IF pcEmailId <> "" THEN	
...	

6. Save the file.

continued on next page

Lab 7 – Using the OpenEdge Editor, continued

Modifying and existing resource, continued

7. Press **Ctrl+Shift+R** in the Resources view or select **Navigate→Open Resource** from Architect's menu to open another resource file.
8. Using the Open Resource dialog, open the **etcustomer.i** file.
9. Add the EmailAddress field to the eCustomer TEMP-TABLE as highlighted below:

```
Program: etcustomer.i
...
DEFINE TEMP-TABLE eCustomer BEFORE-TABLE eCustomerBefore

/* fields from customer table */
FIELD CustomerID           AS CHARACTER LABEL "CustomerID" ...
FIELD CustomerFirstName    AS CHARACTER LABEL "First (Give ...
FIELD CustomerMiddleName  AS CHARACTER LABEL "Middle In ...
FIELD CustomerLastName     AS CHARACTER LABEL "Last(Fami ...
FIELD EmailAddress         AS CHARACTER FORMAT "X(64)"
FIELD BaseCodeSalutationID AS CHARACTER LABEL "Salutat ...
...
```

10. Save the file.
-

Lab 8 – Using the OpenEdge Debugger

Overview

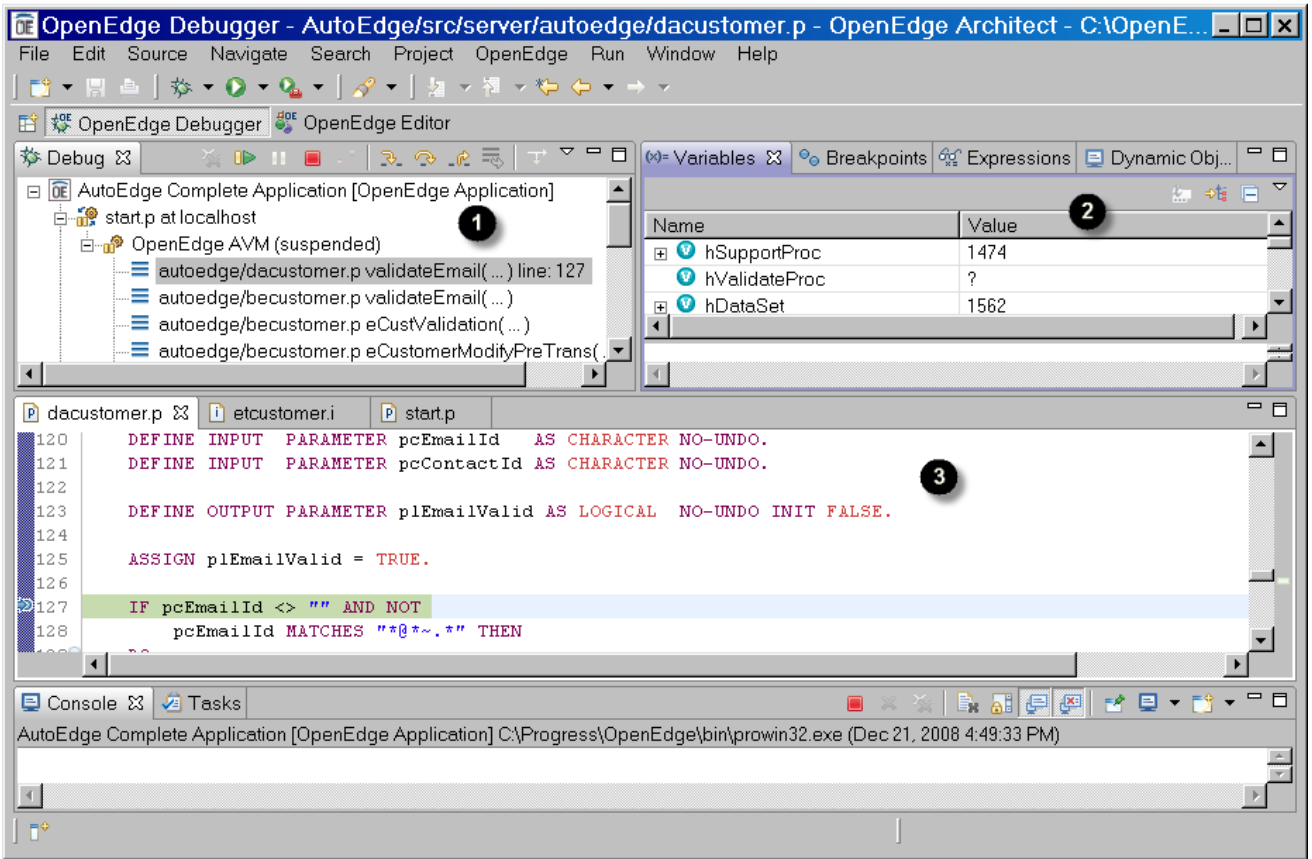
The OpenEdge Debugger is fully integrated into the Architect environment and works seamlessly with other tools in the environment. It also has an intuitive interface that allows for easy organization of debugging data. Multiple debug sessions can be run simultaneously and the tool provides source code viewing while debugging an application.

This lab provides an introduction to the OpenEdge Architect Debugger and how to use some of the available debugging features.

continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

OpenEdge Debugger Perspective



Notes:

- 1 Debug view showing execution stack.
- 2 Information views including Variables, Breakpoints, Expressions, and Dynamic Objects views.
- 3 Editors with current line and debugging markers.

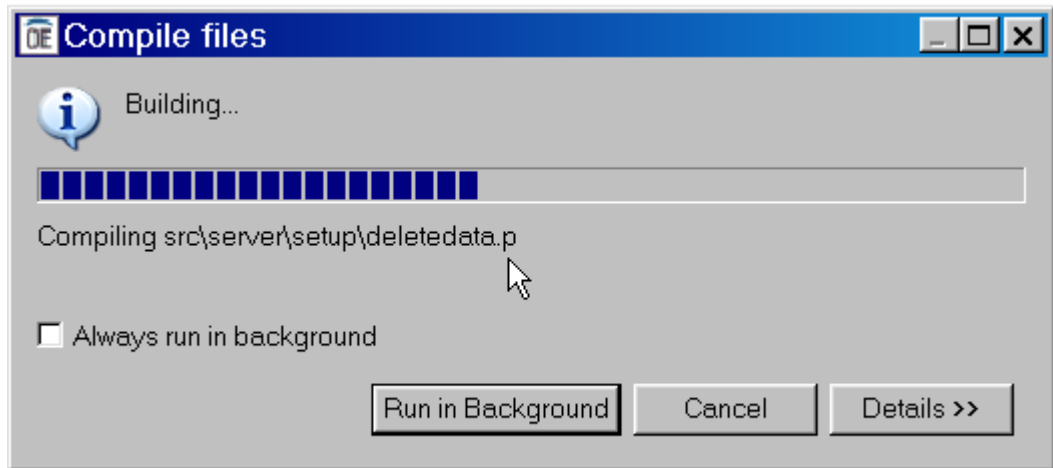
continued on next page


Lab 8 – Using the OpenEdge Debugger, continued


Starting the Debugging Process

This section shows you how to set up and start the debugging process.

1. To make sure that all files are compiled and up to date (especially since changes were made to an include file in the last lab), right-click on the **AutoEdge** project in the Resources view and select **OpenEdge→Compile**. A progress screen will be displayed as the files are compiled.



 **Note:** A program must be compiled before it can be executed in the Debugger. If r-code is not available for the file you are trying to debug, you will receive an error message. Also, make sure that you have saved any changes you have made to any files in the editor and recompile them. Otherwise, you could end up with old r-code being executed by the Debugger.

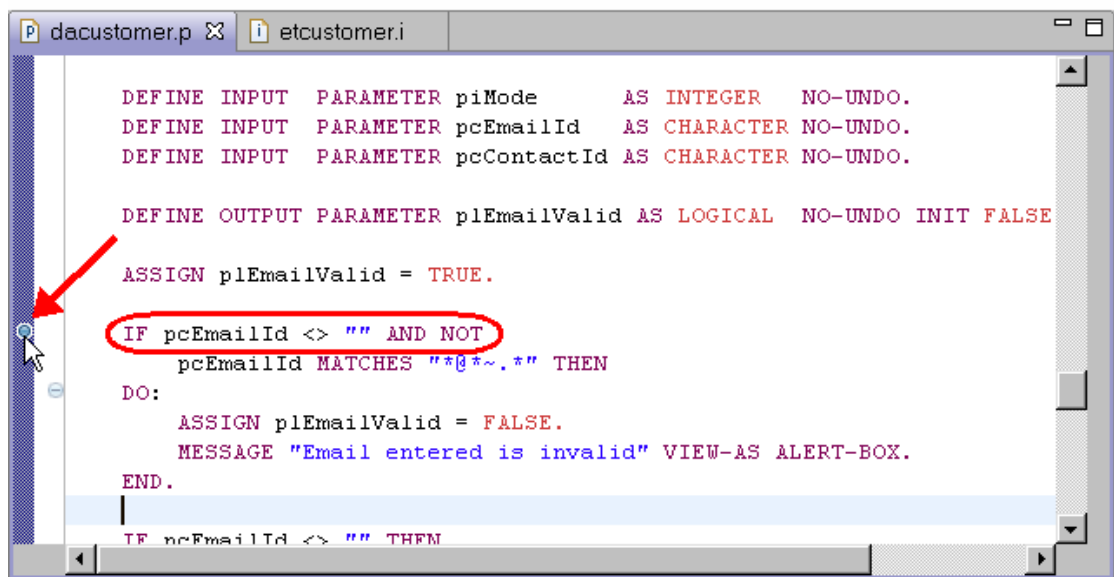
 **Note:** You may notice that the terms compile and build are being used in Architect. The Eclipse framework uses the word build, while OpenEdge has traditionally used the term compile. For the purposes of this workshop, these two terms have the same meaning.

continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Starting the Debugging Process, continued

2. If not already opened, open the **dacustomer.p** resource file.
3. Navigate to **validateEmail** internal procedure (you can use the Outline view to help).
4. Double-click on the debug marker area as shown below. This sets a debugging break point for the program at that location.

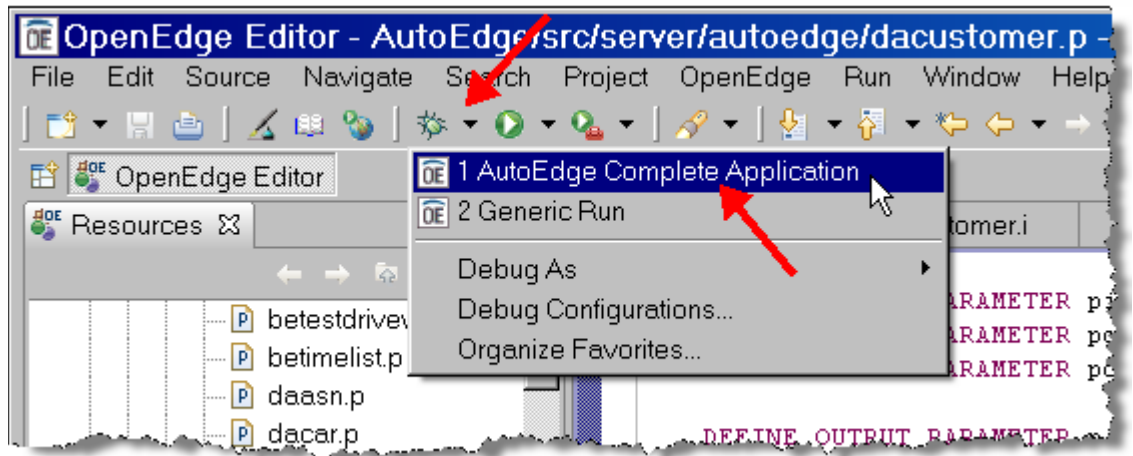



continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

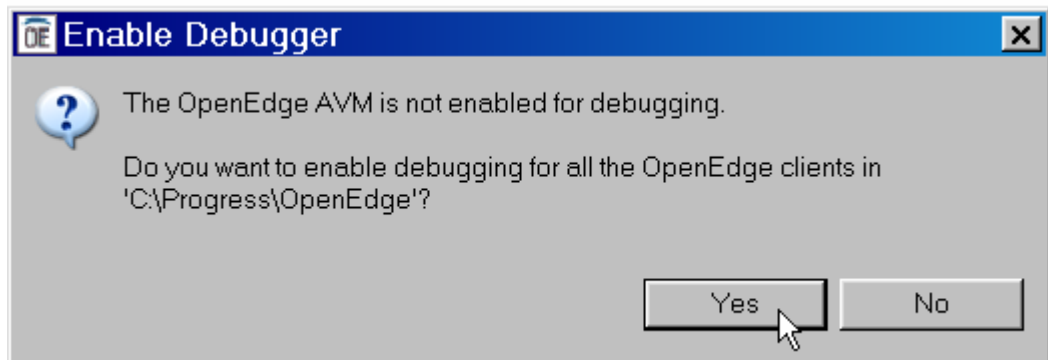
Starting the Debugging Process, continued


5. Select the down-arrow next to the Debug button in the Architect toolbar. Select the **AutoEdge Complete Application** option. This will start the debugger using the Run Configuration you set up previously to run the entire AutoEdge application.



 **Note:** Clicking on the debug or run buttons (instead of the button's drop-down menu) automatically launches the last used configuration.

6. If this is the first time you have run the debugger, you will be asked to enable debugging for all OpenEdge clients. Architect is not shipped with debugging enabled for security reasons. Click **Yes** to enable debugging.



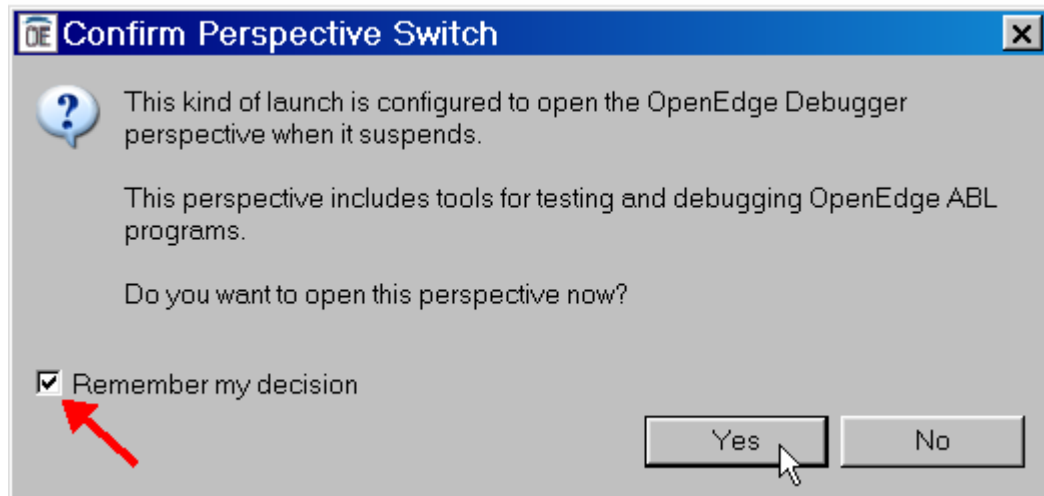
 **Note:** You can enable debugging from the console by typing **-debugenable -enable all** while in the <OpenEdge Install>/dlc directory. Also, under certain circumstances, you may need to restart Architect for debug enabling to take affect.

continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Starting the Debugging Process, continued

7. A dialog will be displayed confirming that you would like to switch to the OpenEdge Debugger perspective. Check the **Remember my decision** box so that Architect does not ask you this question again. Click **Yes**.



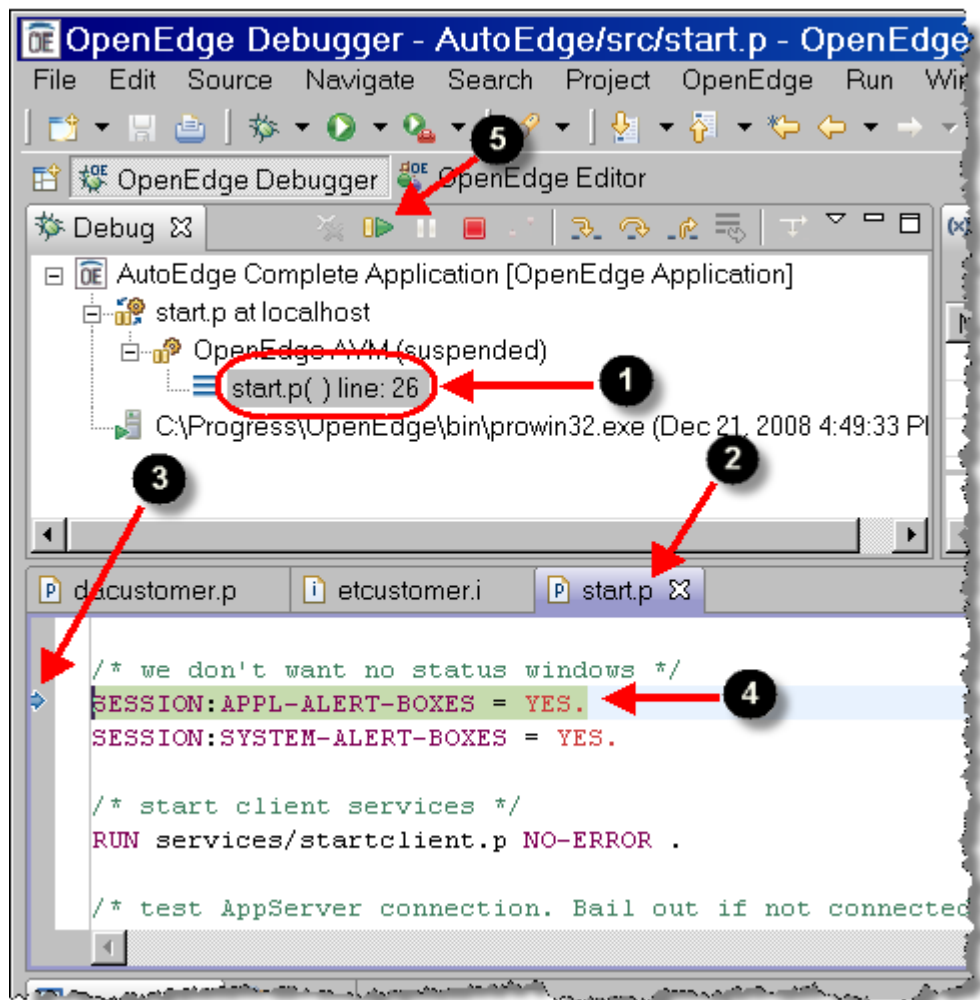
continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Starting the Debugging Process, continued

8. The Debugger will be opened and will immediately suspend execution at the first executable line of code. In this case, the Debug viewer shows that execution has stopped on line 26 of the start.p program (❶). In the editor section, the start.p program is displayed (❷). The current line marker (❸) and blue highlight bar shows where, in the code, execution is currently suspended (❹).


Click on the **Resume** button (❺). The AutoEdge login screen is shown.

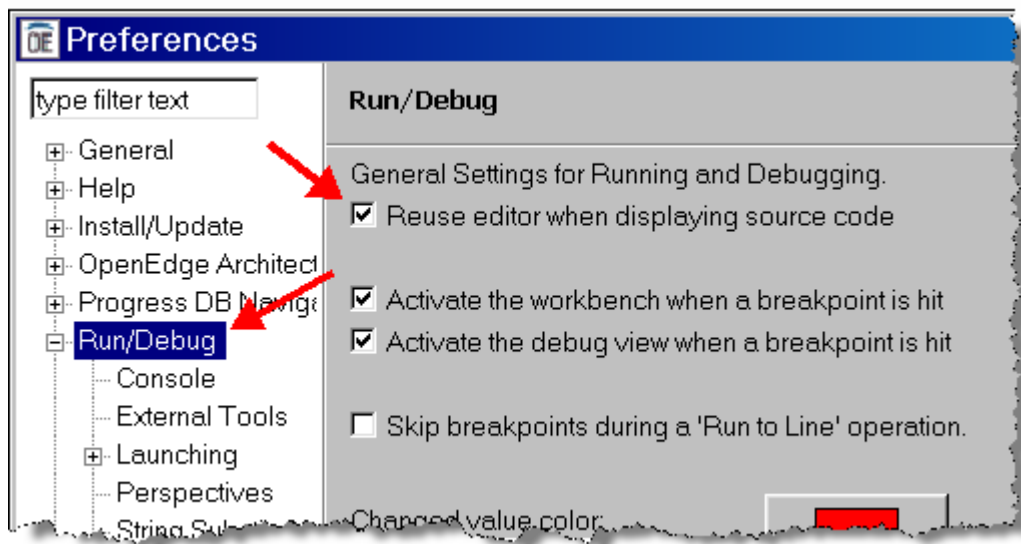



continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Starting the Debugging Process, continued

 **Tip:** An editor is started to show the start.p code because this program was not already open when debugging began. This is a standard feature of the Debugger. During debugging, if more programs are debugged (hit breakpoints) that were not already opened, they will reuse the same Editor. You can specify that each debugged program opens in its own Editor by unchecking the **Reuse editor when displaying source code option** in Preferences under the Run/Debug option.



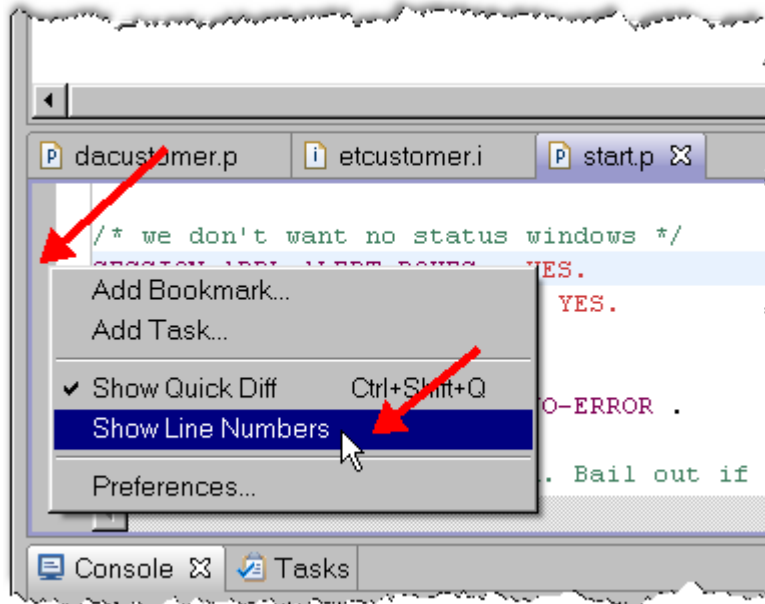
 **Note:** Multiple Debugger sessions can be run at the same time. The Debug view will show root entries [OpenEdge Application] for each Debugger session.


continued on next page


Lab 8 – Using the OpenEdge Debugger, continued

Starting the Debugging Process, continued

9. If not already turned on, toggle on line numbering by right-clicking on the marker bar in the Editor and selecting the **Show Line Numbers** option.



 **Tip:** It can be useful when debugging to see the line numbers; for example, the line number for each program active in the stack is shown in the Debug view and having the line numbers on in the Editor shows the relationship between the Debug view and the Editor.

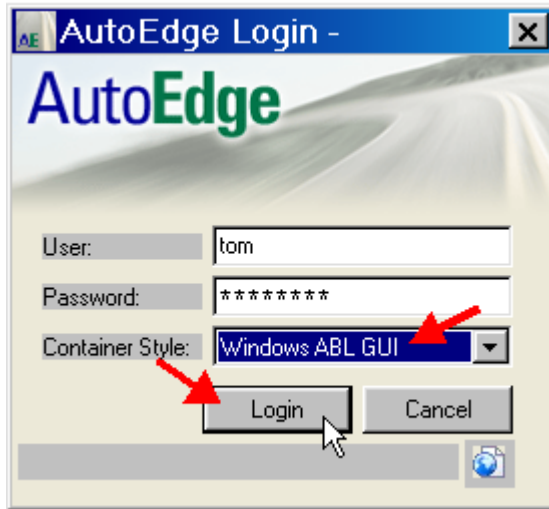
 **Note:** During the following steps, line numbers may be used to help you locate specific sections of code. However, keep in mind that your line numbers may be different depending on how you have typed code in previous labs.


continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

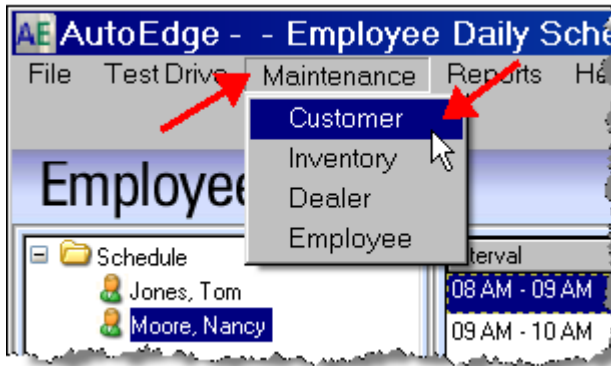
Starting the Debugging Process, continued

10. In the Login window, keep the values for the login User and Password fields but change the Container Style drop-down to **Windows ABL GUI**. Click the **Login** button.



 **Note:** The ability to maintain customer information is not implemented in the OpenEdge GUI for .NET Container Style so the ABL GUI app must be used.

11. Click on **Maintenance** menu and select **Customer**.

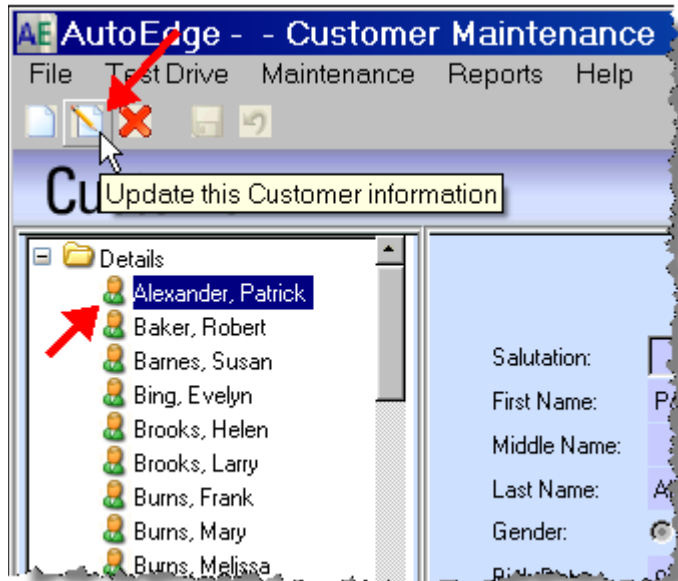


continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Starting the Debugging Process, continued

12. While the **Alexander, Patrick** entry is selected, click on the **Update** button.

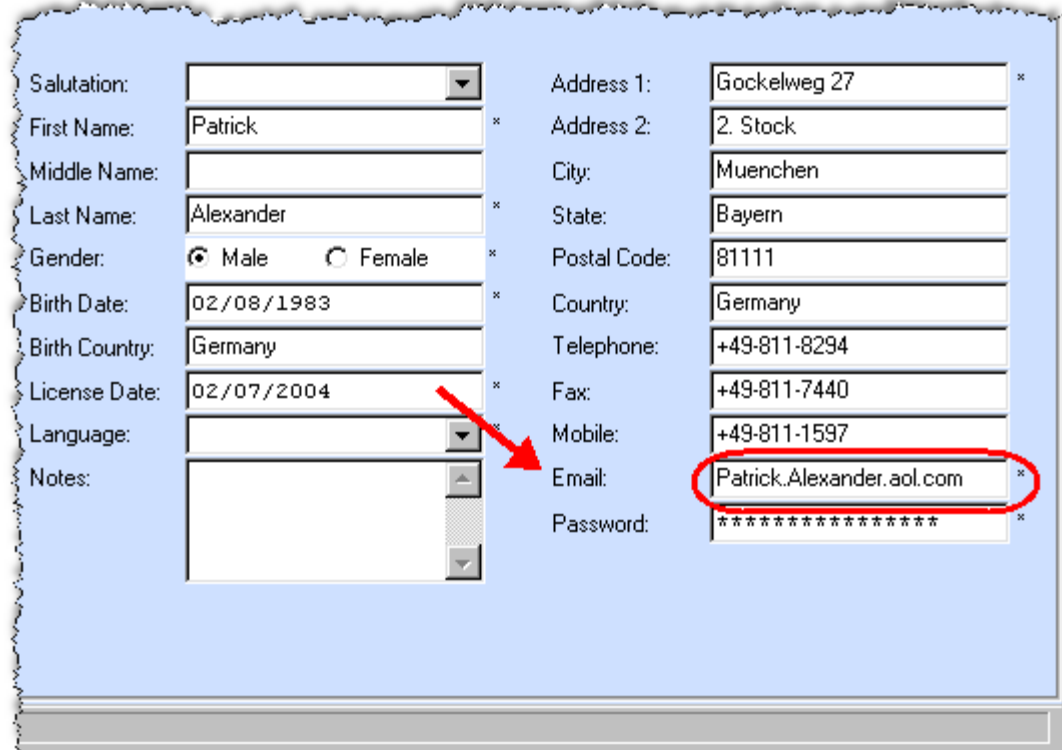


continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Starting the Debugging Process, continued

13. Change the Email field's value to **Patrick.Alexander.aol.com**.



A screenshot of a user profile form with a light blue background and a torn-edge effect. The form contains various fields for personal information. A red arrow points from the 'Language' field to the 'Email' field, which is circled in red. The 'Email' field contains the text 'Patrick.Alexander.aol.com'. The 'Password' field is masked with asterisks. Asterisks are also present at the end of several other text input fields.

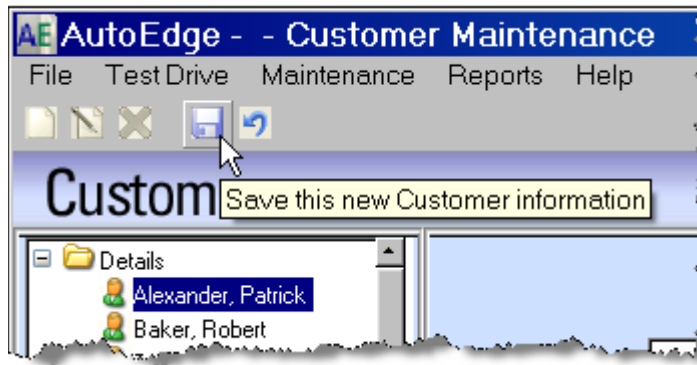
Salutation:	<input type="text"/>	Address 1:	Gockelweg 27 *
First Name:	Patrick *	Address 2:	2. Stock
Middle Name:	<input type="text"/>	City:	Muenchen
Last Name:	Alexander *	State:	Bayern
Gender:	<input checked="" type="radio"/> Male <input type="radio"/> Female *	Postal Code:	81111
Birth Date:	02/08/1983 *	Country:	Germany
Birth Country:	Germany	Telephone:	+49-811-8294
License Date:	02/07/2004 *	Fax:	+49-811-7440
Language:	<input type="text"/>	Mobile:	+49-811-1597
Notes:	<input type="text"/>	Email:	Patrick.Alexander.aol.com *
		Password:	***** *

continued on next page

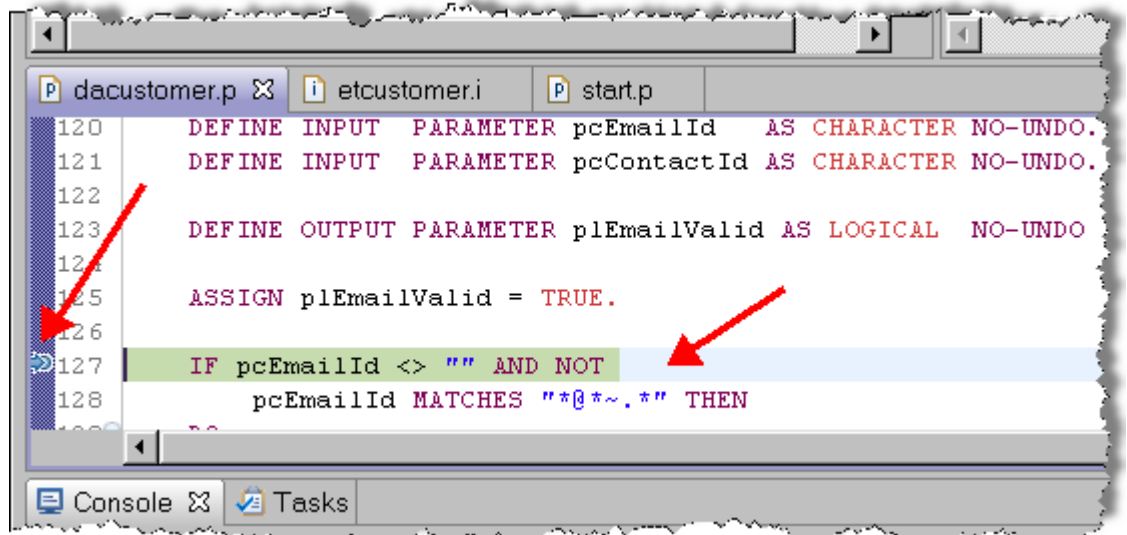
Lab 8 – Using the OpenEdge Debugger, continued

Starting the Debugging Process, continued

14. Click the **Save** button.



The Debugger will be brought to the front because the debugging break point you set in the **validateEmail** internal procedure has been hit. The current line marker will show execution stopped on line where the debugging marker is set and a blue highlight bar will highlight the code on that line.



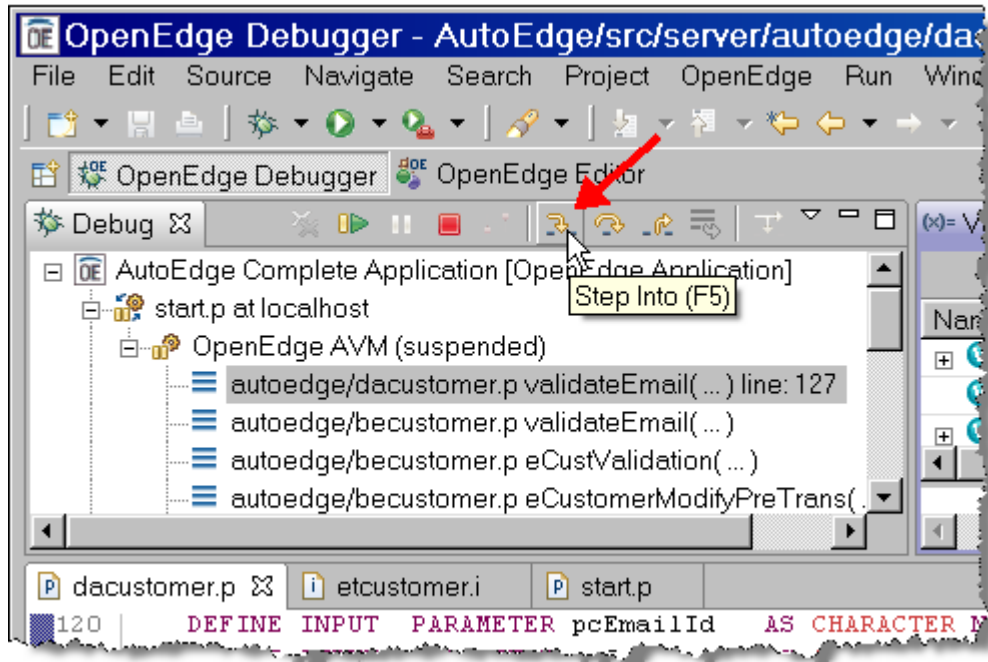
continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

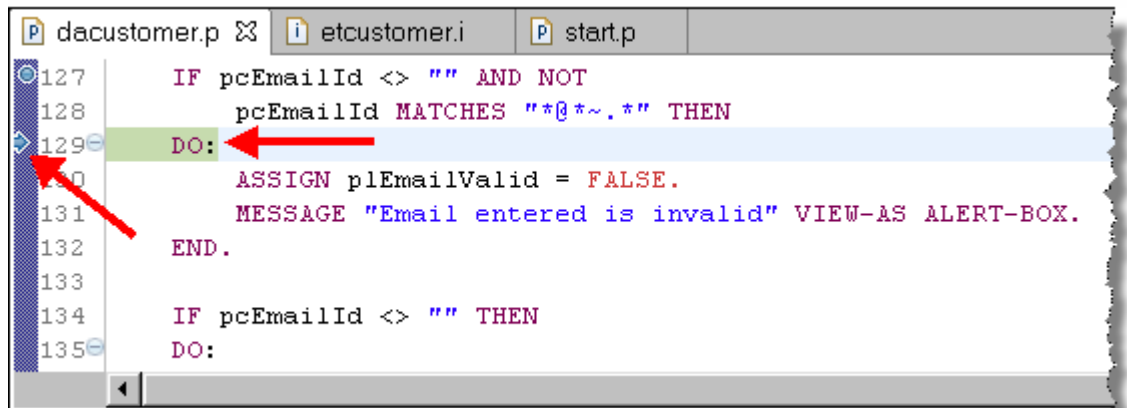
Debugging Code

The Debugger Perspective provides a wealth of information about the application being debugged and action buttons to control the execution of the application. This section covers some of the basic features of the Debug Perspective.

1. Click the **Step Into** button.



The pointer moves to the next executable line of code.

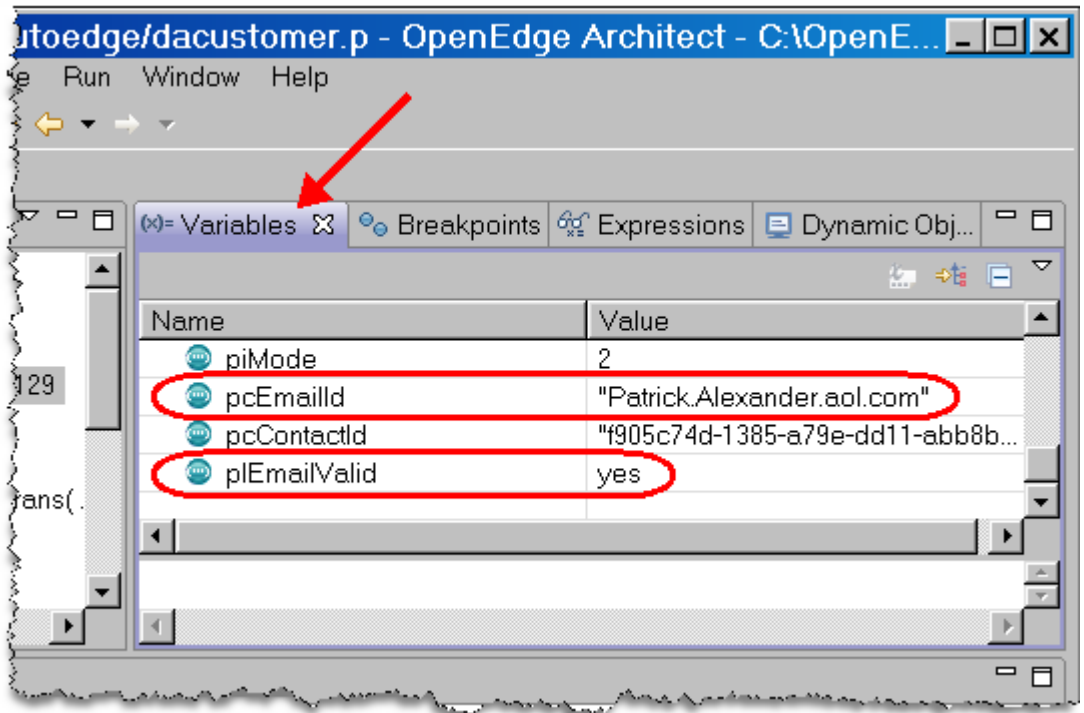


continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Debugging Code, continued

Notice the variables view located (by default) in the upper right portion of the perspective. This view shows all the variables and their run time values. Scroll down the list until you see the **pcEmailId** and **plEmailValid** variables. Notice that the **plEmailValid** variable is currently set to **yes**.

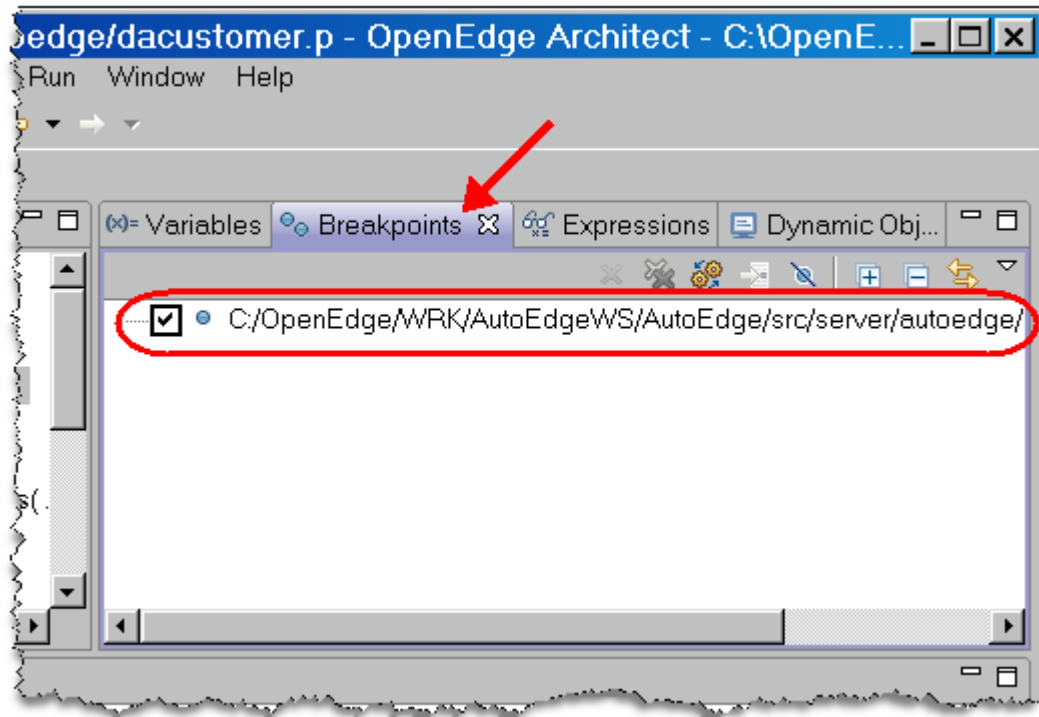



continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Debugging Code, continued

2. Click on the **Breakpoints** tab. All the breakpoints that have been set will be listed here.



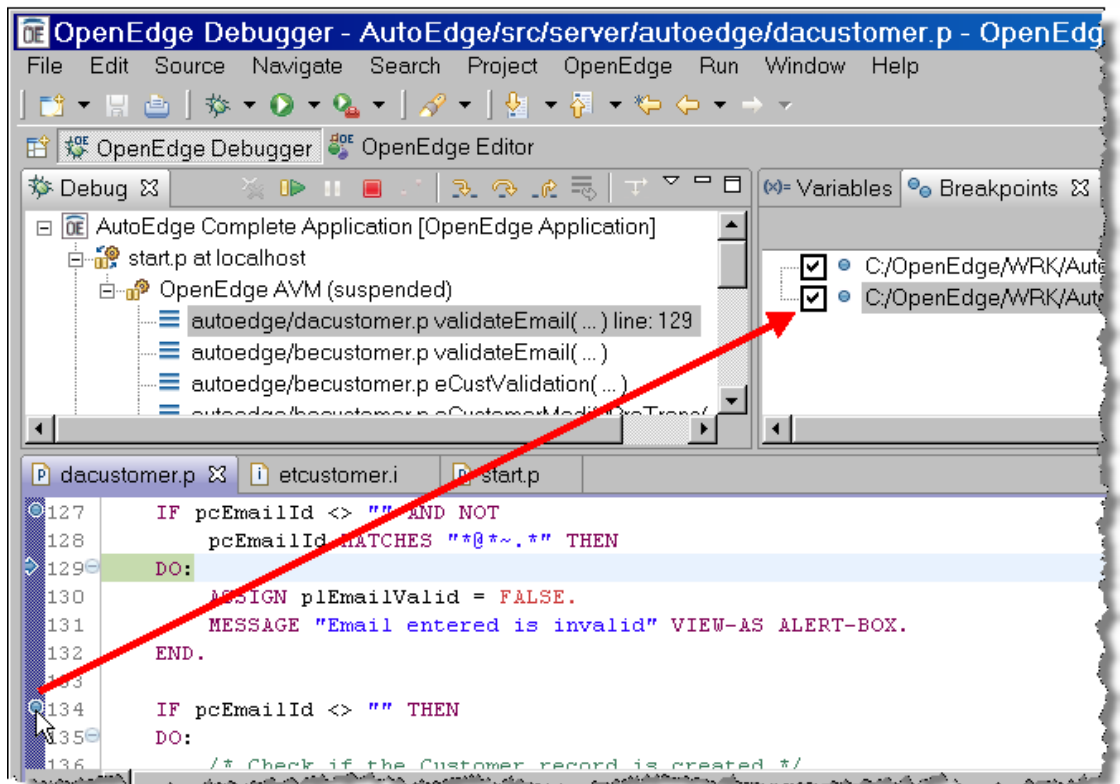
 **Tip:** A workspace keeps track of all the breakpoints that are defined for its resources. You can open the OpenEdge Debugger perspective at any time and switch to the Breakpoints view to see all the defined breakpoints for all the resources in the workspace. Also, you can double-click on one of the breakpoints in the Breakpoints view to open the program in an editor and reposition to the line where the breakpoint is defined.


continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Debugging Code, continued

3. Back in the editor, double-click on line 134 (which is the next IF condition statement if your line numbers do not match). Notice that a new breakpoint has been added to the Breakpoints View. This is a very helpful feature that allows you to add breakpoints while actively debugging an application.



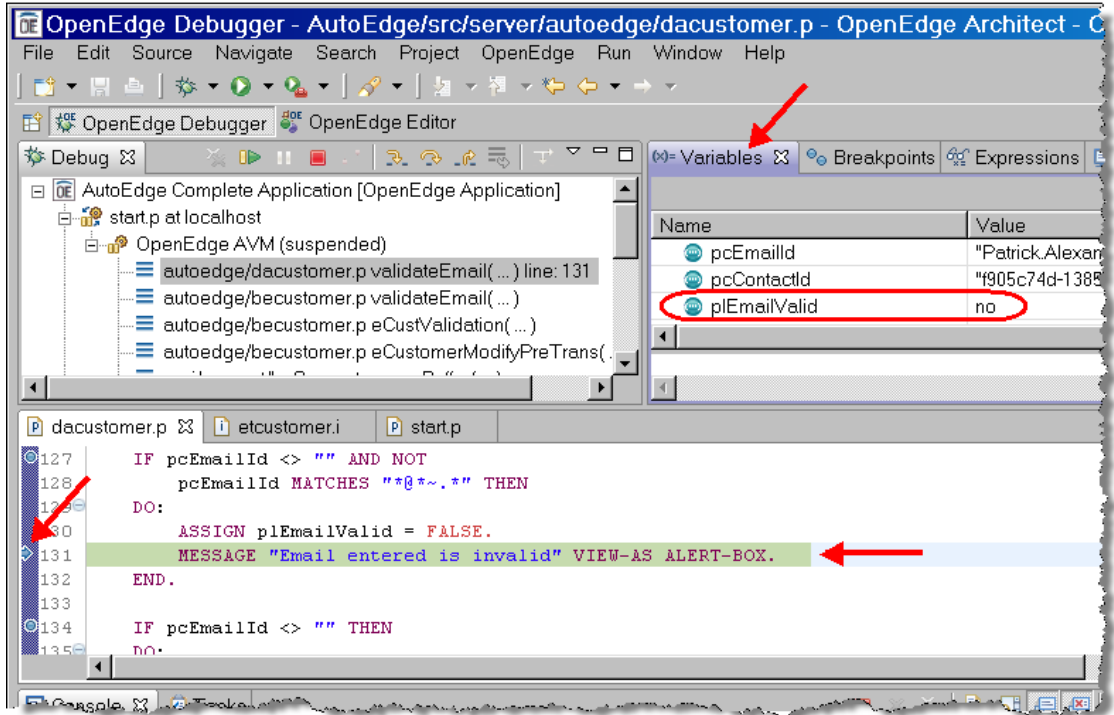
 **Note:** You can turn on or turn off a breakpoint directly from the Breakpoint view by checking or un-checking the box beside the listed breakpoint.

continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Debugging Code, continued

4. Click **Step Into** (twice) until the current line marker goes to the MESSAGE statement (line 132). Click the **Variables** view. The plEmailValid is now set to **no**.

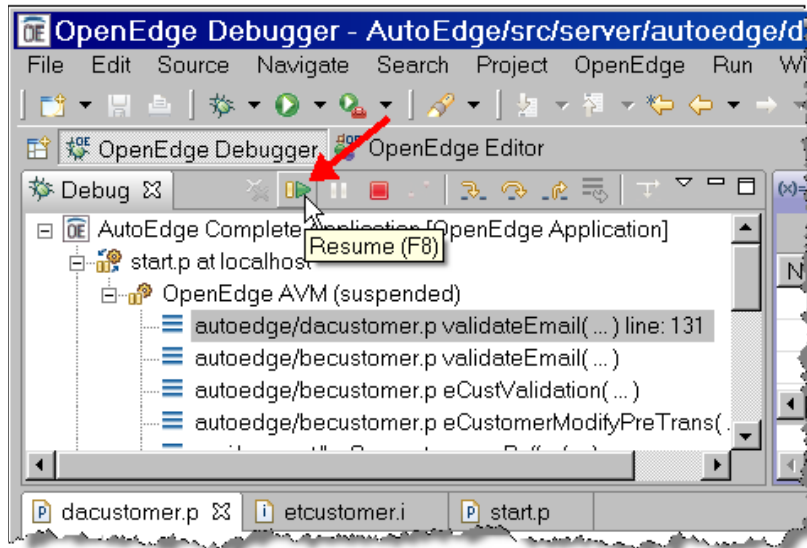


continued on next page

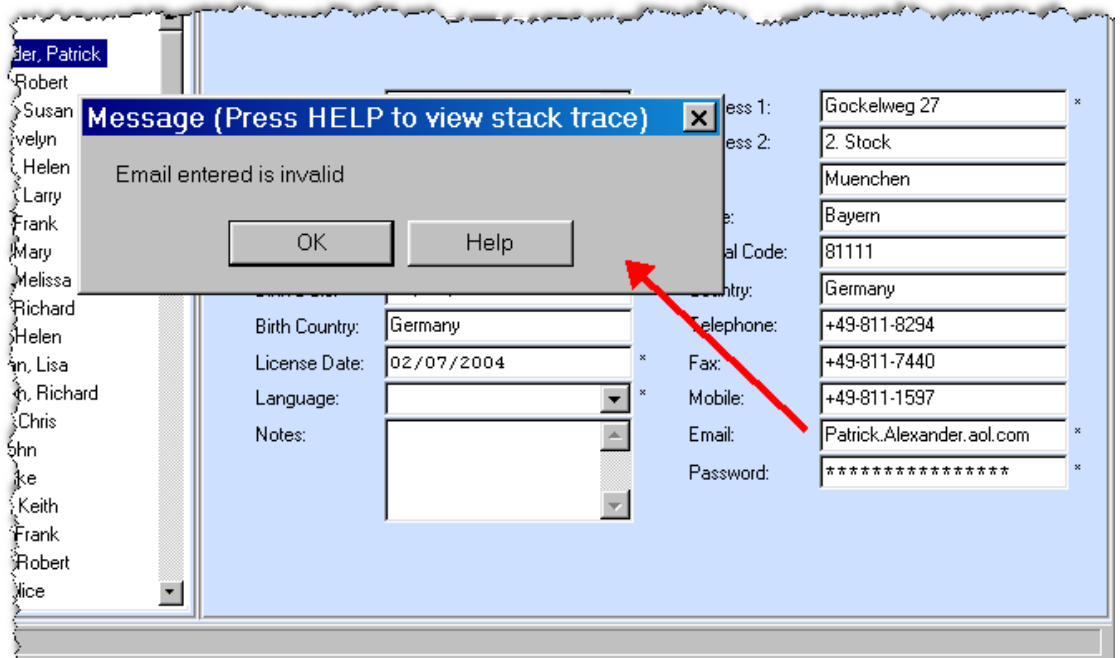
Lab 8 – Using the OpenEdge Debugger, continued

Debugging Code, continued

5. Click the **Resume** button.



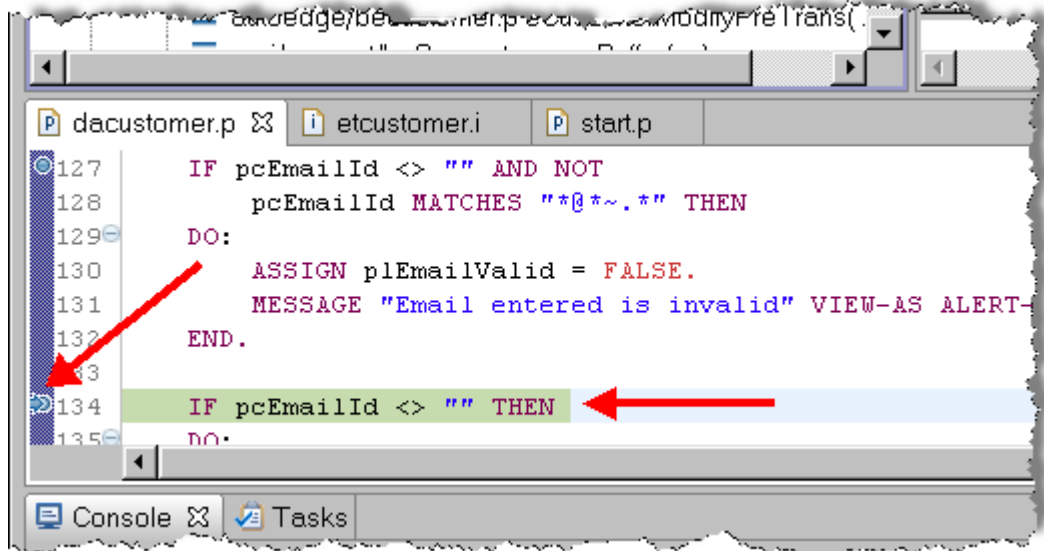
6. The application will show the error message is displayed. You may need to bring the AutoEdge application back into focus (click on the window) to see the message.



Lab 8 – Using the OpenEdge Debugger, continued

Debugging Code, continued

7. Click **OK** to dismiss the error message. The Debugger moves to the new breakpoint that you created in a previous step.



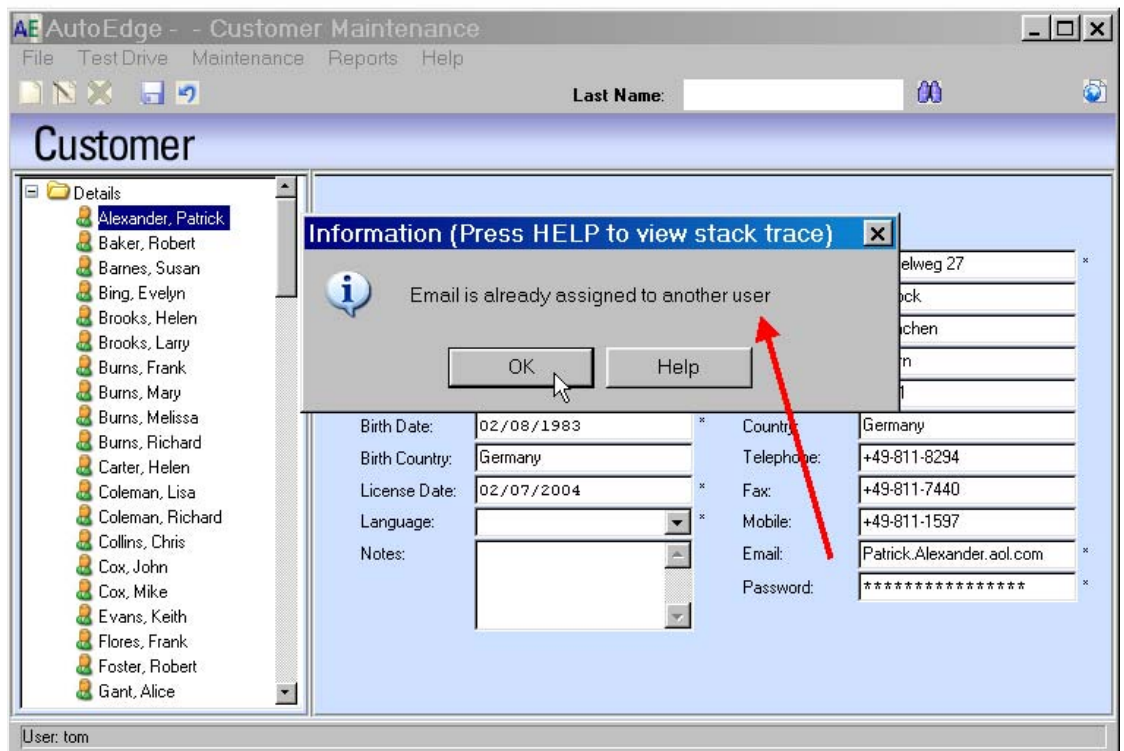
continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Debugging Code, continued

8. Click the **Resume** button.
9. The application will show another error message indicating that the email is assigned to another user. Obviously you would not want two error messages in an application, so one of the messages should be deleted in the code and the other message should be generic enough to handle any errors.

Click the **OK** button.

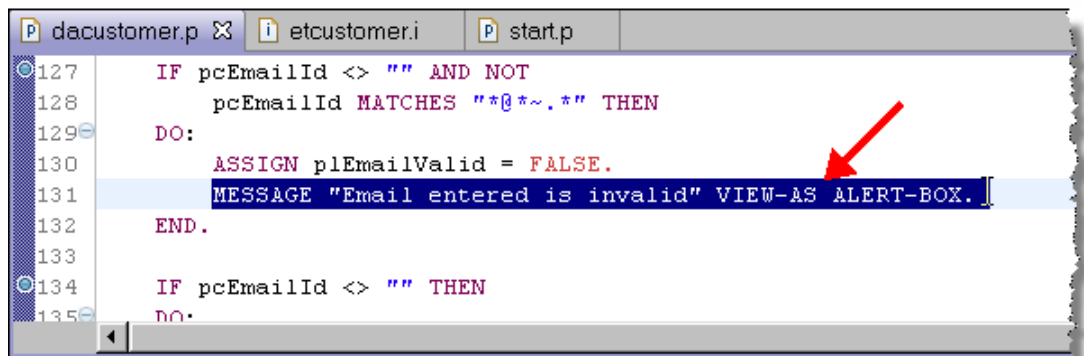


continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Debugging Code, continued

10. Switch back to the Debugger Perspective. Find the line of code with the MESSAGE statement that you previously added to the validateEmail procedure.



```
dacustomer.p  etcustomer.i  start.p
127  IF pcEmailId <> "" AND NOT
128      pcEmailId MATCHES "**@*~.*" THEN
129  DO:
130      ASSIGN plEmailValid = FALSE.
131  MESSAGE "Email entered is invalid" VIEW-AS ALERT-BOX.
132  END.
133
134  IF pcEmailId <> "" THEN
135  DO:
```

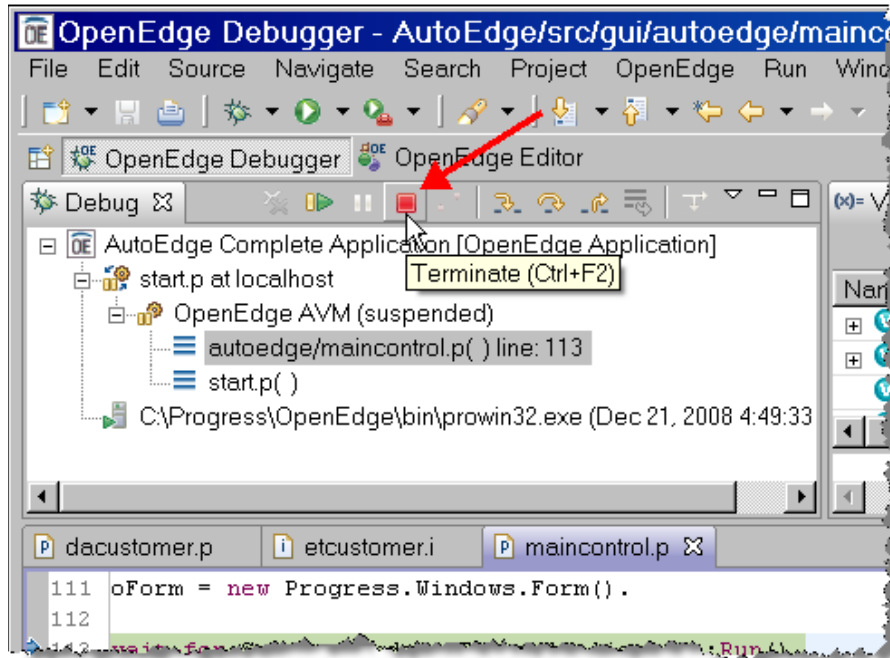
11. Delete the line of code and save the file. Although the coding changes will not be reflected until the next time the code is run or debugged, this is a handy feature for fixing code while debugging.


continued on next page

Lab 8 – Using the OpenEdge Debugger, continued

Debugging Code, continued

12. Stop the application by selecting the **Terminate** button.



 **Note:** From here a developer could continue debugging to make the second error message more generic.

13. Close the Debugger perspective. Close all source code files in editors.
-

Lab 9 – Using Tools for Business Logic

Overview

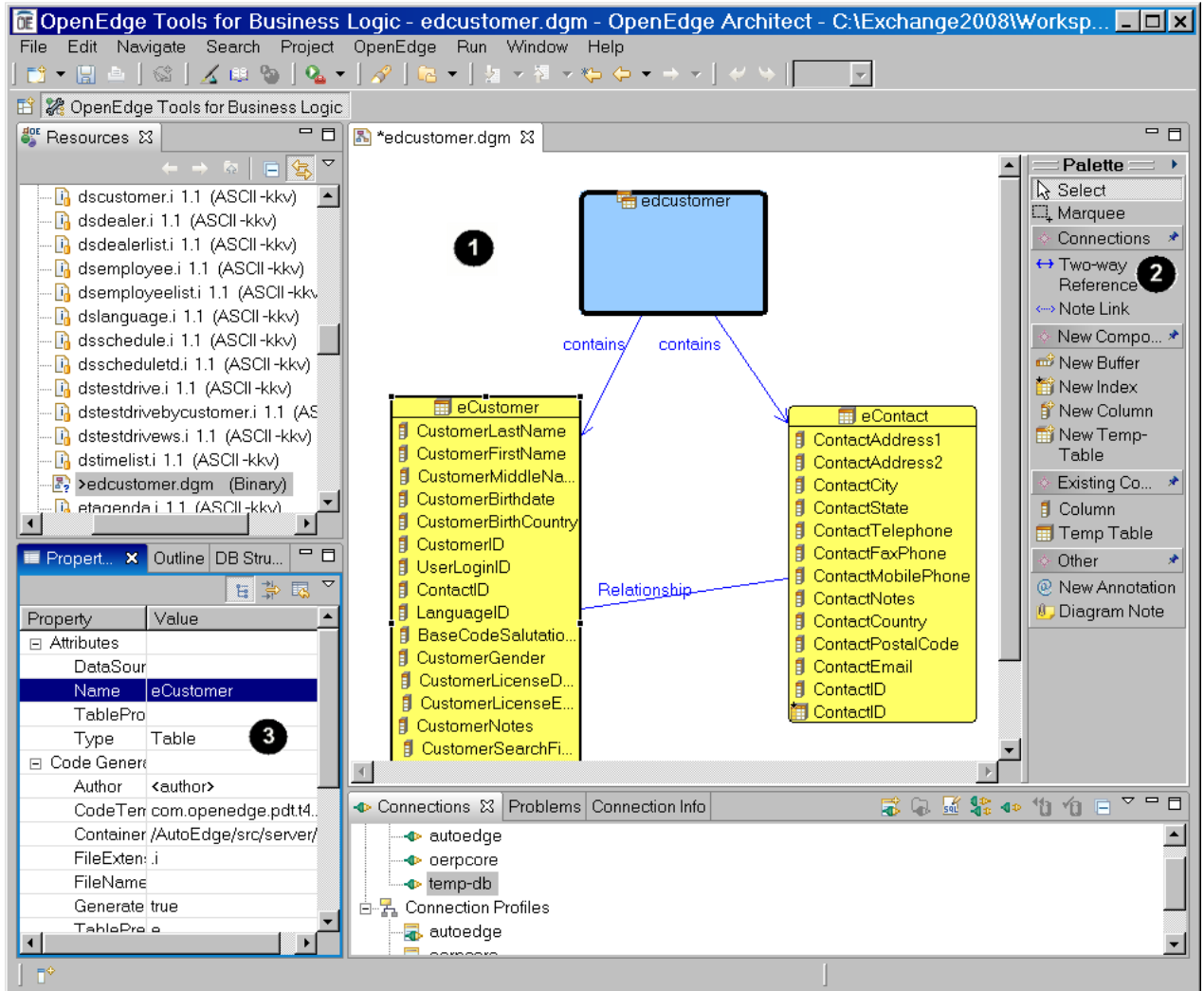
The Tools for Business Logic provide an intuitive drag and drop interface for modeling in-memory business structure. Models can be created for Temp-tables and ProDataset objects. These models can then be stored and reused for either designing or development purposes. You can generate code for these objects directly from the models. Models can also be generated from existing code.

This lab demonstrates some of the available features in Architect's Tools for Business Logic.

continued on next page

Lab 9 – Using Tools for Business Logic, continued

Tools for Business Logic Perspective



Notes:

- ❶ Component Designer Canvas Area
- ❷ Designer Palette
- ❸ Properties, Outline, and DB Structure views

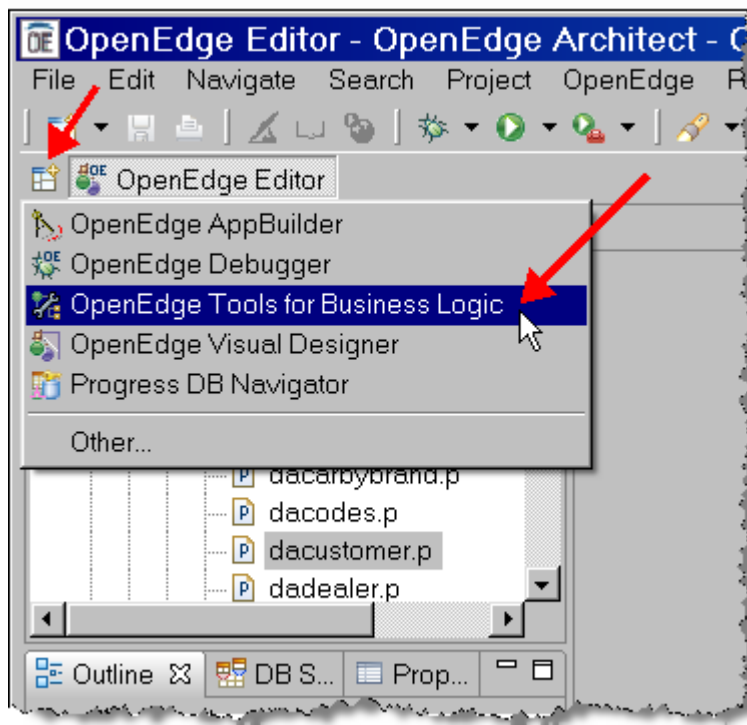
continued on next page

Lab 9 – Using Tools for Business Logic, continued

Creating a ProDataSet Component

The following steps start the process for creating a ProDataSet Component that can hold Customer, Test Drive, and Car information from the AutoEdge Database.

1. Open the Tools for Business Logic Perspective by selecting the **OpenEdge Tools for Business Logic** option from the Open Perspective button.

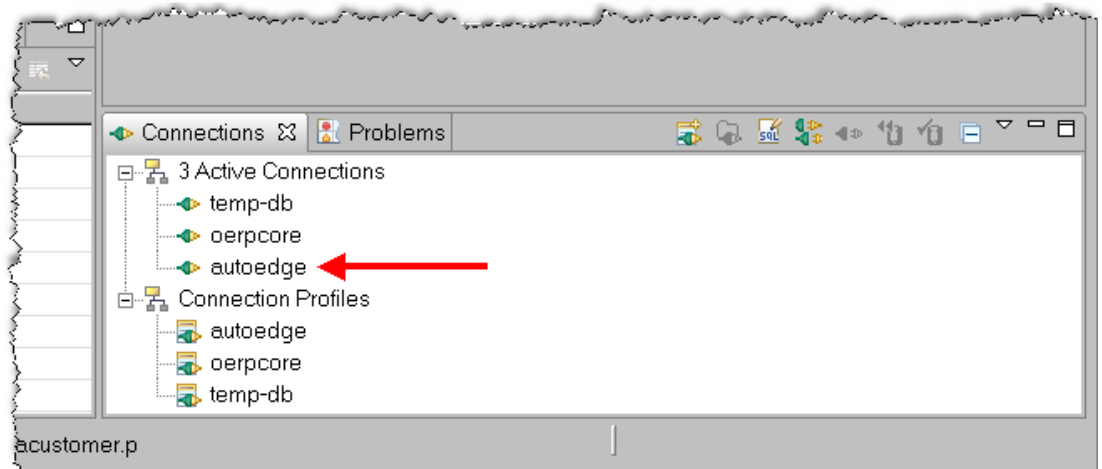


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Creating a ProDataSet Component, continued

2. Since the Tools for Business Logic use a SQL connection to retrieve schema information from a database, make sure that the **autoedge** connection is active in the Connections View. If not, double-click on the **autoedge** entry under the Connection Profiles node to connect.

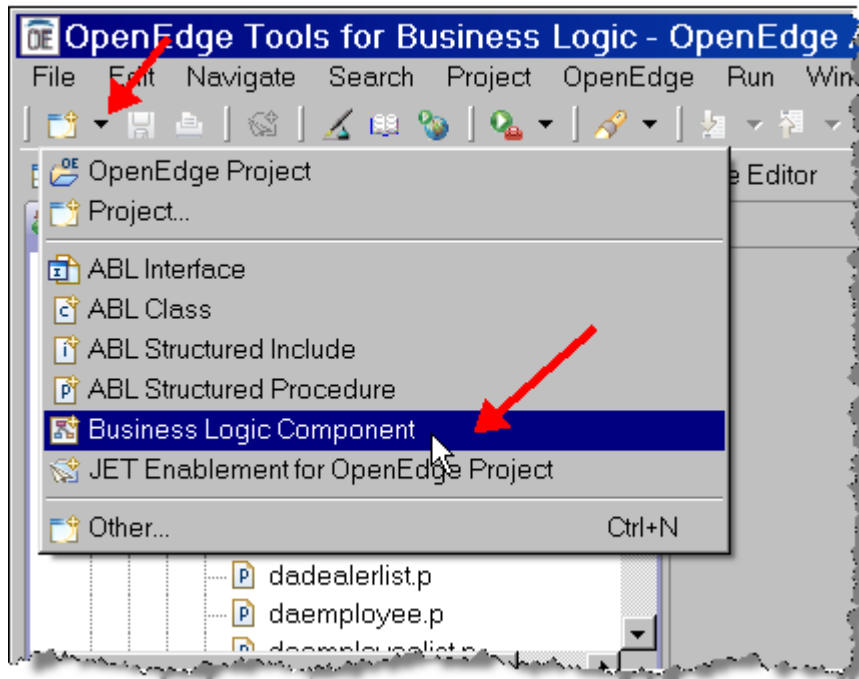


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Creating a ProDataSet Component, continued

3. Create a new Business Logic Component by selecting the down arrow next to the New button and selecting the **Business Logic Component** option.

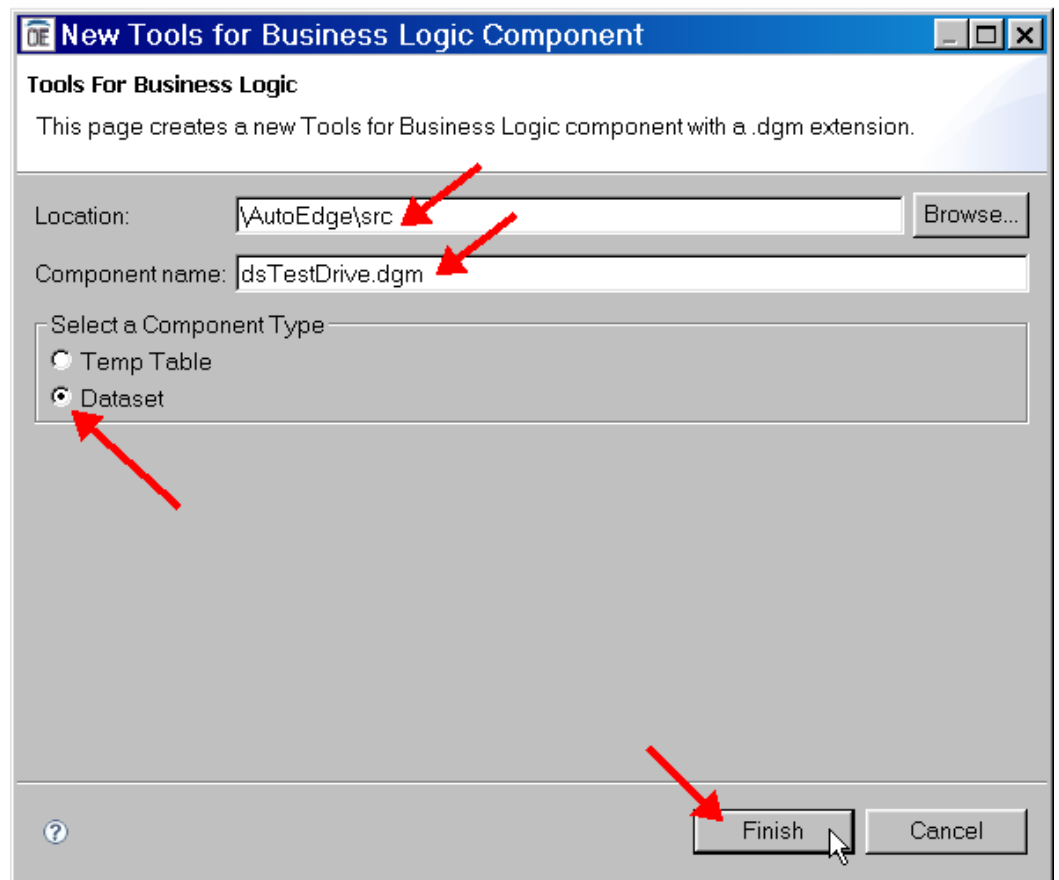


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Creating a ProDataSet Component, continued

4. In the New Component dialog-box, set the following values and then select the **Finish** button:
 - Location: `\AutoEdge\src`
 - Component Name: `dsTestDrive.dgm`
 - Component Type: **Dataset**

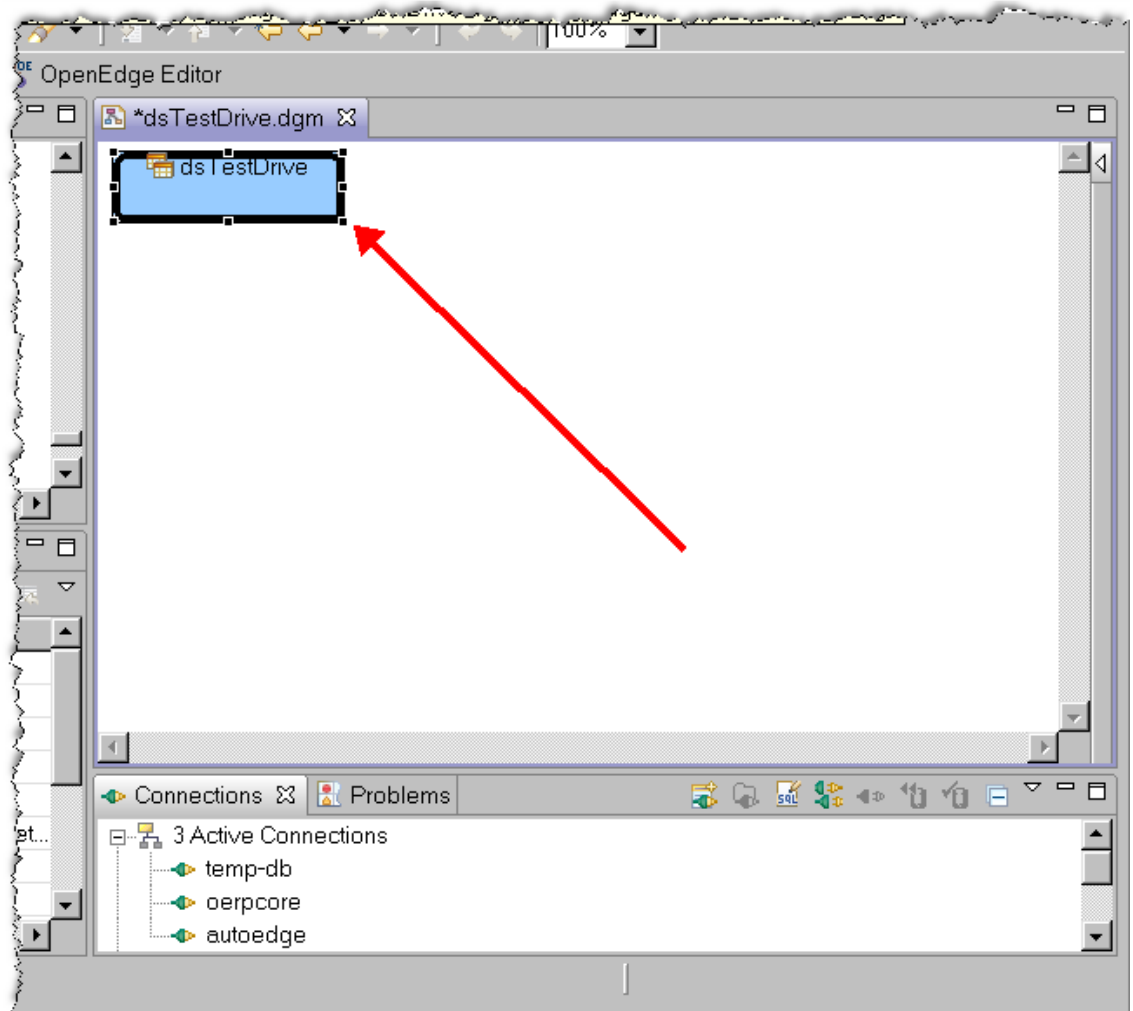


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Creating a ProDataSet Component, continued

5. The diagram file is created and the dsTestDrive ProDataSet component will be shown in the Component Designer editor. Click and drag the component to the top left hand corner of the Component Designer's Canvas Area and resize it accordingly.



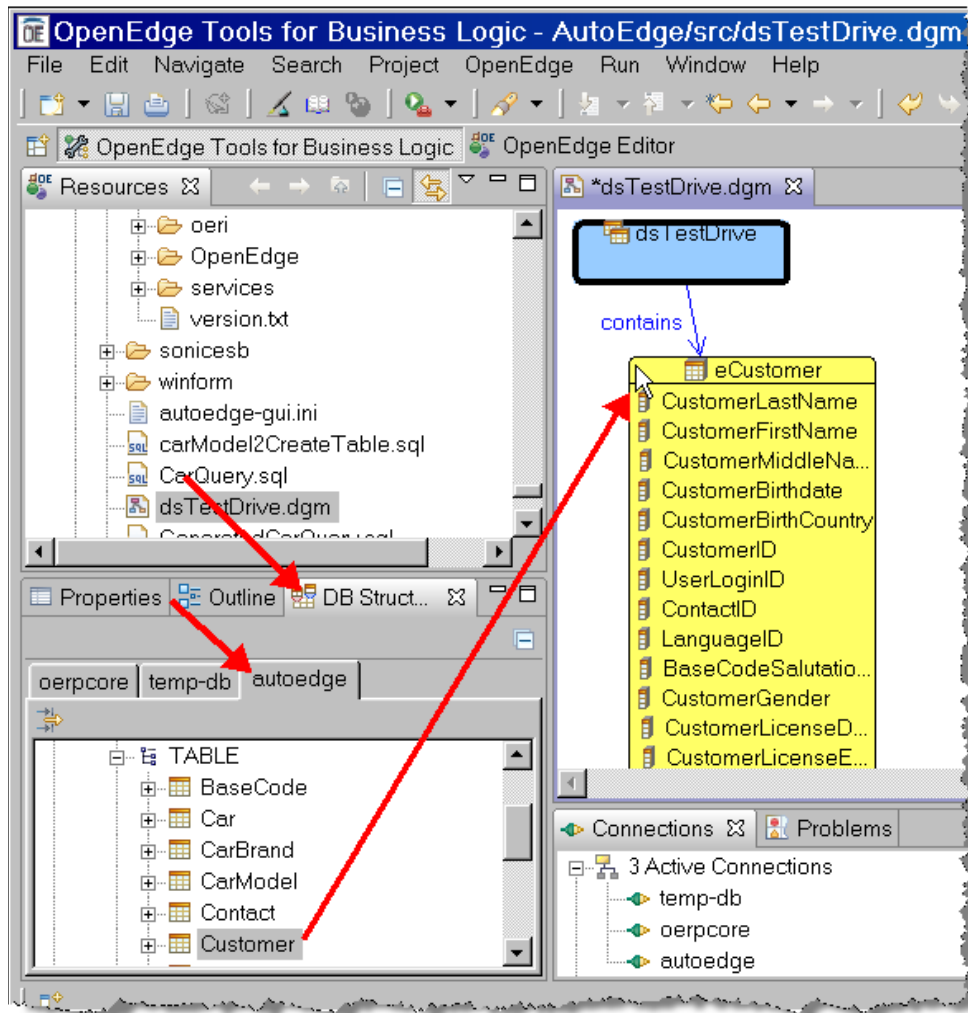
continued on next page

Lab 9 – Using Tools for Business Logic, continued

Creating Temp-Tables Components from Database Schema Information

ProDataset Components are built using one or more temp-table Components. There are many ways in which to create temp-table components, but one quick way is by copying the schema structure of a table from a database and turning it directly into a temp-table component. The following exercise uses the DB Structure view to copy schema structures from the AutoEdge database to create the temp-table components for the TestDrive ProDataset Component.

1. In the DB Structure View, select the **autoedge** tab. Expand the **Database→AutoEdge→Table** nodes. Click and drag the **Customer** entry over to the canvas area and release the button. A new customer temp-table component will be created.

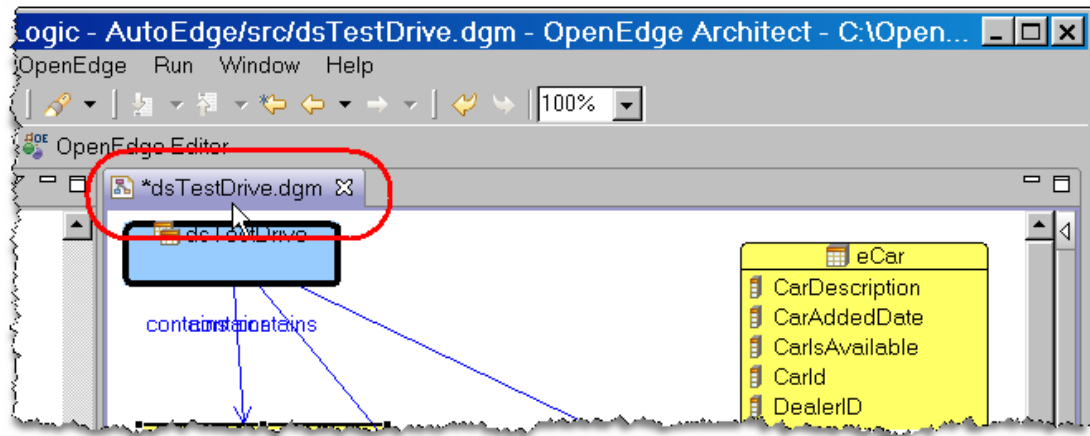


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Creating Temp-Tables Components from Database Schema Information, continued

2. Repeat Step 1 for the **TestDrive** and **Car** tables:
3. Maximize the Component Designer editor by double-clicking on the diagram's tab label.

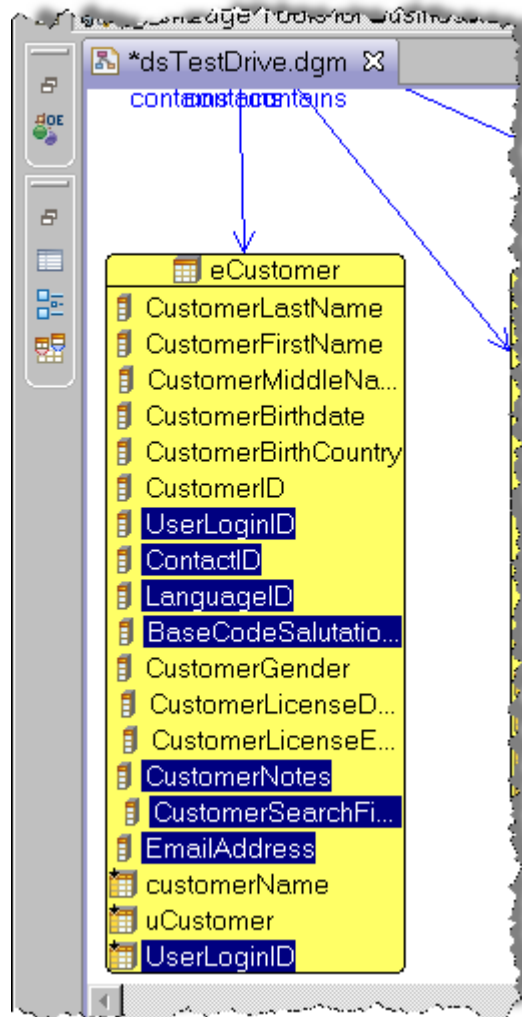


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Creating Temp-Tables Components from Database Schema Information, continued

4. In the Customer component, select the following elements by using the CTRL key and clicking on their entry:
 - UserLoginID
 - ContactID
 - LanguageID
 - BaseCodeSalutation
 - CustomerNotes
 - CustomerSearchField
 - EmailAddress
 - userLoginID (index)

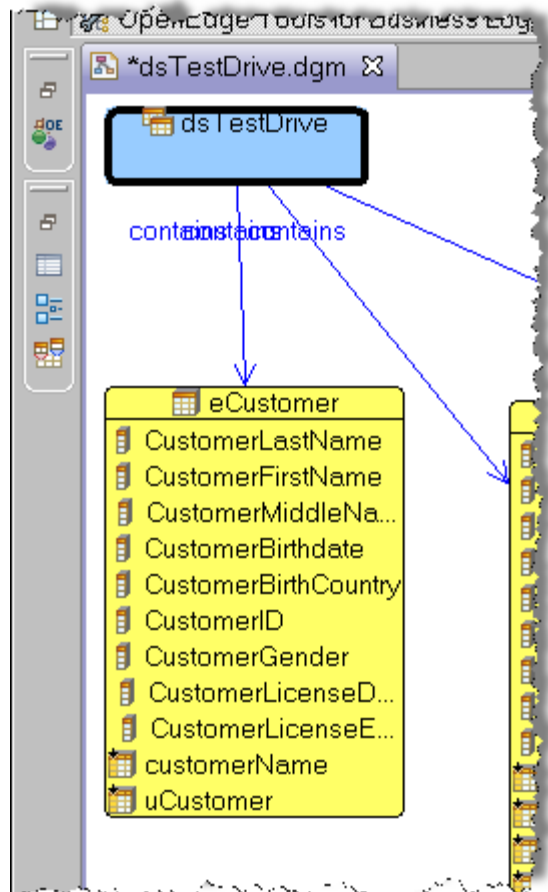


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Creating Temp-Tables Components from Database Schema Information, continued

5. Press the **Delete** key to delete the elements.



continued on next page

Lab 9 – Using Tools for Business Logic, continued

Creating Temp-Tables Components from Database Schema Information, continued

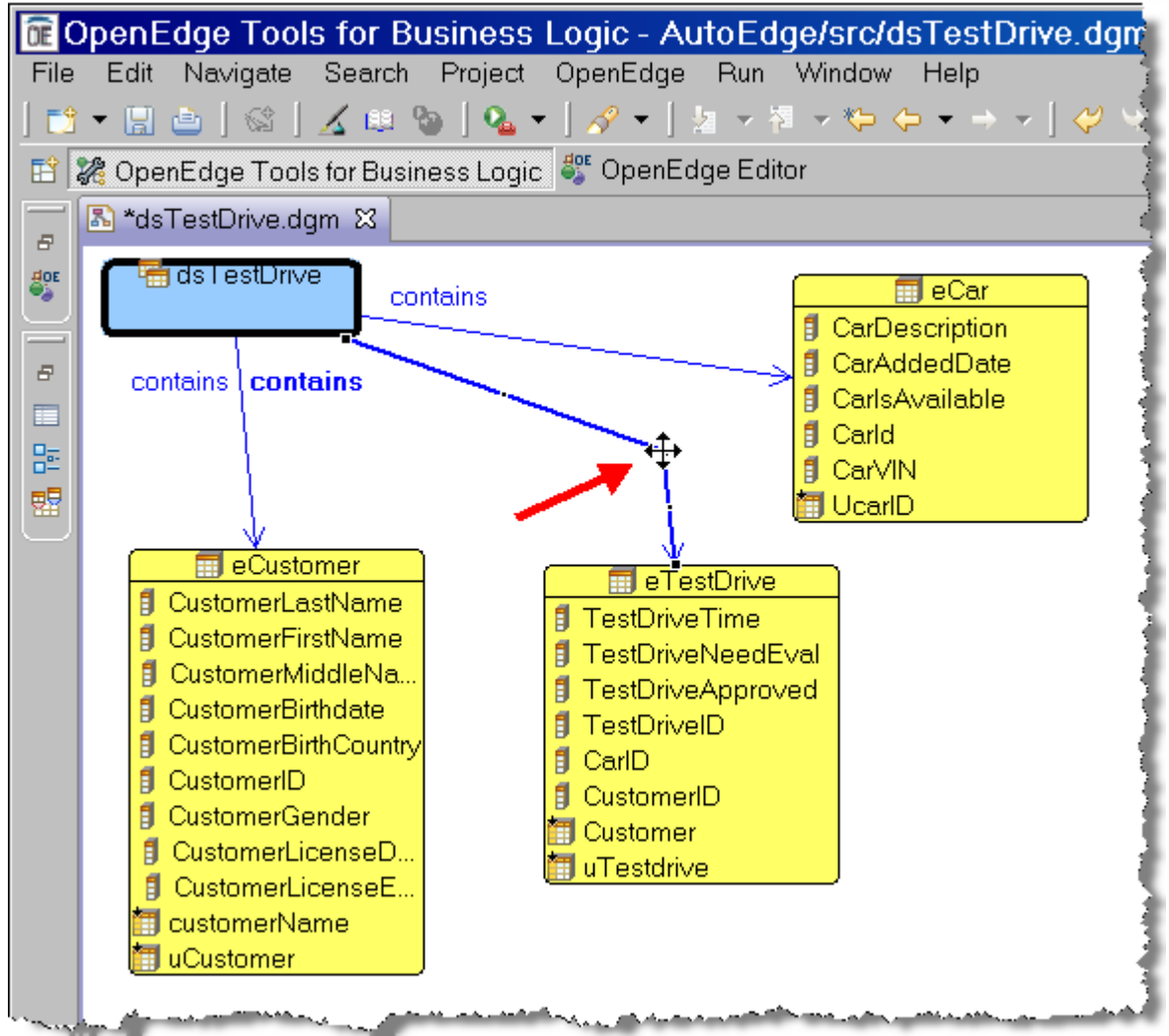
6. Repeat steps 4 and 5 for the TestDrive and Car components.
 - TestDrive
 - ◆ TestDriveNotes
 - ◆ DealerID
 - ◆ EmployeeID
 - ◆ DealerTime (index)
 - ◆ Employee (index)
 - ◆ TestDriveTime (index)
 - Car
 - ◆ DealerID
 - ◆ CarBrandID
 - ◆ CarModelID
 - ◆ BaseCodeStyleID
 - ◆ BaseCodeColorID
 - ◆ BaseCodeEngineID
 - ◆ BaseCodeFuelID
 - ◆ Brand (index)
 - ◆ ColorCode (index)
 - ◆ DealerCar (index)
 - ◆ Engine (index)
 - ◆ Fuel (index)
 - ◆ Model (index)
 - ◆ Style (index)

continued on next page

Lab 9 – Using Tools for Business Logic, continued

Creating Temp-Tables Components from Database Schema Information, continued

7. Position the components in the canvas area to look similar to the image below. The lines can be repositioned by clicking on the line and then clicking and dragging on one of the black boxes that are shown.



8. Restore the editor to its original size by double-clicking on its tab.

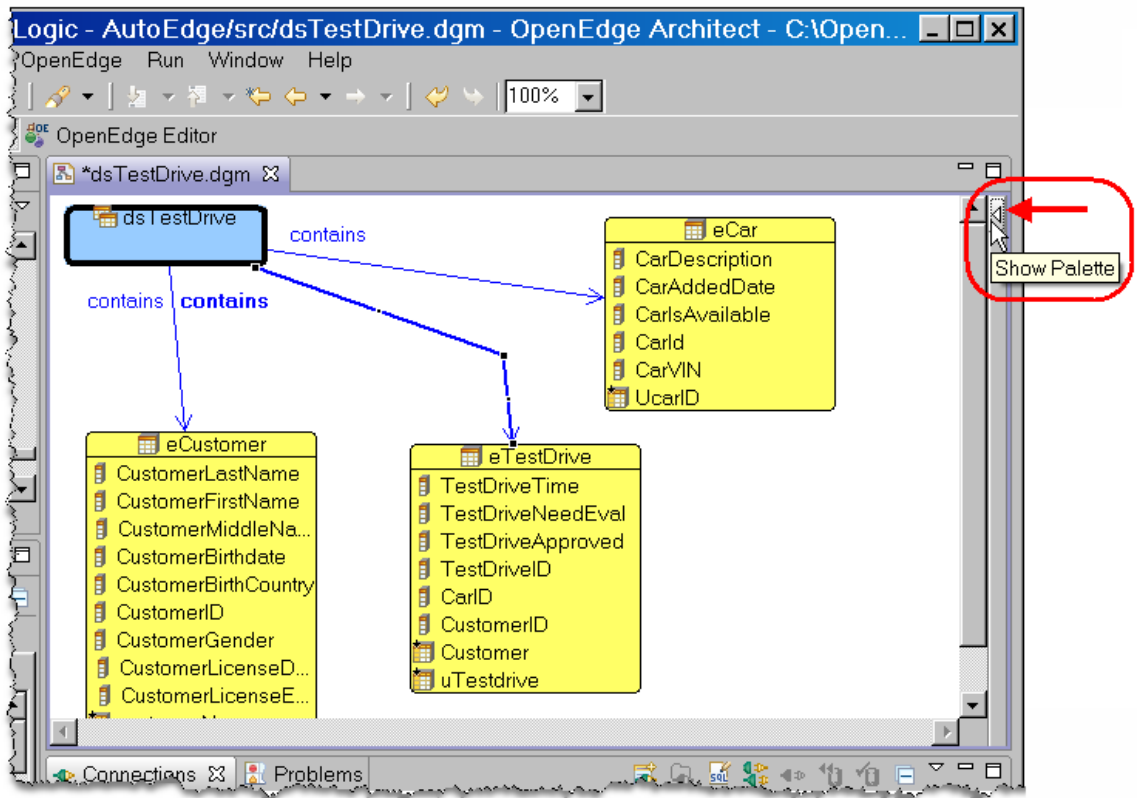
continued on next page

Lab 9 – Using Tools for Business Logic, continued

Adding Relationship Links

Complete the following steps to add Data Relationships between the temp-table components in the Test Drive Component Diagram:

1. Open the Palette by selecting the arrow above the Palette label. Note that you also can hover your mouse over the Palette strip to have the Palette open temporarily while you select a single option.

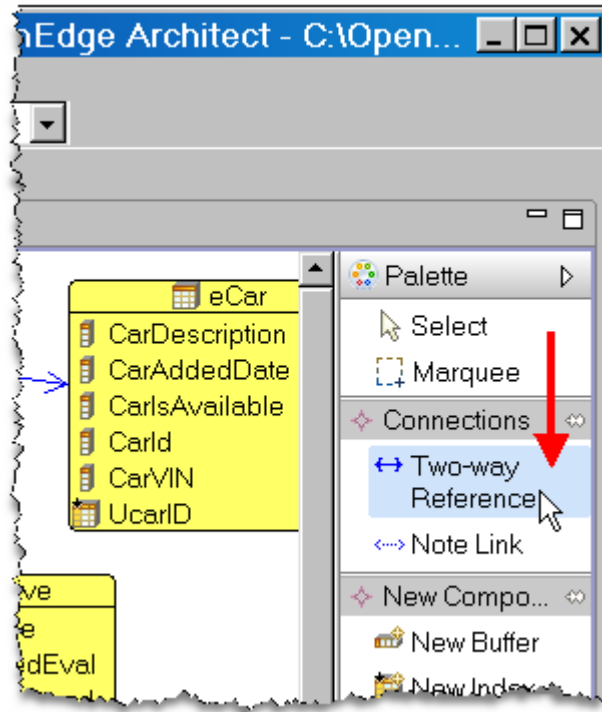


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Adding Relationship Links, continued

2. In the Palette under the Connections label, select the **Two-way Reference** option.

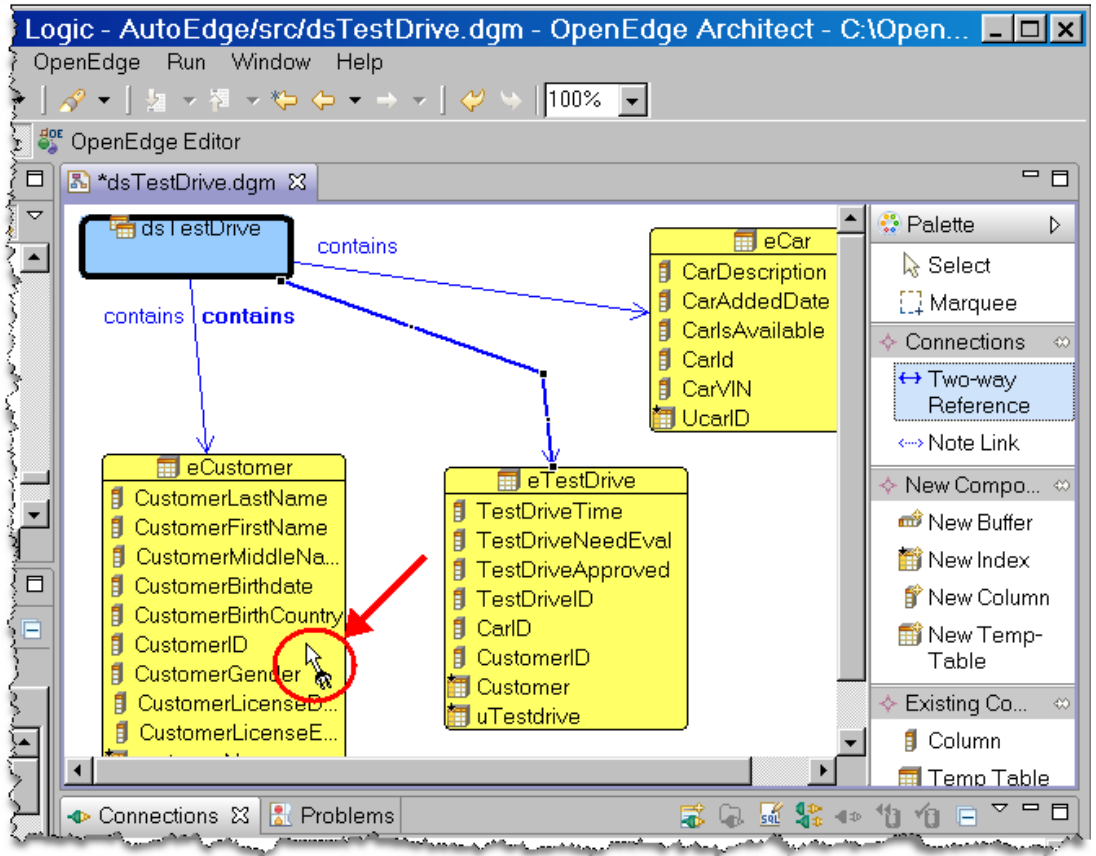


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Adding Relationship Links, continued

3. Place the cursor over the Customer temp-table component.

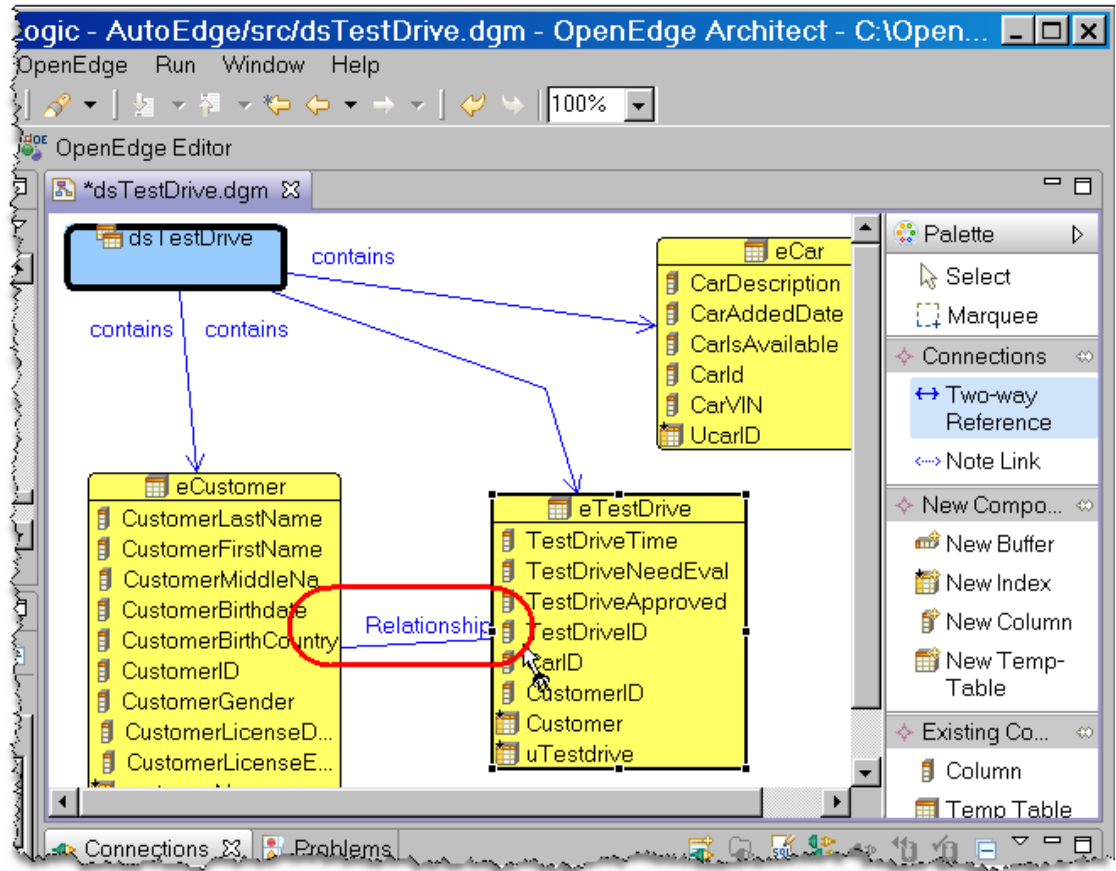


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Adding Relationship Links, continued

- Click and drag the Relationship Line from the Customer component (Source) to the TestDrive component (Target) and click again. The relationship will be created.

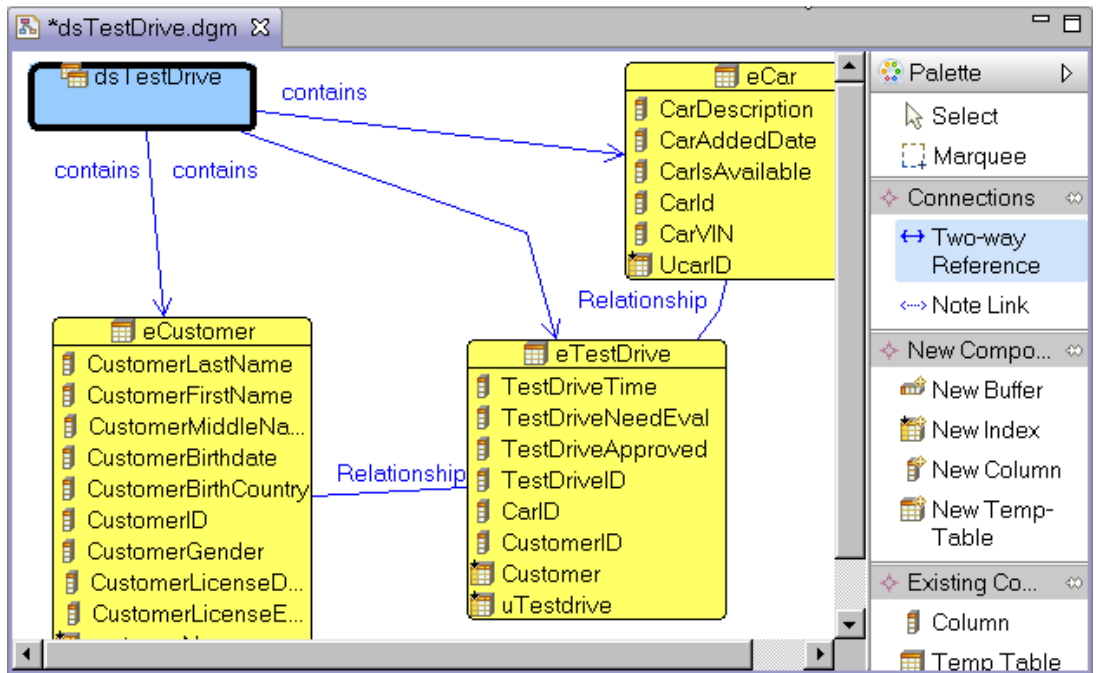


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Adding Relationship Links, continued

- Repeat Step 4 to create a relationship between the TestDrive and Car components. The diagram with the Relationships should look like the following:

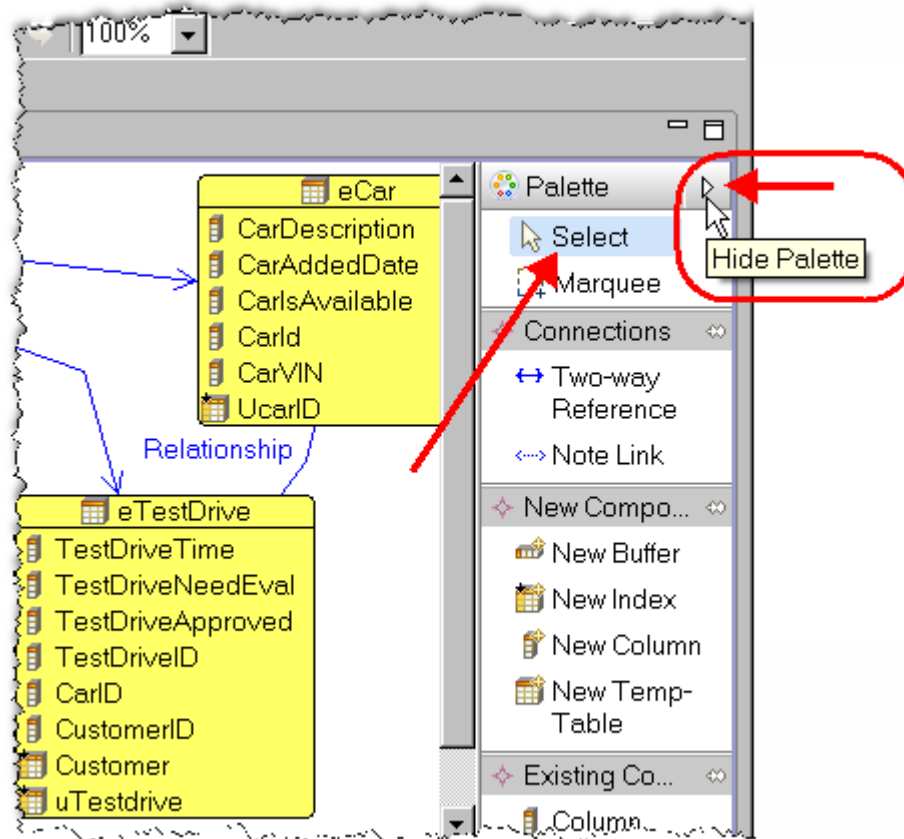


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Adding Relationship Links, continued

- Return to the Palette and select the **Select** option. Hide the Palette by clicking on the arrow to the right of the Palette label.



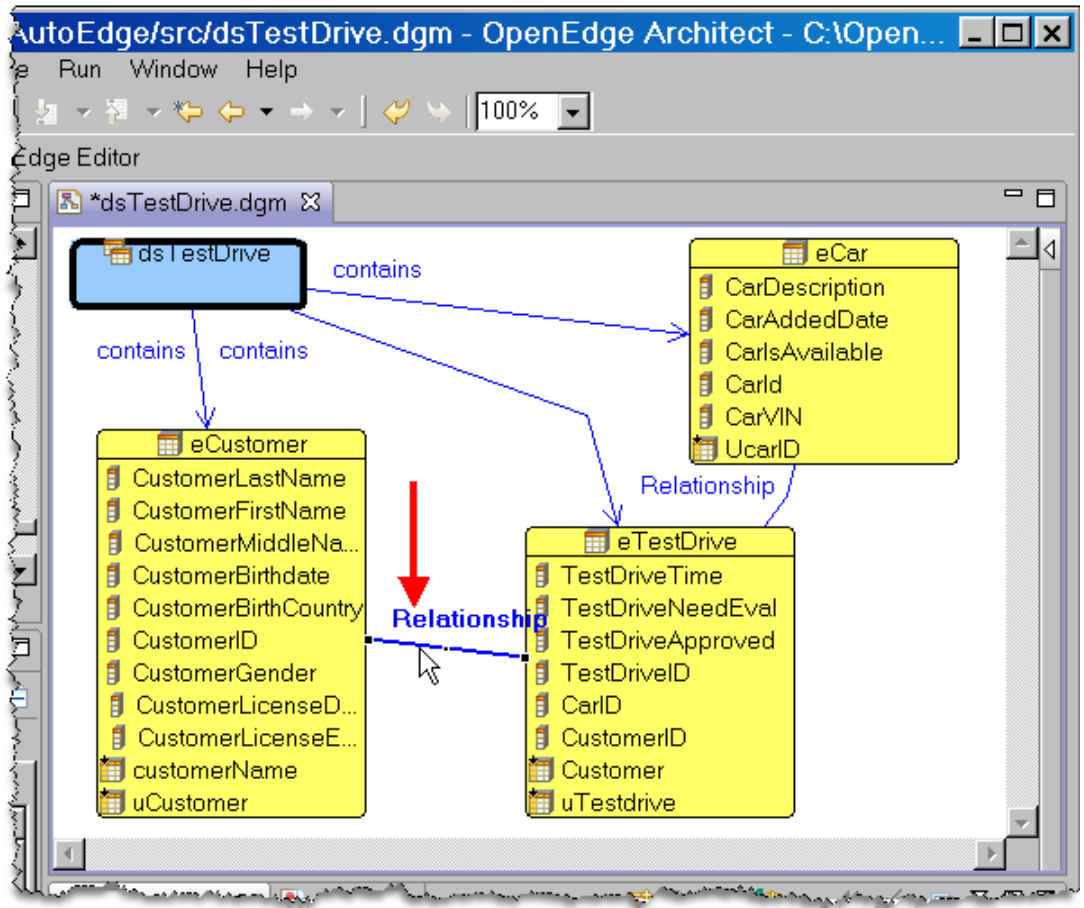
continued on next page

Lab 9 – Using Tools for Business Logic, continued

Setting Properties

This section shows how to set properties for the elements in the diagram.

1. Select the relationship line between the Customer and TestDrive components.

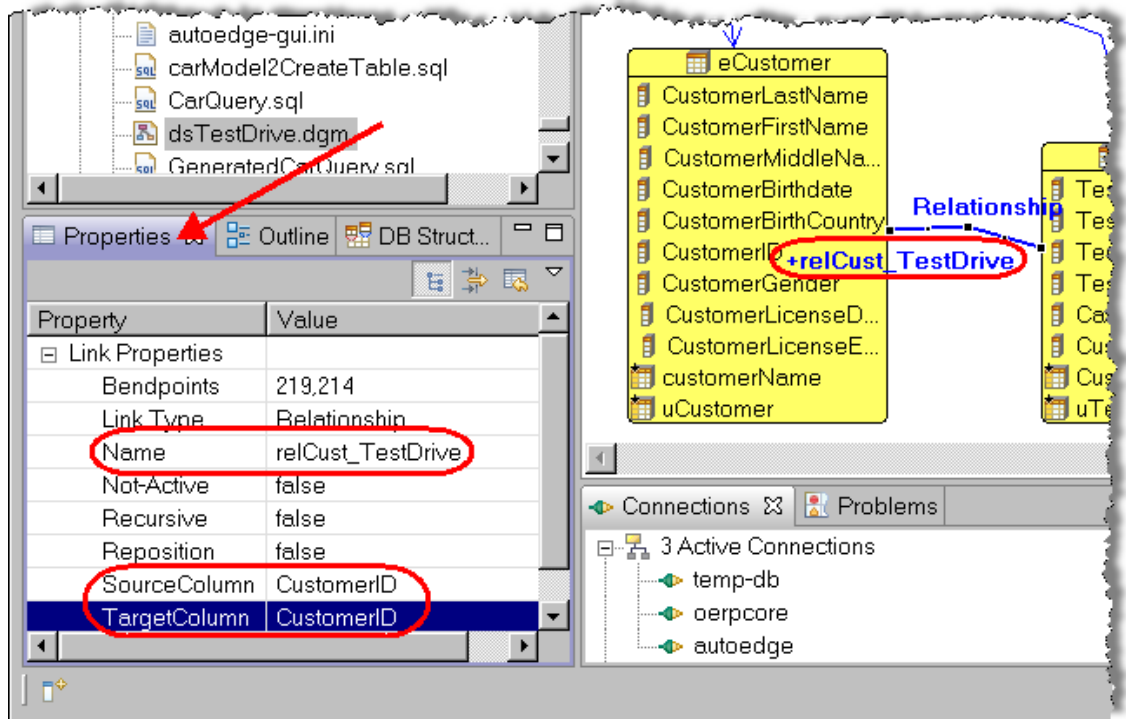


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Setting Properties, continued

2. Select the Properties view. Change the Name property to **relCust_TestDrive** and press the **Enter** key. The name change will be reflected in the diagram. Change the values for the SourceColumn and TargetColumn properties to **CustomerID**. This sets the primary/foreign key relationship between the two Components

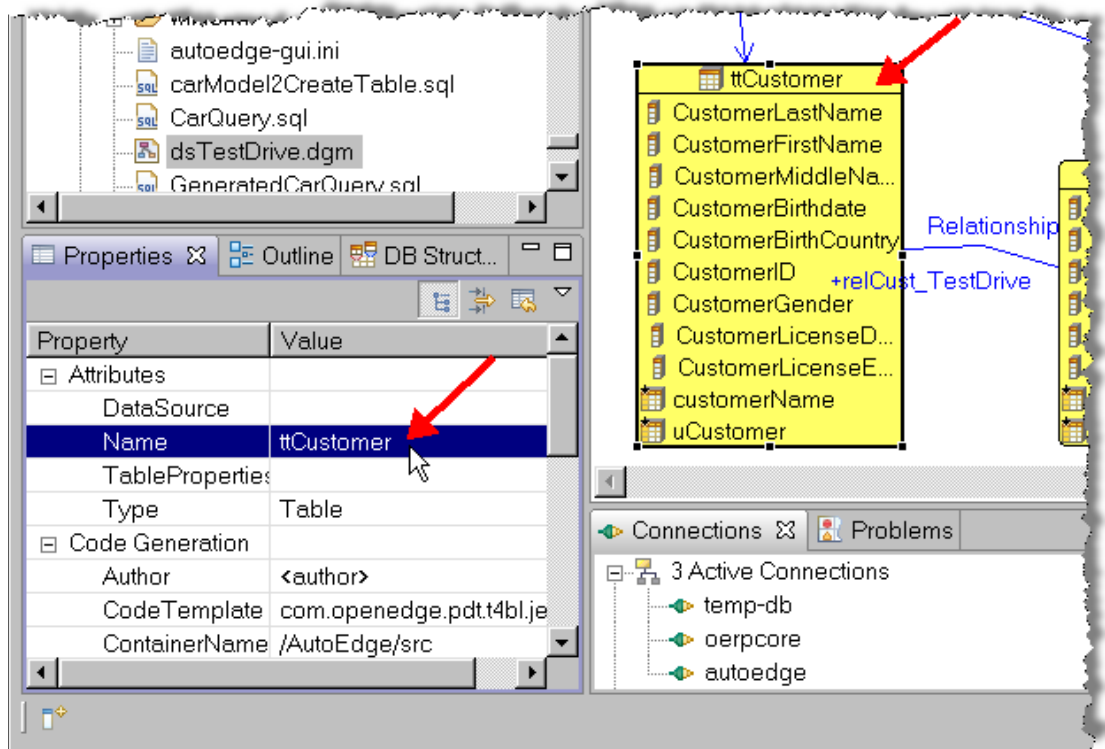


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Setting Properties, continued

- Repeat Step 8 the relationship between the TestDrive and Car components using the following settings:
 - Name: **relTestDrive_Car**
 - SourceColumn: **CarID**
 - TargetColumn: **CarID**
- Select the Customer component. In the Properties view, change the Name property to **ttCustomer**. Press the **Enter** key. The component's name will reflect the change.

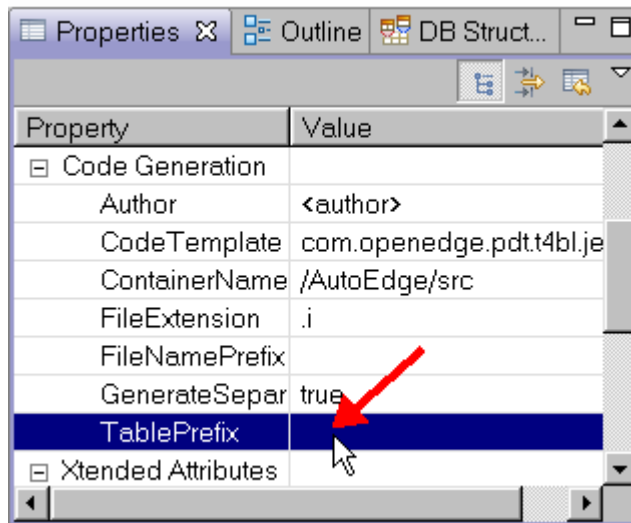


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Setting Properties, continued

5. In the Properties view, scroll down to the TablePrefix property and delete its value so that the value is blank.



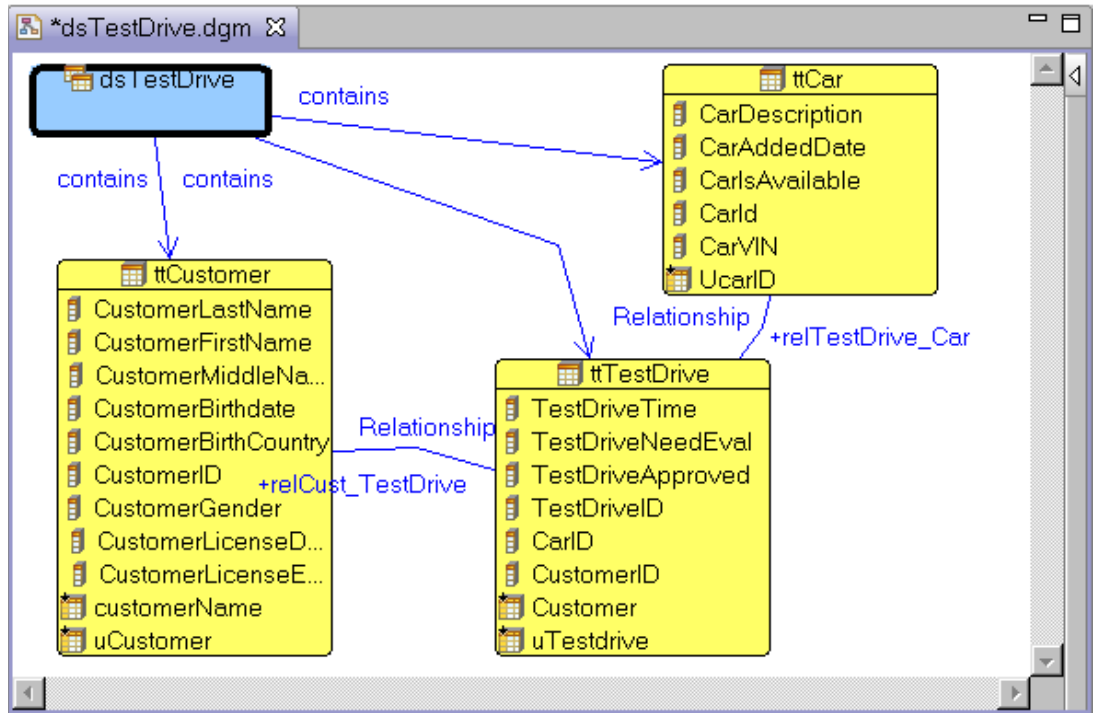
6. Repeat steps 10 and 11 for the TestDrive and Car componets:
 - TestDrive
 - ◆ Name: ttTestDrive
 - ◆ TablePrefix: (blank)
 - Car
 - ◆ Name: ttCar
 - ◆ TablePrefix: (blank)

continued on next page

Lab 9 – Using Tools for Business Logic, continued

Setting Properties, continued

7. The component diagram is now complete and should look like the following:



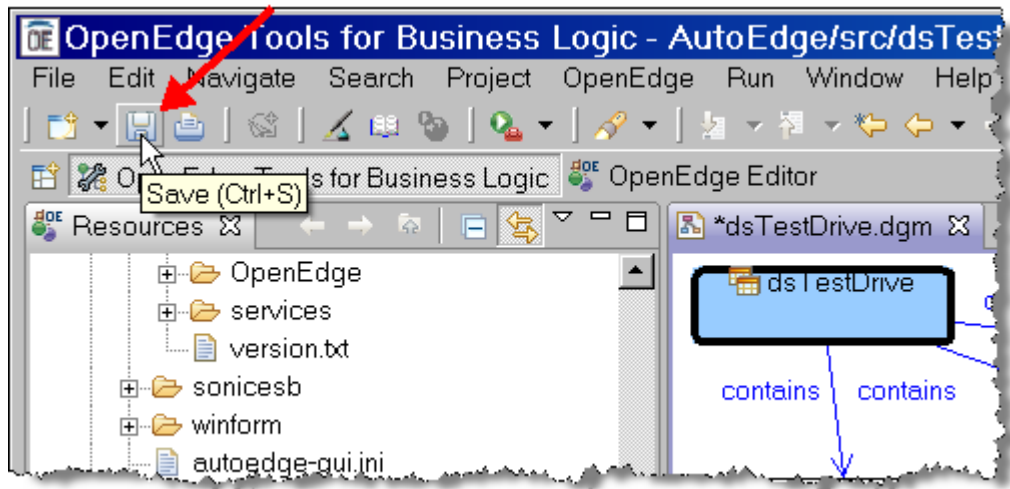
continued on next page

Lab 9 – Using Tools for Business Logic, continued

Forward Engineering Code

Once a Component Diagram is completed, you can use the model to generate ABL code based on the structure and properties of the components in the diagram. Complete the following steps to generate code for the TestDrive ProDataSet, including code for all the related temp-tables:

1. Save the diagram by selecting the **Save** button in Architect’s toolbar.

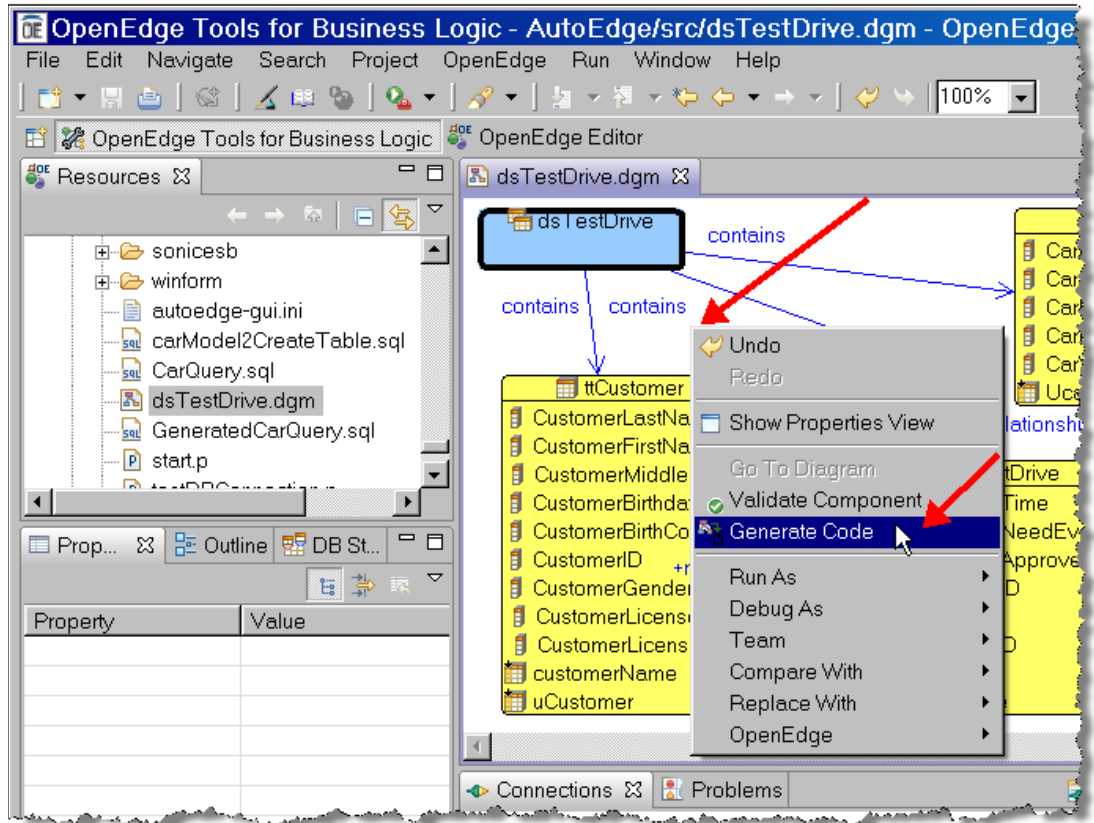


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Forward Engineering Code, continued

2. Right-click in the canvas area and select the **Generate Code** option.

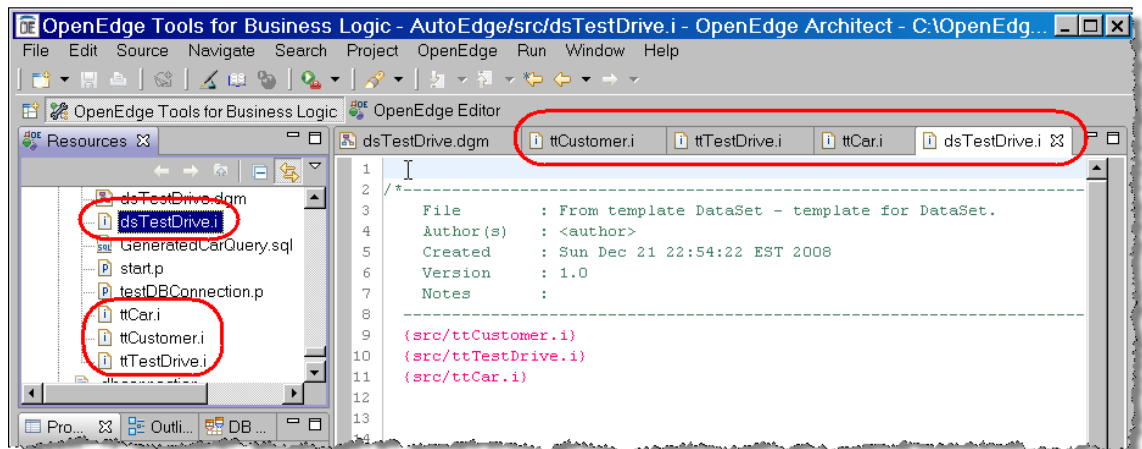


continued on next page

Lab 9 – Using Tools for Business Logic, continued

Forward Engineering Code, continued

3. Include files will be created to hold the ABL code for the components. Review and close the files, including the Component Diagram file.



4. Close the Tools for Business Logic perspective.
-

Lab 10 – Using the Visual Designer

Overview

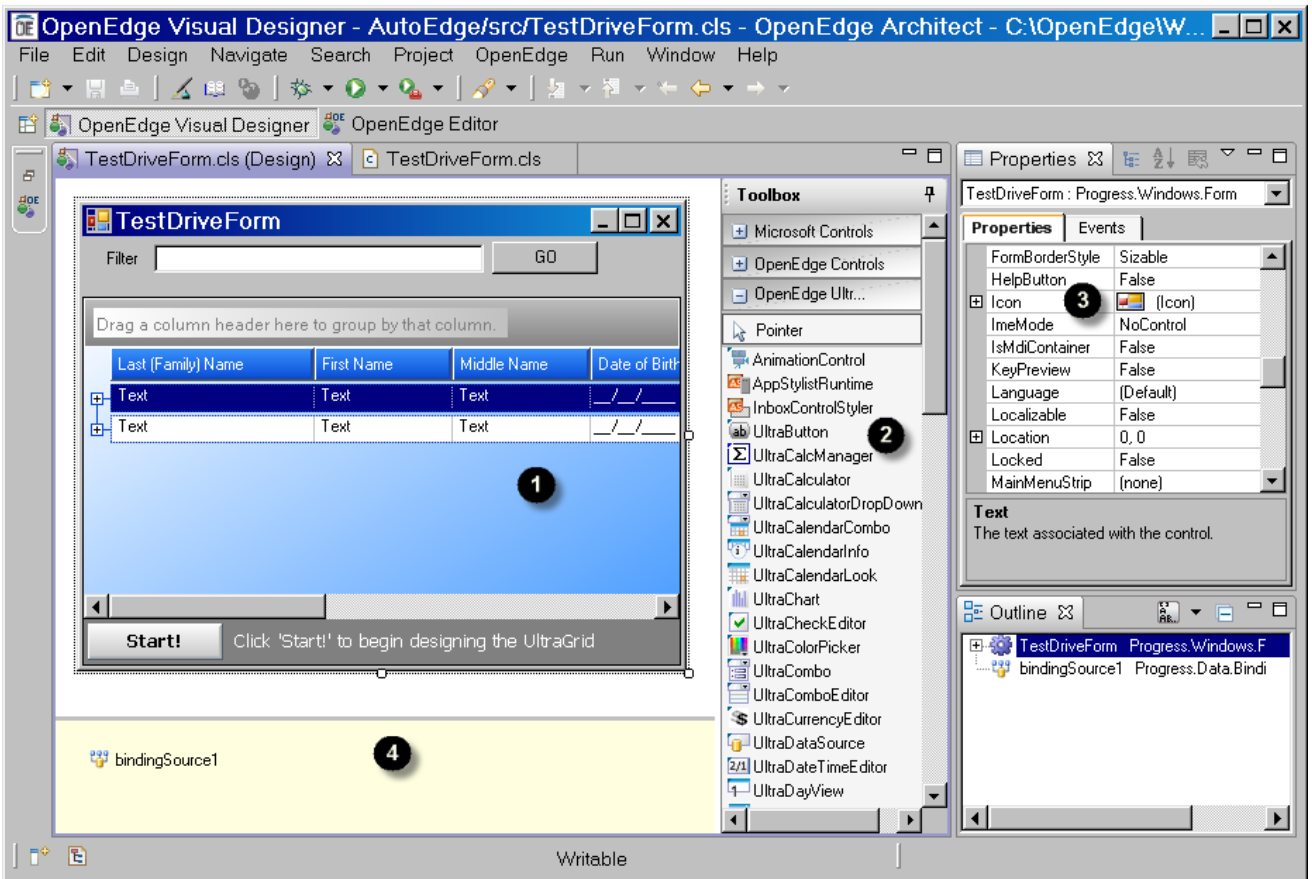
Architect's Visual Designer allows developers to build ABL GUI applications using .NET controls.

This lab shows some of the features of the Visual Designer and builds a sample OpenEdge GUI for .NET client application.

continued on next page

Lab 10 – Using the Visual Designer, continued

Visual Designer Perspective



Notes:

- 1 Visual Designer Editor's Canvas Area
- 2 Visual Designer Editor's Toolbox
- 3 Properties view for property and event settings
- 4 Non-visual controls area

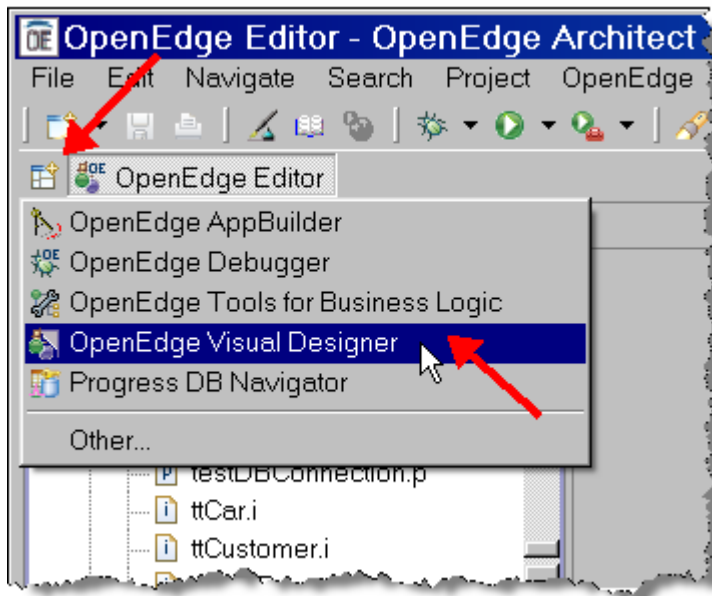
continued on next page

Lab 10 – Using the Visual Designer, continued

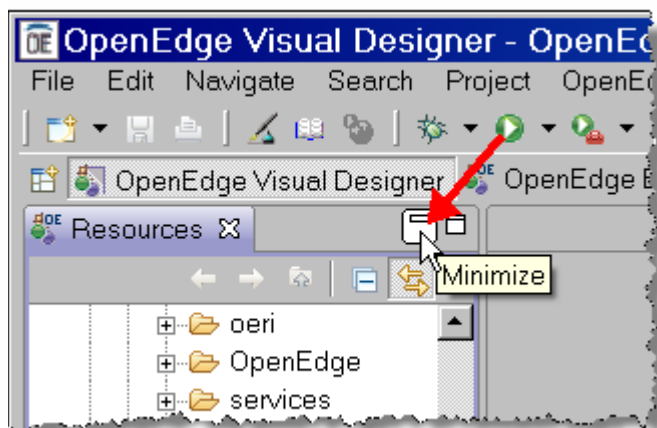
Creating a Form

OpenEdge GUI for .NET client development begins by creating an ABL Form to hold components. This section shows how to create an ABL Form and place components on it.

1. Open the Visual Designer perspective by selecting the **OpenEdge Visual Designer** option from the Open Perspective button.



2. Minimize the Resources view to provide more room for editing the form.

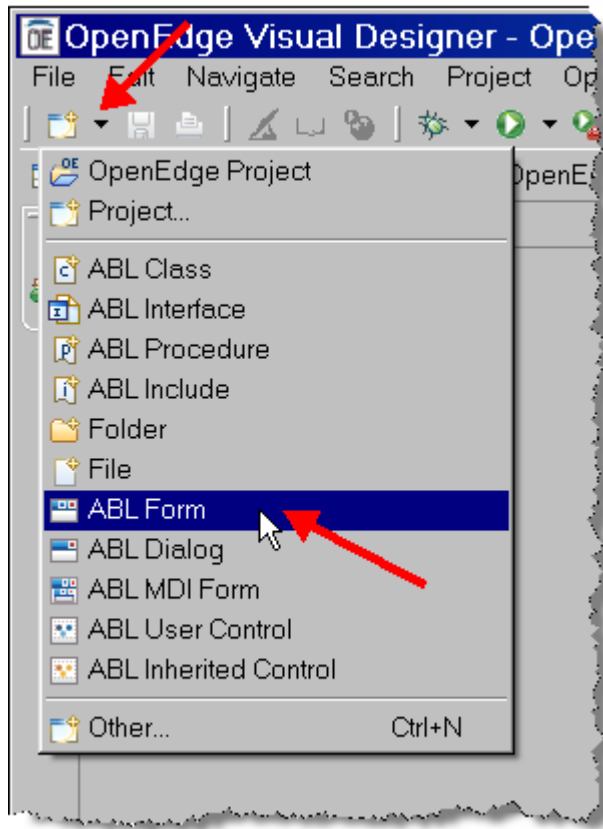


continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Form, continued

3. Select the down-arrow beside the New button and then select the **ABL Form** option.



continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Form, continued

4. In the New ABL Form wizard, enter `\AutoEdge\src` for the Package root and `TestDriveForm` for the Form name. Select the **Finish** button.

The screenshot shows the 'New ABL Form' wizard dialog box. The title bar reads 'New ABL Form'. The main area is titled 'Create a Form Class' and contains the instruction: 'Enter a name for the form. Do not use spaces or special characters.' Below this are several input fields and checkboxes:

- Package root:** \AutoEdge\src (with a red arrow pointing to the text)
- Package:** (empty)
- Form name:** TestDriveForm (with a red arrow pointing to the text)
- Modifiers:** Final Widget pool
- Inherits:** Progress.Windows.Form
- Implements:** (empty)

Below the input fields are two sections for code generation options:

- Select which code elements you would like to generate:**
 - Generate default constructor
 - Generate destructor
 - Generate super class constructors
 - Add routine level error handling
- Select which style of method code you would like to create:**
 - Generate exceptions for required methods
 - Generate default values for required methods

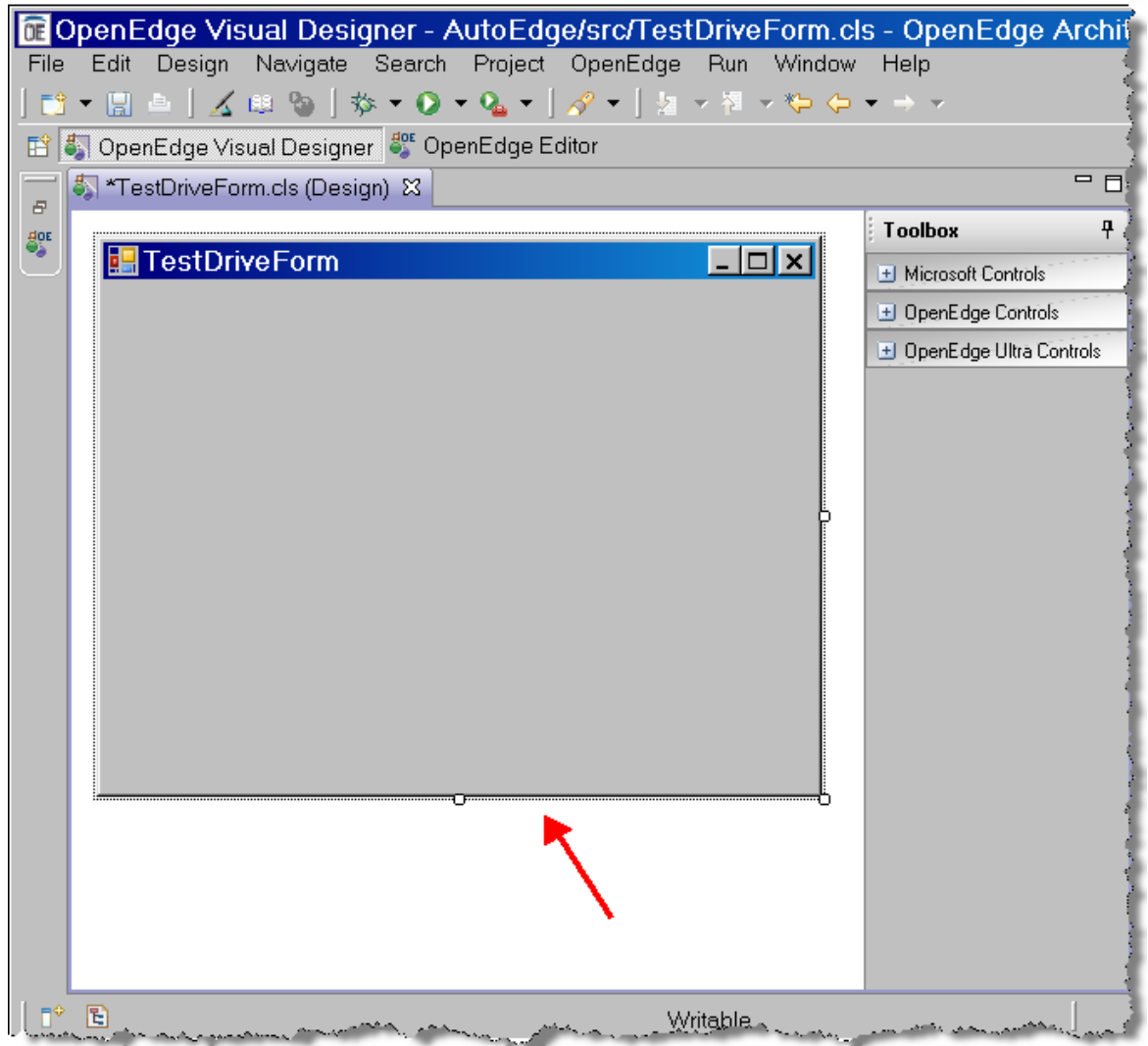
At the bottom, there are two text boxes for 'Description:' and 'Purpose:'. At the very bottom right, there are two buttons: 'Finish' (with a red arrow pointing to it) and 'Cancel'.

continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Form, continued

5. Resize the new form to look similar to the following image:



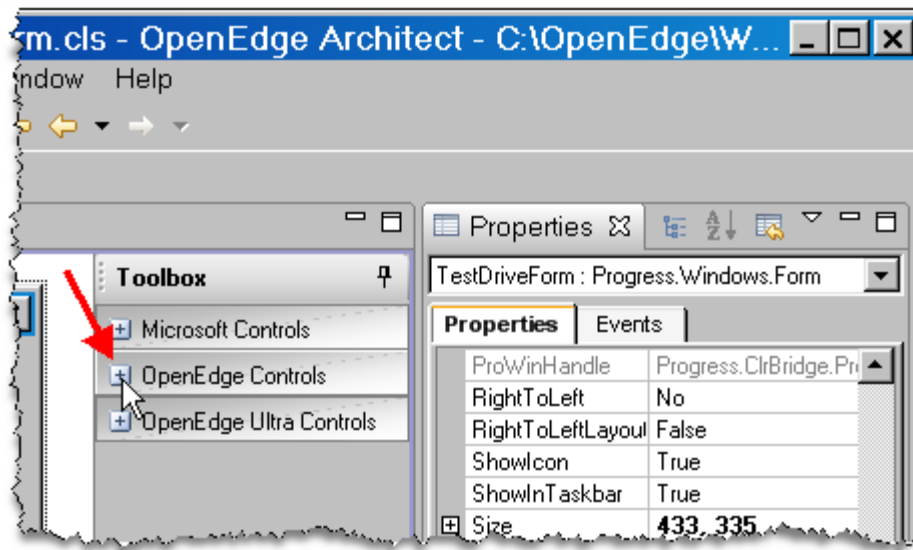
continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Data Source

There are a number of ways of supplying data to a form's controls, but the most common is to use a ProBindingSource control that is tied to a temp-table or ProDataSet. The ProBindingSource is a non-visual control supplied by OpenEdge that acts as a data source to .NET controls. This section shows how to add a ProBindingSource to the form.

1. Expand the **OpenEdge Controls** group in the Toolbox by clicking on the + (Plus) button.

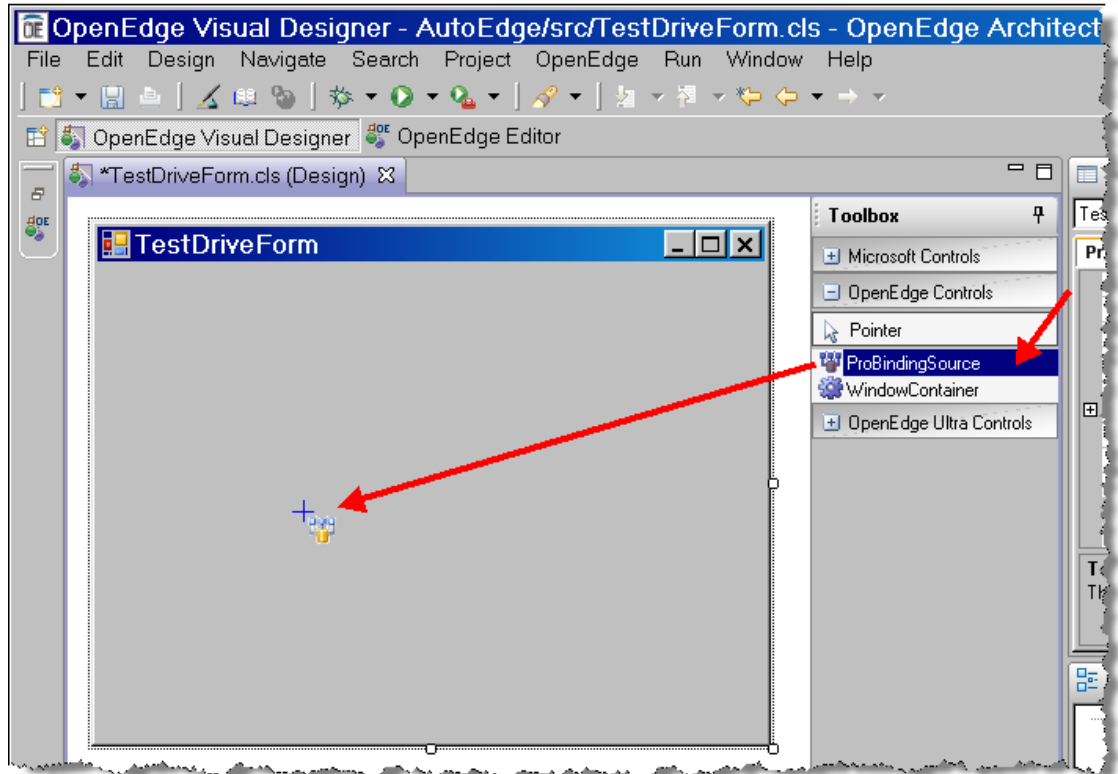


continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Data Source, continued

2. Select the **ProBindingSource** control and position the cursor over the form.

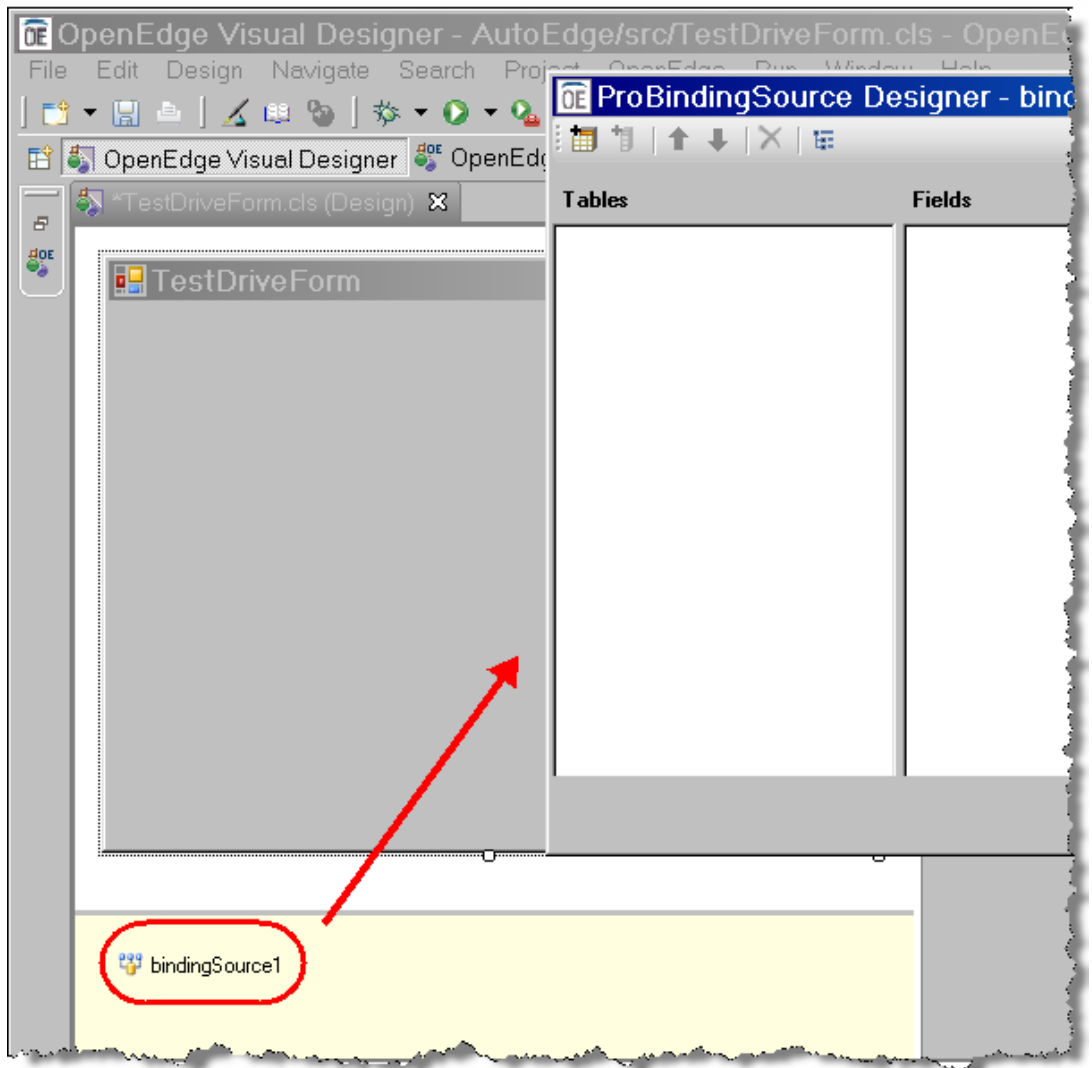



continued on next page


Lab 10 – Using the Visual Designer, continued

Creating a Data Source, continued

3. Click on the form. The control will be placed in the non-visual controls area below the form. A start wizard also will be displayed.



 **Note:** The non-visual controls area helps keep the non-visual controls out of way so that they do not interfere with any visual controls in the form. It also makes finding and selecting non-visual controls easier.

 **Note:** It is very common for controls to have a start wizard that is displayed when they are placed on a form.

continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Data Source, continued

4. The start wizard's toolbar can be used to manually create the schema structure for the data source. However, you can also import the schema using an XSD file that can be directly generated from a ProDataSet's definition.



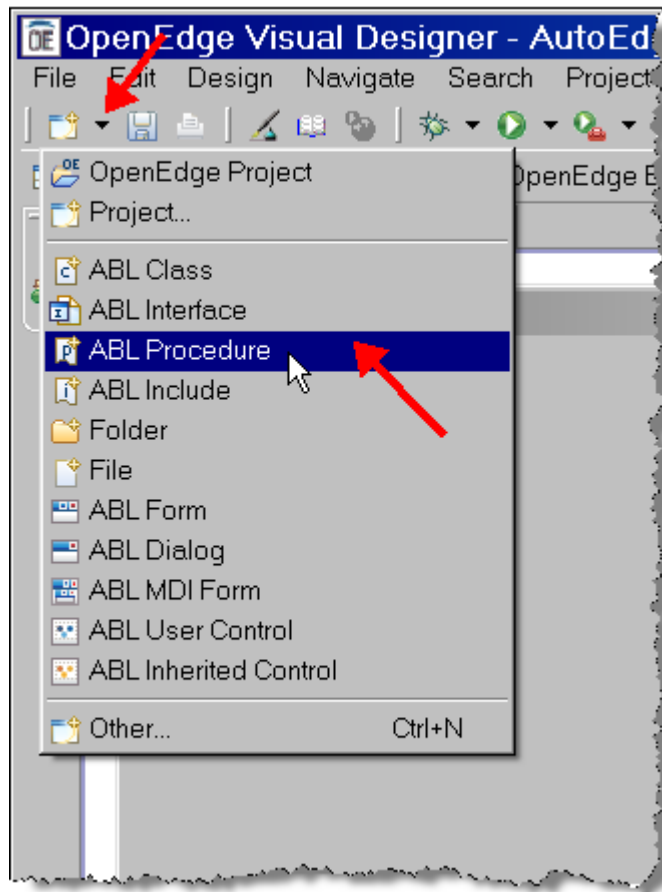
Select the **Cancel** button in the start wizard (you will return to soon).

continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Data Source, continued

5. Select the down-arrow beside the New button and select the **ABL Procedure** option.

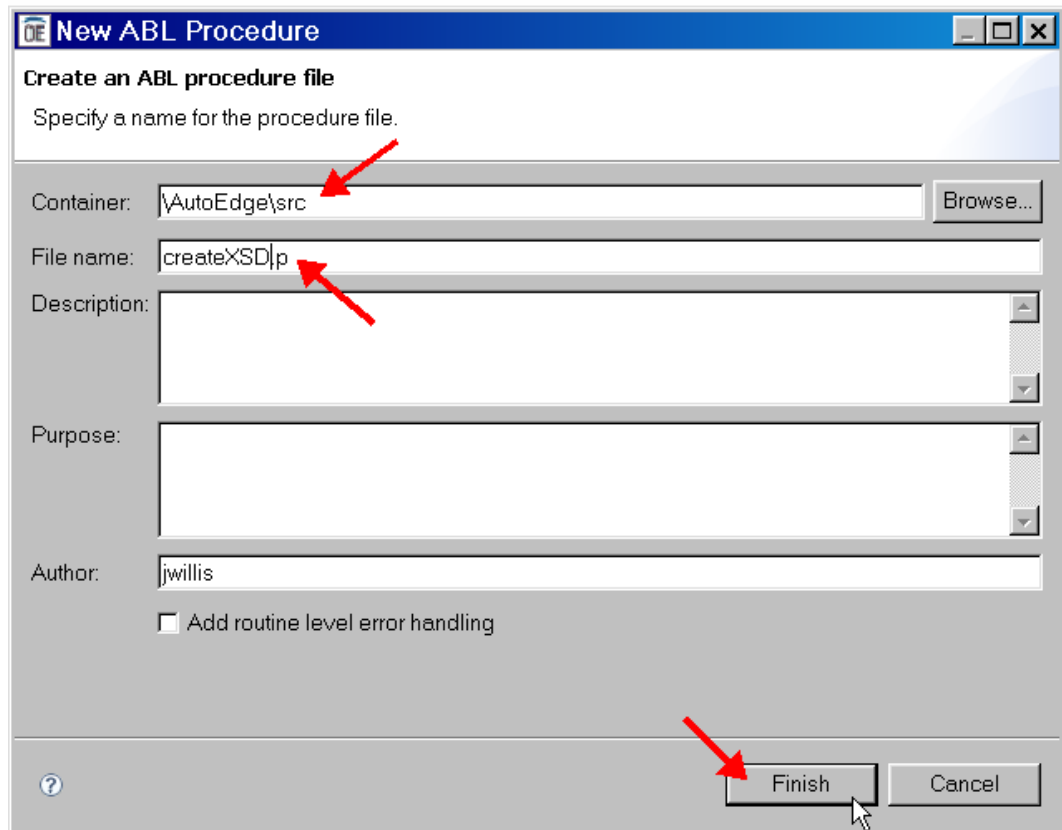


continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Data Source, continued

6. In the New ABL Procedure wizard, set the following and then select **Finish**:
 - Container: `\AutoEdge\src`
 - File name: `createXSD.p`

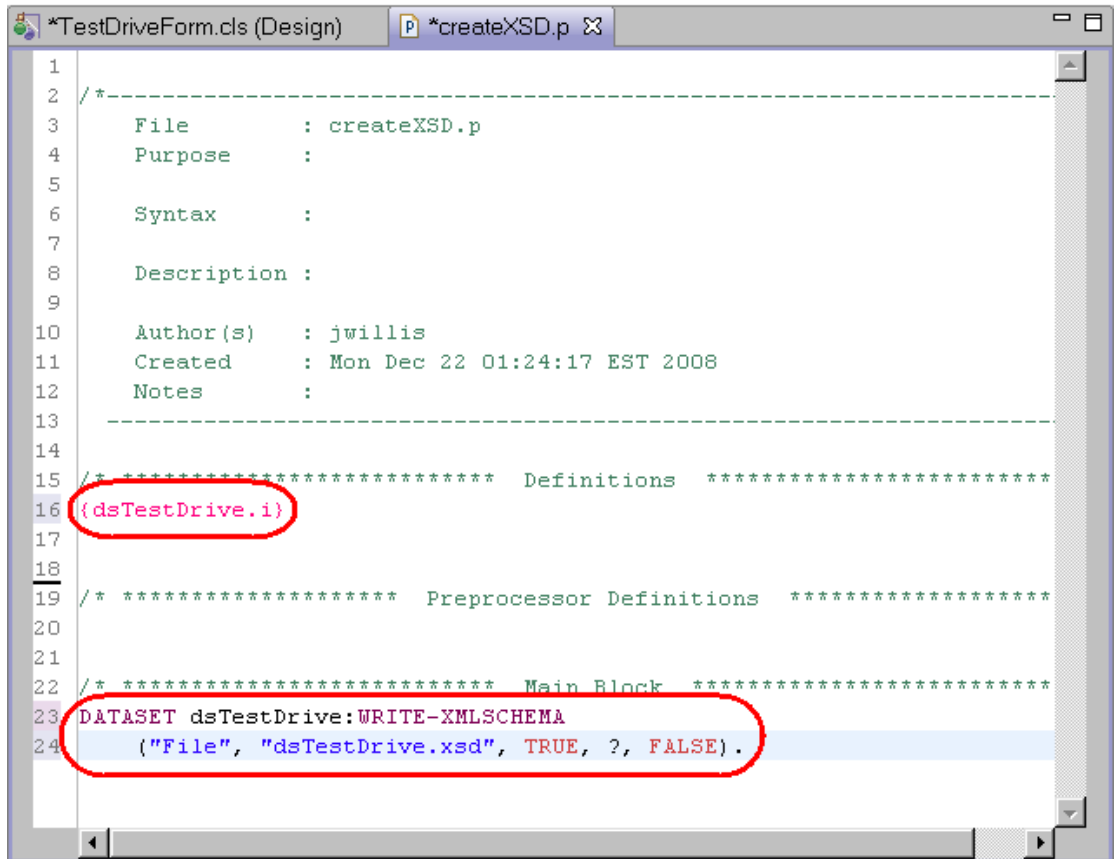


continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Data Source, continued

7. Enter the following code as highlighted below. This code will take the ProDataSet definition created in a previous lab and create an XSD file from the structure.



```
1
2 /*-----
3   File       : createXSD.p
4   Purpose    :
5
6   Syntax     :
7
8   Description :
9
10  Author(s)  : jwillis
11  Created    : Mon Dec 22 01:24:17 EST 2008
12  Notes      :
13 -----
14 /* ***** Definitions *****
15 {dsTestDrive.i}
16
17
18
19 /* ***** Preprocessor Definitions *****
20
21
22 /* ***** Main Block *****
23 DATASET dsTestDrive:WRITE-XMLSCHEMA
24 ("File", "dsTestDrive.xsd", TRUE, ?, FALSE).
```

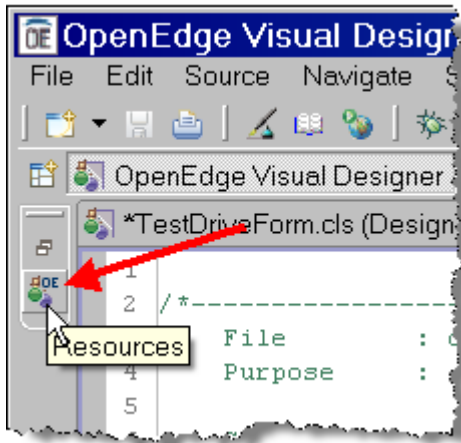
8. **Save** the file and run it using the **Generic Run** launch configuration. If you are prompted to save the form file, select **No**.

continued on next page

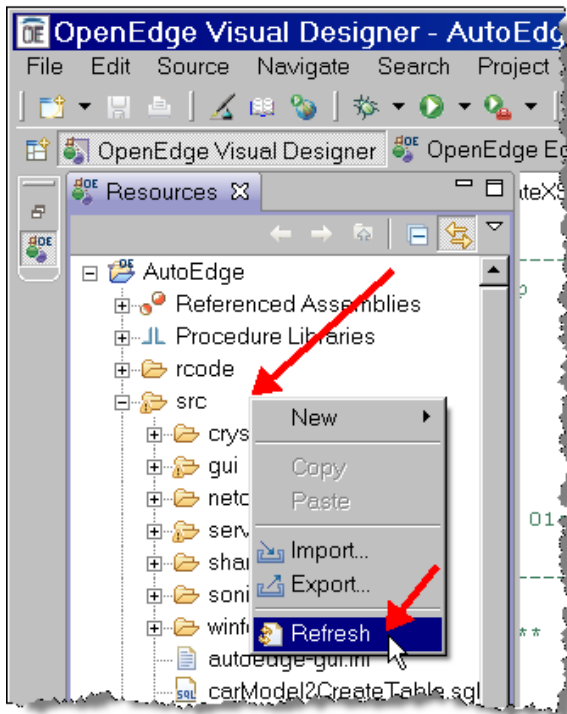
Lab 10 – Using the Visual Designer, continued

Creating a Data Source, continued

9. Once program has executed (control will quickly be returned to you after a splash screen is shown), restore the Resources view by clicking the Resources button.



10. Right-click in the view and select the **Refresh** option.

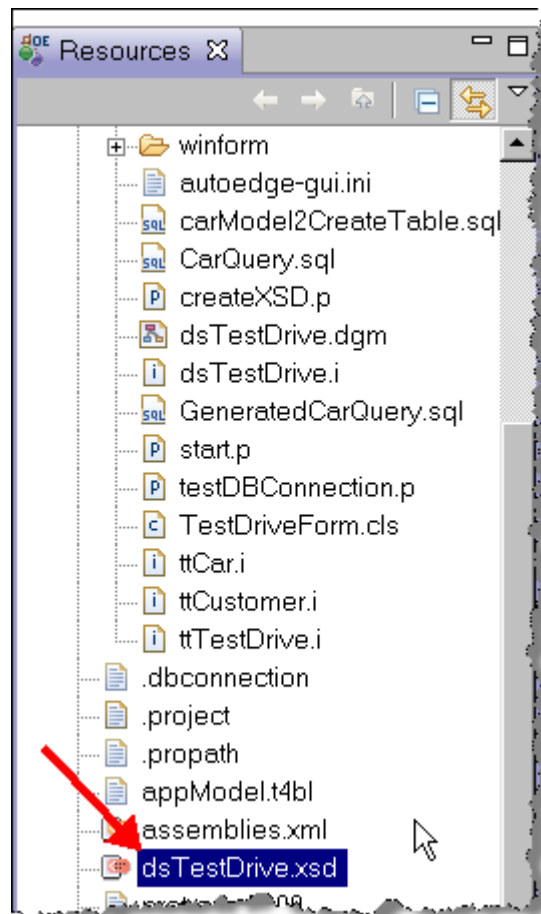


continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Data Source, continued

11. Scroll the view and make sure the dsTestDrive.xsd file was created.



12. Minimize the view.

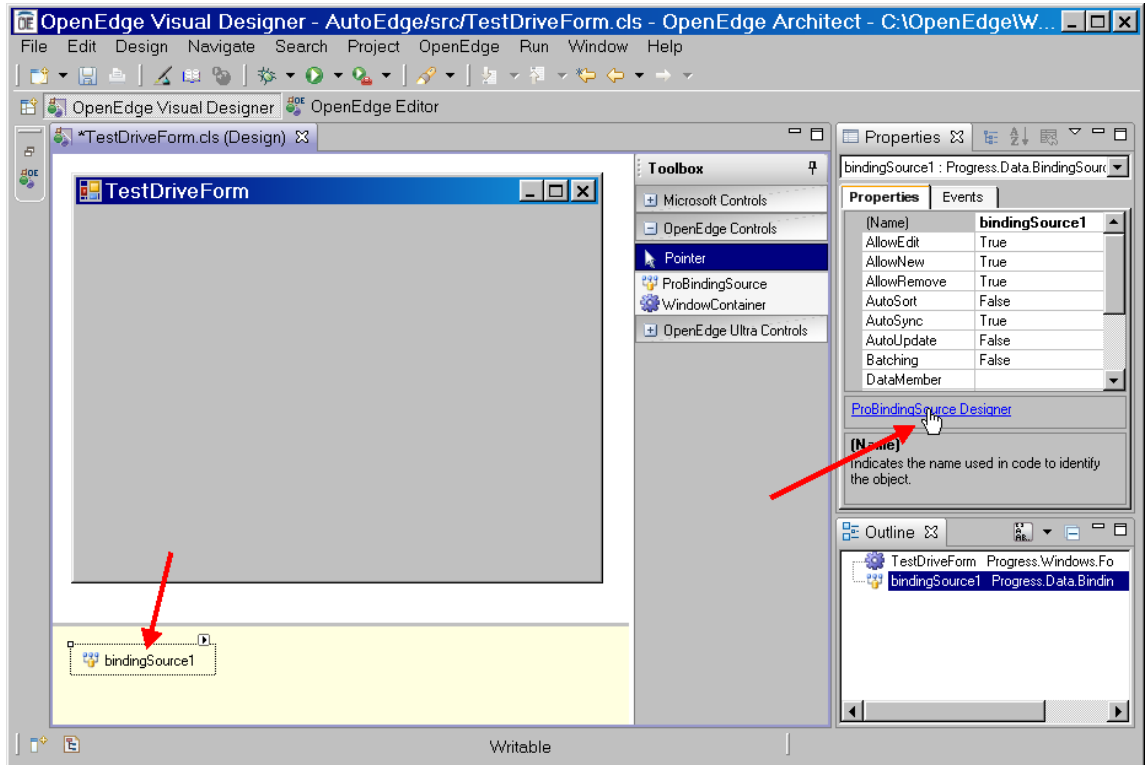
13. Close the createXSD.p file.

continued on next page

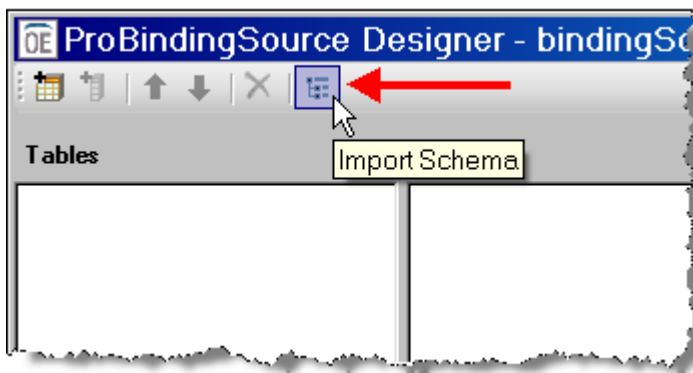
Lab 10 – Using the Visual Designer, continued

Creating a Data Source, continued

14. Select the **bindingSource1** control in the non-visual controls area and then click on the **ProBindingSource Designer** link in the launch wizards area of the Properties view.



15. In the control's start wizard, select the **Import Schema** button.

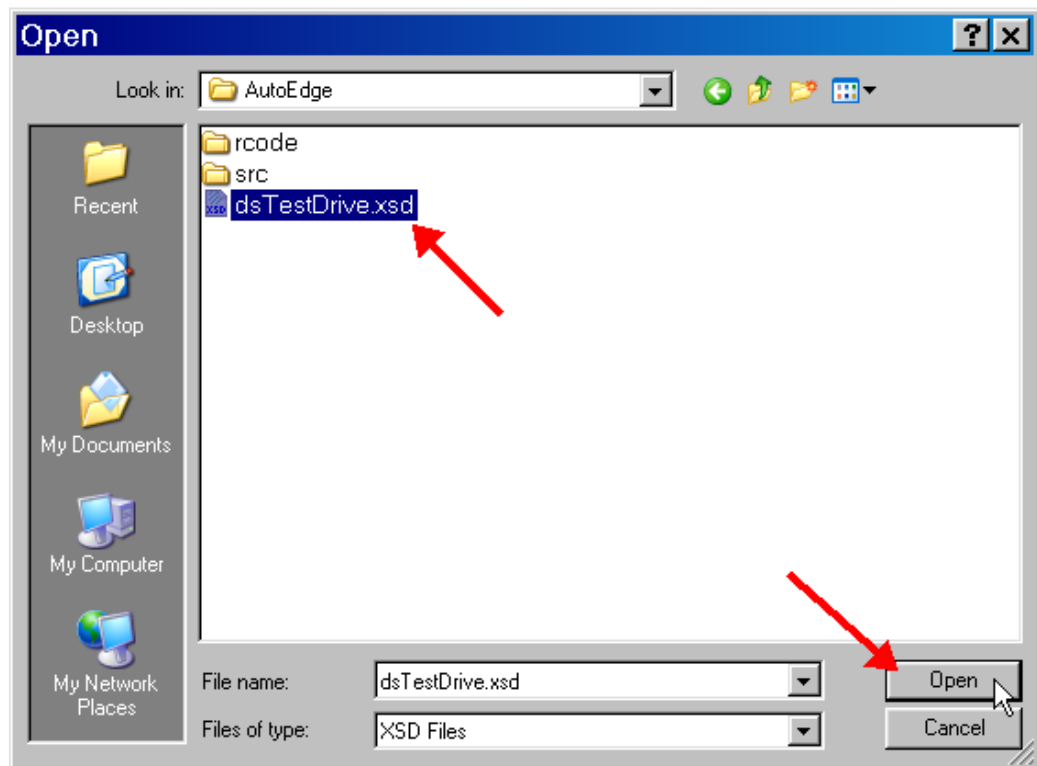


continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Data Source, continued

16. Navigate to the C:\OpenEdge\WRK\AutoEdgeWS\AutoEdge directory and select the **dsTestDrive.xsd** file. Select the **Open** button.

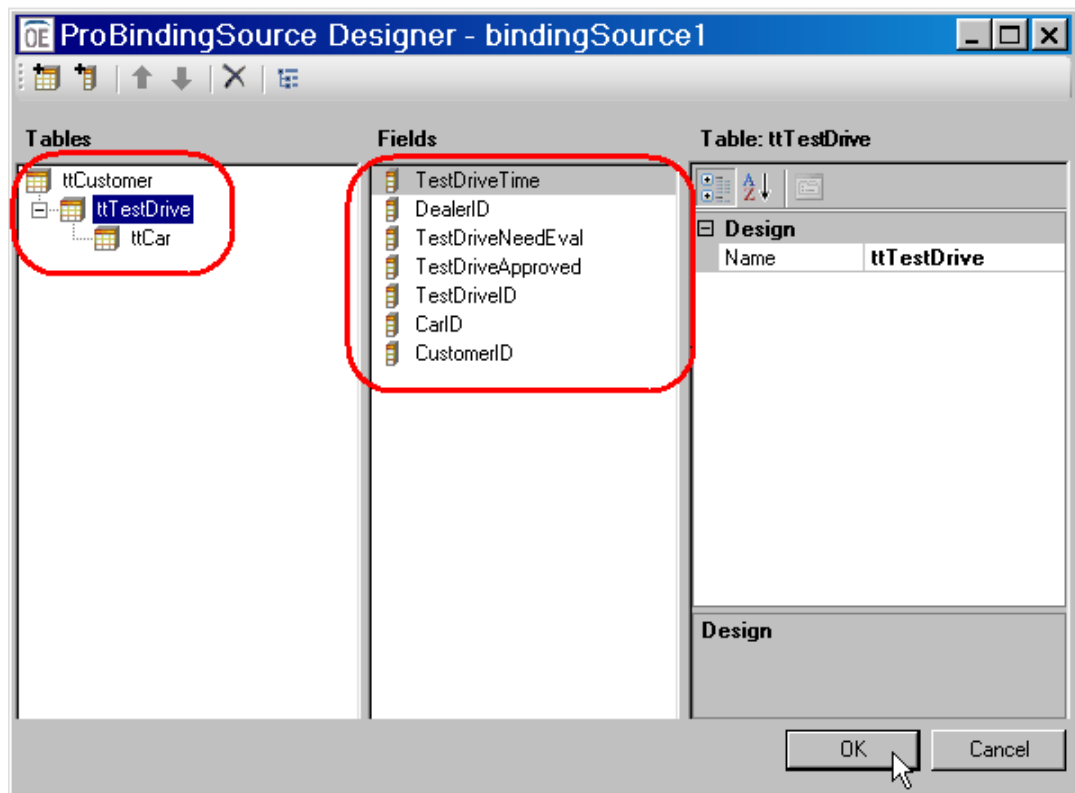


continued on next page

Lab 10 – Using the Visual Designer, continued

Creating a Data Source, continued

17. The wizard will use the file to create the schema structure. Select the OK button to close the wizard.



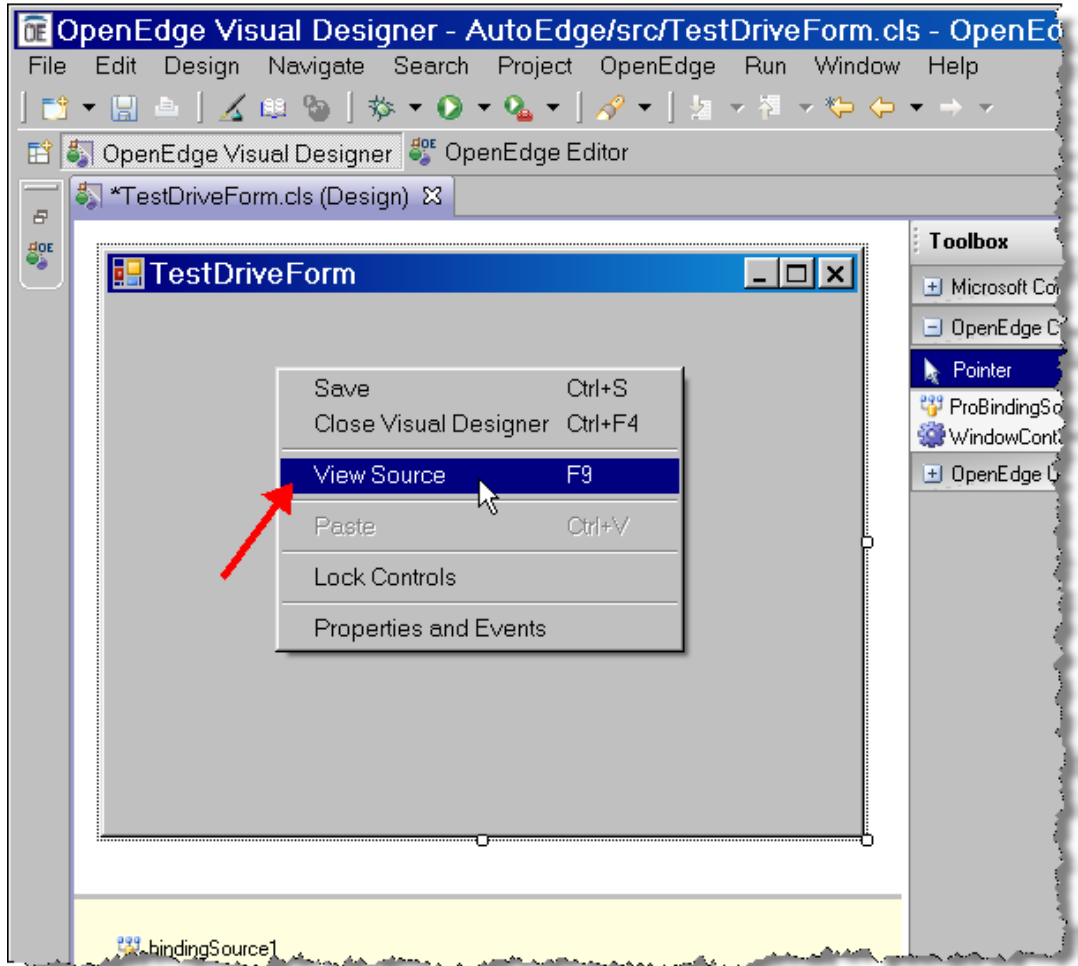
continued on next page

Lab 10 – Using the Visual Designer, continued

Writing Code for the Data Source

Code must be written to tie the ProBindingSource control to the ProDataSet. This section shows how to write that code.

1. Right-click on the form and select the **View Source** option.

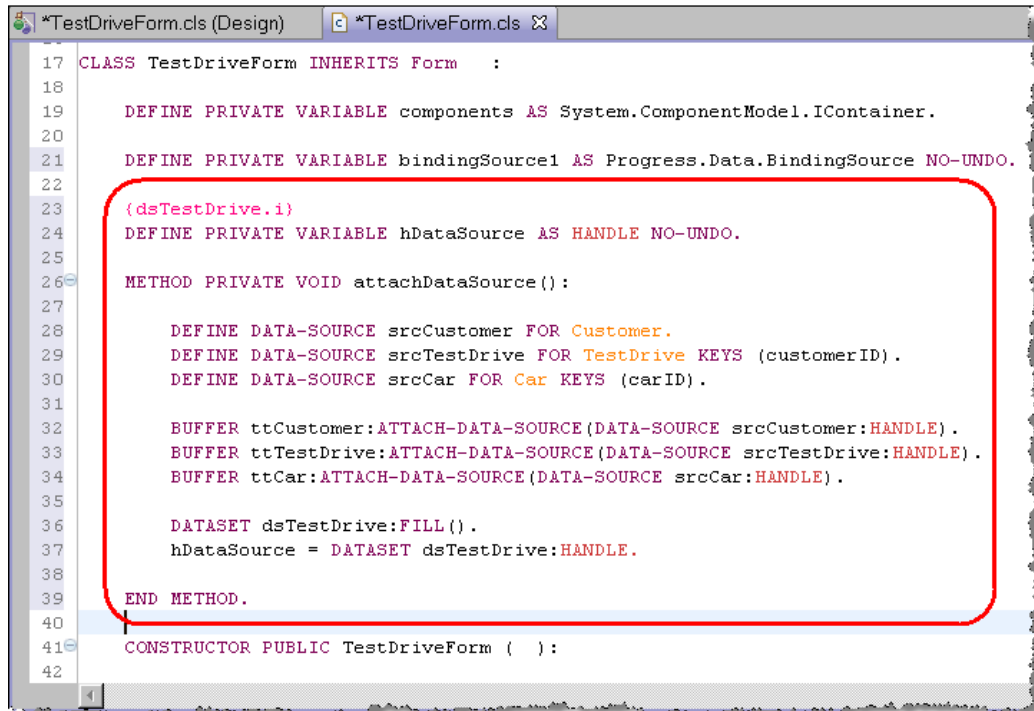


continued on next page

Lab 10 – Using the Visual Designer, continued

Writing Code for the Data Source, continued

2. Add the code shown below to define the ProDataSet and provide a method for populating the ProDataSet with data. You can copy the code from the box below.



```
17 CLASS TestDriveForm INHERITS Form :
18
19     DEFINE PRIVATE VARIABLE components AS System.ComponentModel.IContainer.
20
21     DEFINE PRIVATE VARIABLE bindingSource1 AS Progress.Data.BindingSource NO-UNDO.
22
23     {dsTestDrive.i}
24     DEFINE PRIVATE VARIABLE hDataSource AS HANDLE NO-UNDO.
25
26     METHOD PRIVATE VOID attachDataSource():
27
28         DEFINE DATA-SOURCE srcCustomer FOR Customer.
29         DEFINE DATA-SOURCE srcTestDrive FOR TestDrive KEYS (customerID).
30         DEFINE DATA-SOURCE srcCar FOR Car KEYS (carID).
31
32         BUFFER ttCustomer:ATTACH-DATA-SOURCE(DATA-SOURCE srcCustomer:HANDLE).
33         BUFFER ttTestDrive:ATTACH-DATA-SOURCE(DATA-SOURCE srcTestDrive:HANDLE).
34         BUFFER ttCar:ATTACH-DATA-SOURCE(DATA-SOURCE srcCar:HANDLE).
35
36         DATASET dsTestDrive:FILL().
37         hDataSource = DATASET dsTestDrive:HANDLE.
38
39     END METHOD.
40
41     CONSTRUCTOR PUBLIC TestDriveForm ( ):
42
```

Program: TestDriveForm.cls

```
{dsTestDrive.i}
DEFINE PRIVATE VARIABLE hDataSource AS HANDLE NO-UNDO.

METHOD PRIVATE VOID attachDataSource():

    DEFINE DATA-SOURCE srcCustomer FOR Customer.
    DEFINE DATA-SOURCE srcTestDrive FOR TestDrive KEYS (customerID).
    DEFINE DATA-SOURCE srcCar FOR Car KEYS (carID).

    BUFFER ttCustomer:ATTACH-DATA-SOURCE(DATA-SOURCE srcCustomer:HANDLE).
    BUFFER ttTestDrive:ATTACH-DATA-SOURCE(DATA-SOURCE srcTestDrive:HANDLE).
    BUFFER ttCar:ATTACH-DATA-SOURCE(DATA-SOURCE srcCar:HANDLE).

    DATASET dsTestDrive:FILL().
    hDataSource = DATASET dsTestDrive:HANDLE.

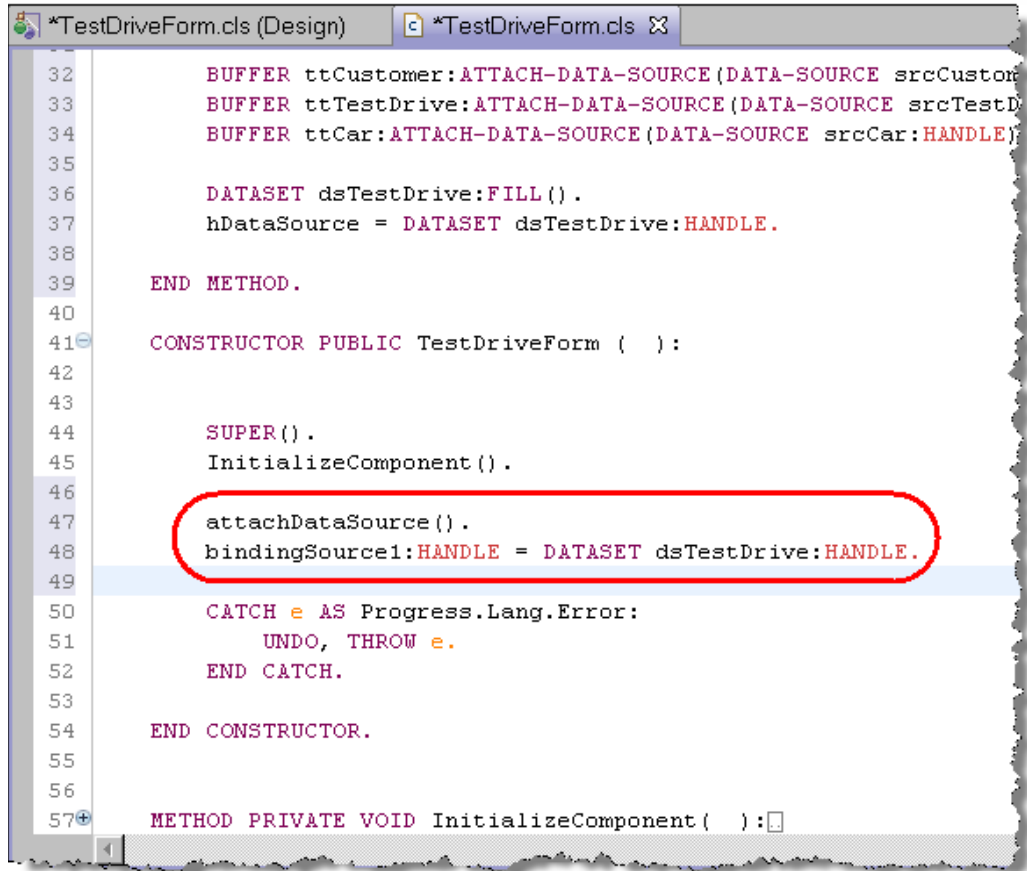
END METHOD.
```

continued on next page

Lab 10 – Using the Visual Designer, continued

Writing Code for the Data Source, continued

3. Add the code shown below to run the method that populates the ProDataSet at initialization and binds the ProBindingSource control to the ProDataSet. You can copy the code from the box below.



```
32     BUFFER ttCustomer:ATTACH-DATA-SOURCE (DATA-SOURCE srcCustom
33     BUFFER ttTestDrive:ATTACH-DATA-SOURCE (DATA-SOURCE srcTestD
34     BUFFER ttCar:ATTACH-DATA-SOURCE (DATA-SOURCE srcCar:HANDLE)
35
36     DATASET dsTestDrive:FILL().
37     hDataSource = DATASET dsTestDrive:HANDLE.
38
39     END METHOD.
40
41     CONSTRUCTOR PUBLIC TestDriveForm ( ):
42
43
44     SUPER().
45     InitializeComponent().
46
47     attachDataSource().
48     bindingSource1:HANDLE = DATASET dsTestDrive:HANDLE.
49
50     CATCH e AS Progress.Lang.Error:
51         UNDO, THROW e.
52     END CATCH.
53
54     END CONSTRUCTOR.
55
56
57     METHOD PRIVATE VOID InitializeComponent ( ):□
```

Program: TestDriveForm.cls

```
attachDataSource().
bindingSource1:HANDLE = DATASET dsTestDrive:HANDLE.
```

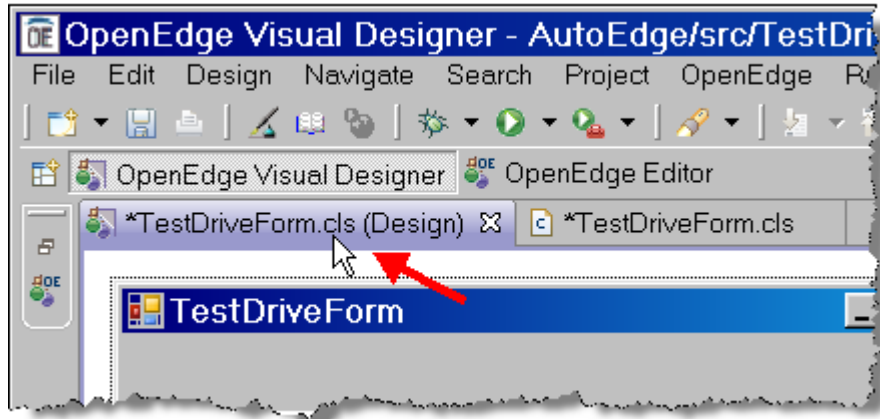
continued on next page

Lab 10 – Using the Visual Designer, continued

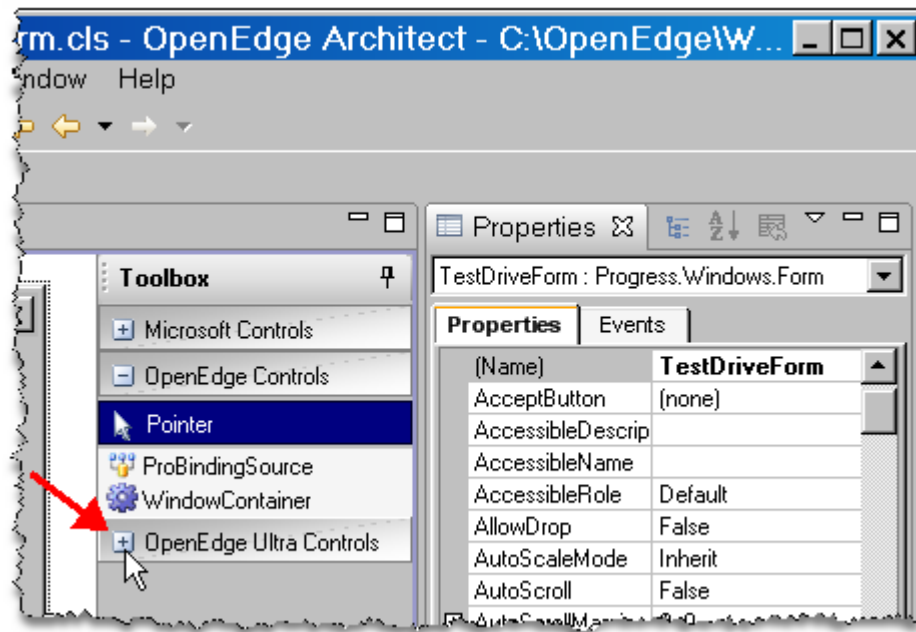
Working with Visual Controls

The Visual Designer offers many visual .NET controls that can be used to create a client form. This section shows how to work with visual controls.

1. Select the **TestDriveForm.cls (Design)** tab to switch back to the visual editor.



2. Expand the **OpenEdge Ultra Controls** group in the Toolbox by clicking on the + (Plus) button.

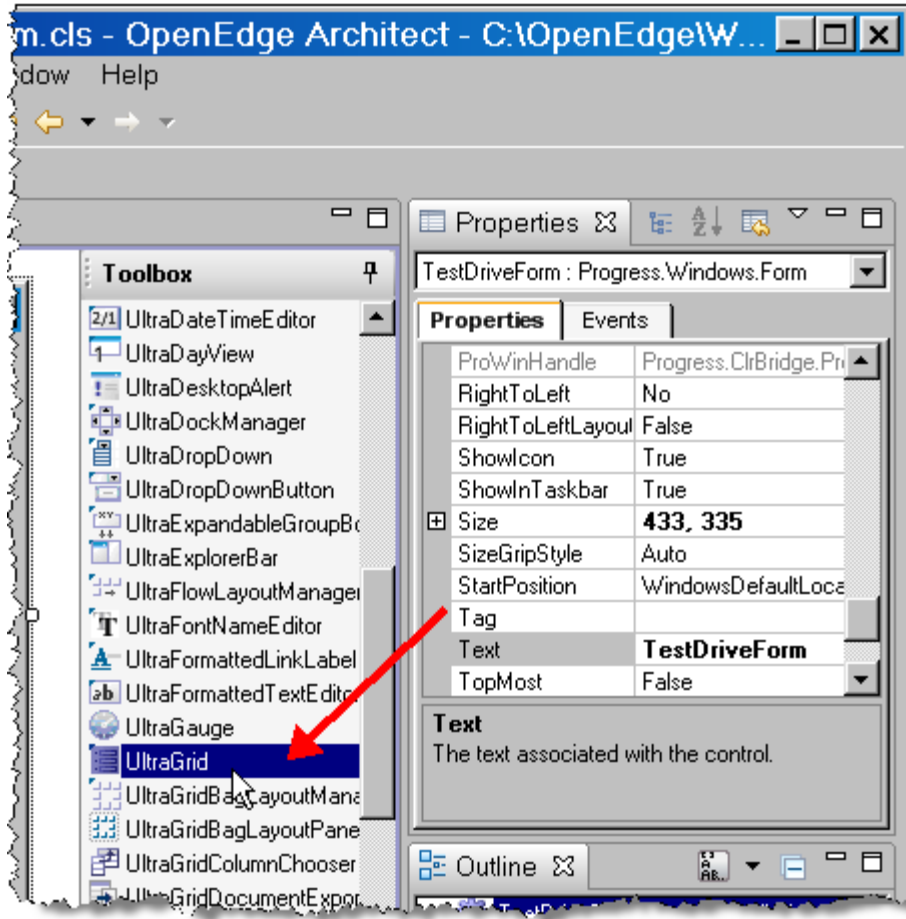


continued on next page

Lab 10 – Using the Visual Designer, continued

Working with Visual Controls, continued

3. Scroll the Toolbox and find the **UltraGrid** control. Select the control.

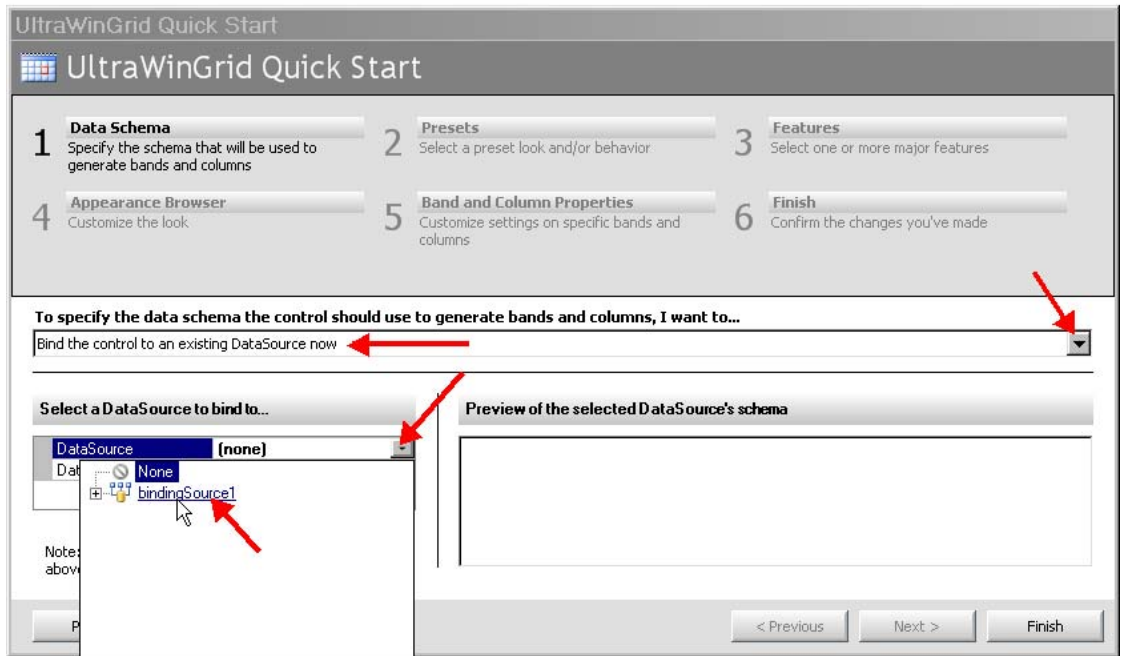


continued on next page

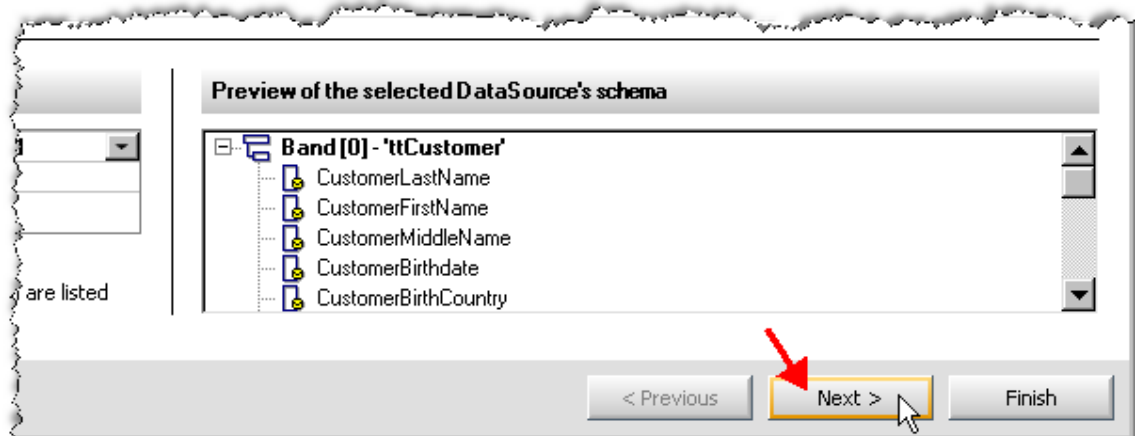
Lab 10 – Using the Visual Designer, continued

Working with Visual Controls, continued

4. Click on the form to place the control. The control's start wizard will be shown.
5. In the wizard, select **Bind the control to an existing DataSource now** option from the pull-down menu. Then, select the **bindingSource1** option from the DataSource pull-down menu.



6. Click the **Next** button.

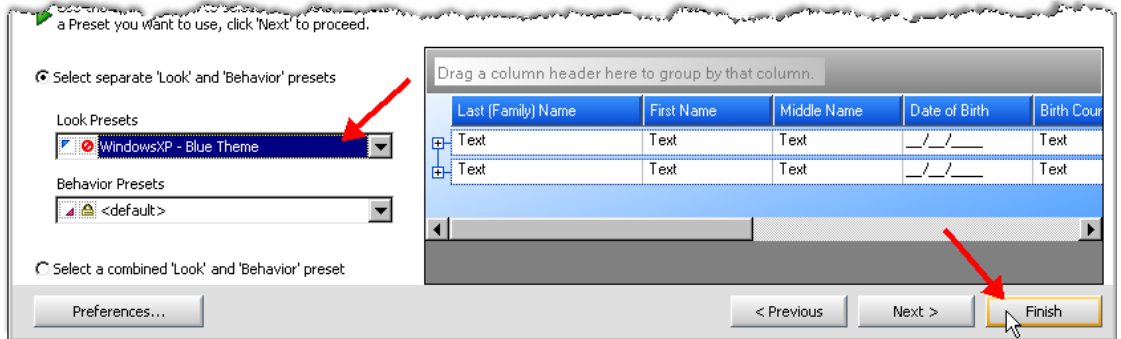


continued on next page

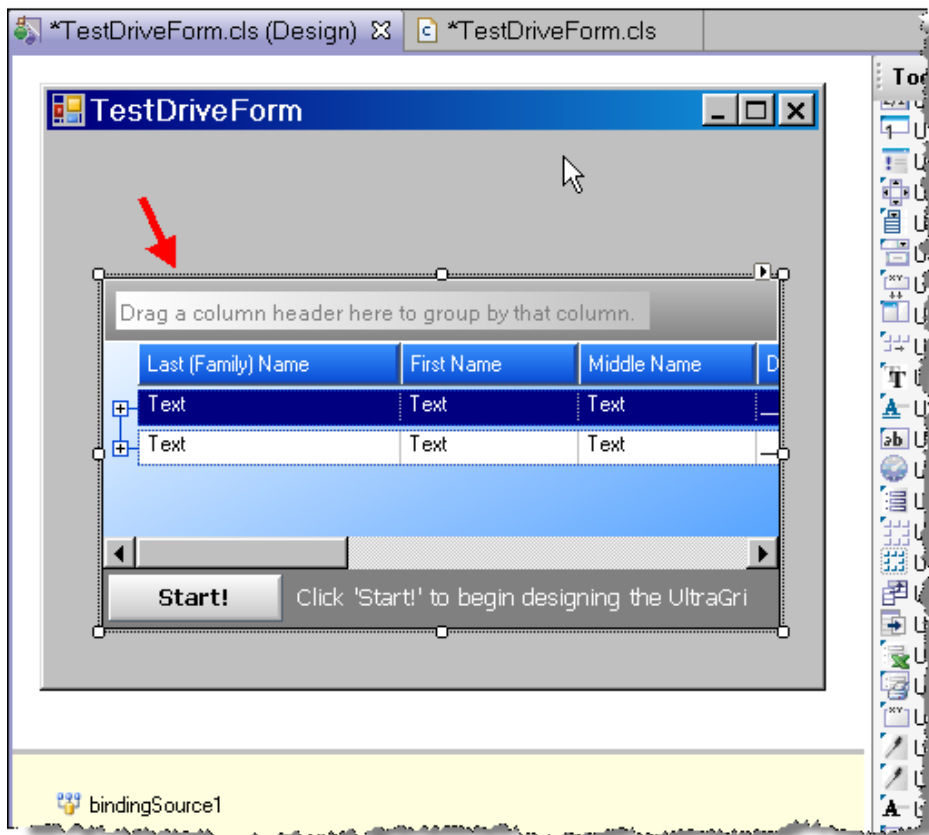
Lab 10 – Using the Visual Designer, continued

Working with Visual Controls, continued

7. In the Look Presets pull-down select the **WindowsXP – Blue Theme** option and then click the **Finish** button.



8. Back in the form, resize the visual control to fit in the form.

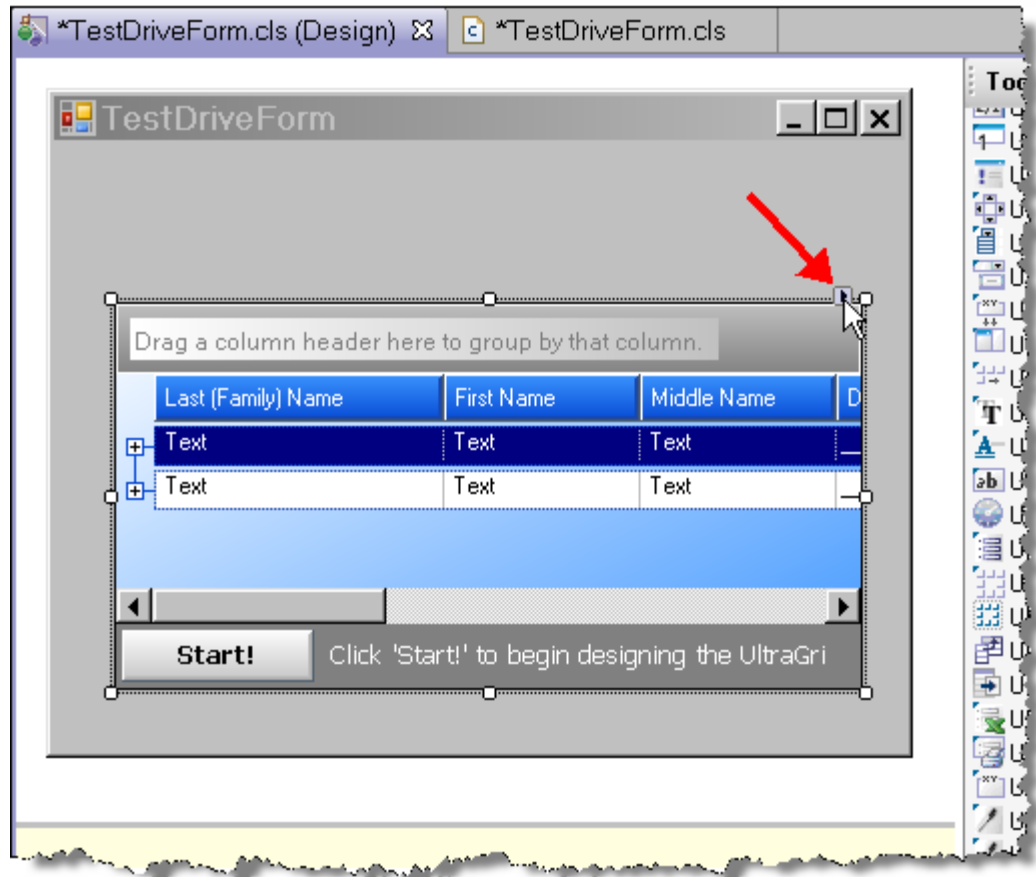


continued on next page

Lab 10 – Using the Visual Designer, continued

Working with Visual Controls, continued

9. Click the Smart Tag button for the control.

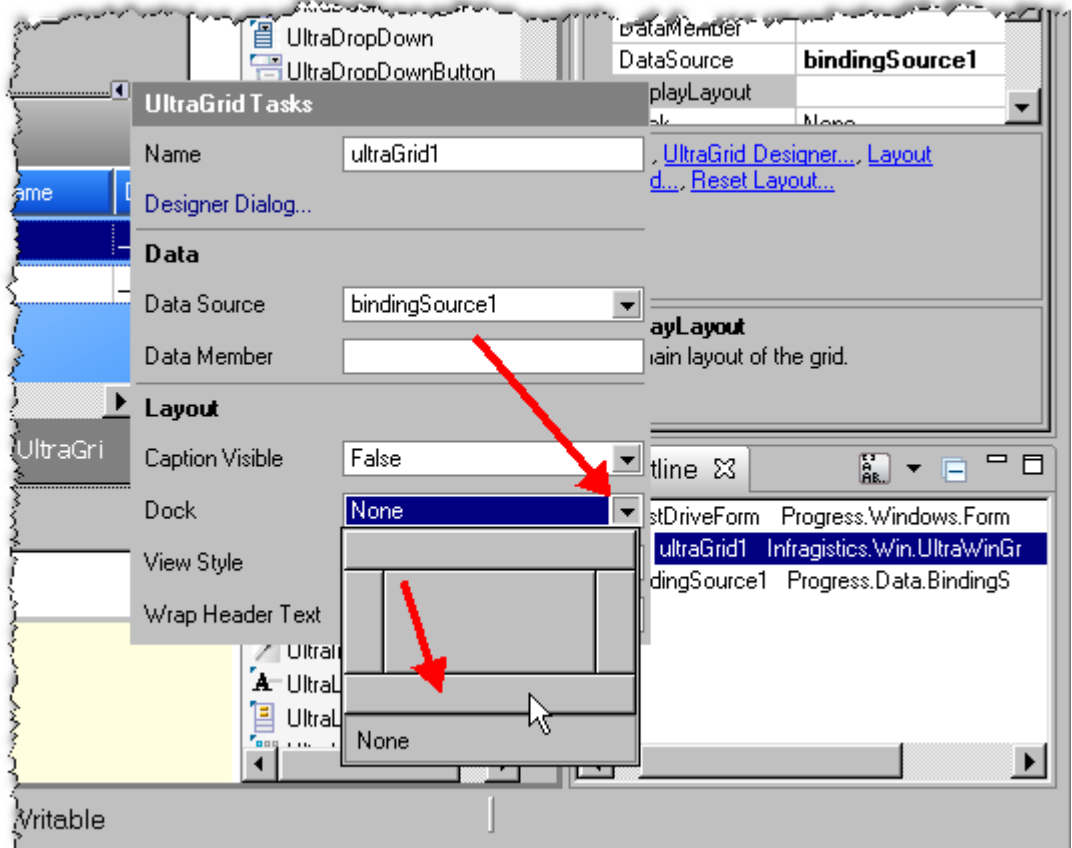


continued on next page

Lab 10 – Using the Visual Designer, continued

Working with Visual Controls, continued

10. In the Smart Tag pop-up box, select the Dock drop-down and click on the bottom button to dock the visual control to the bottom of the form.

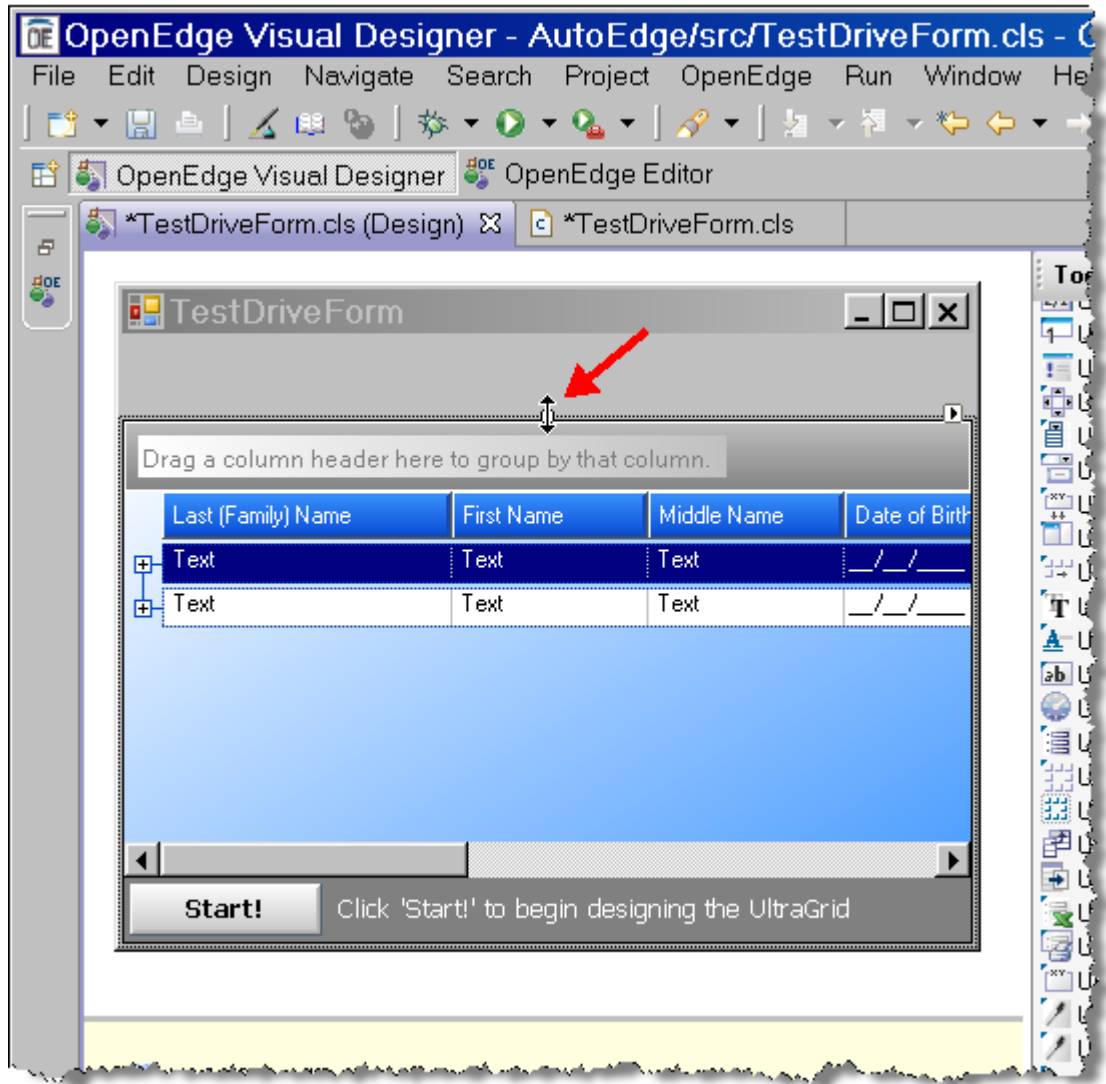


continued on next page

Lab 10 – Using the Visual Designer, continued

Working with Visual Controls, continued

11. Resize the control to be similar to the image below.



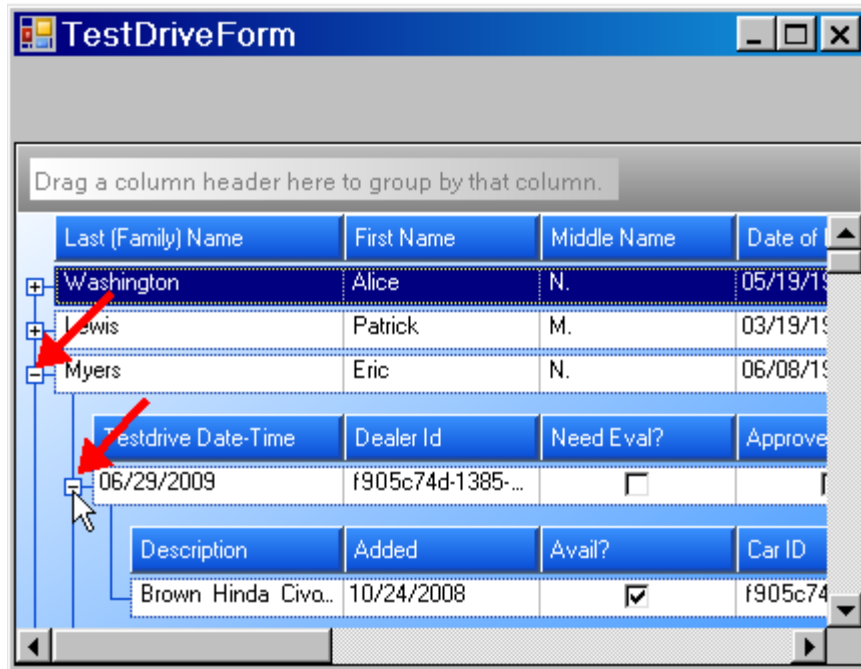
12. Save the file.
13. Run the file using the **Generic Run** configuration.

continued on next page

Lab 10 – Using the Visual Designer, continued

Working with Visual Controls, continued

14. Test the form by clicking on some of the nodes in the grid.



15. Close the program when finished.

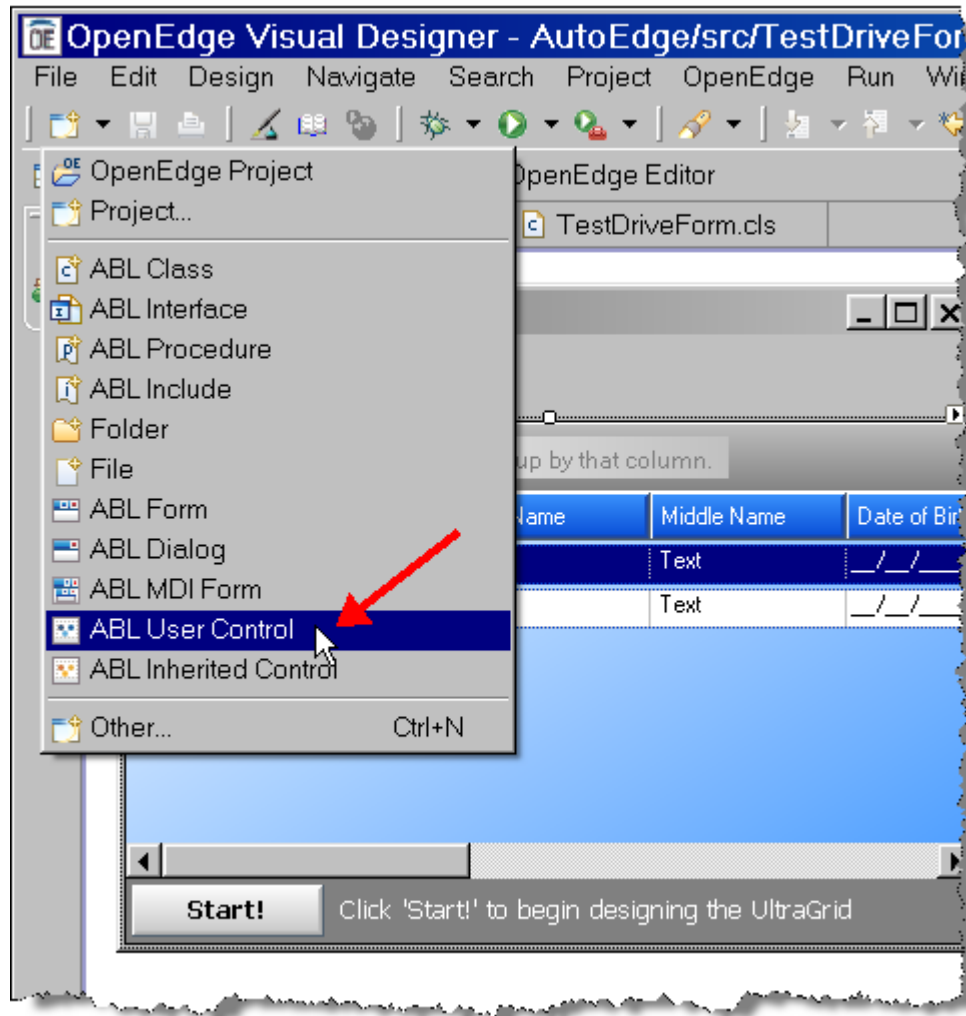
continued on next page

Lab 10 – Using the Visual Designer, continued

Creating User Controls

You can extend existing .NET controls by combining them to form a new control. This section shows how to create a user control.

1. Select the down-arrow beside the New button and select **ABL User Control**.

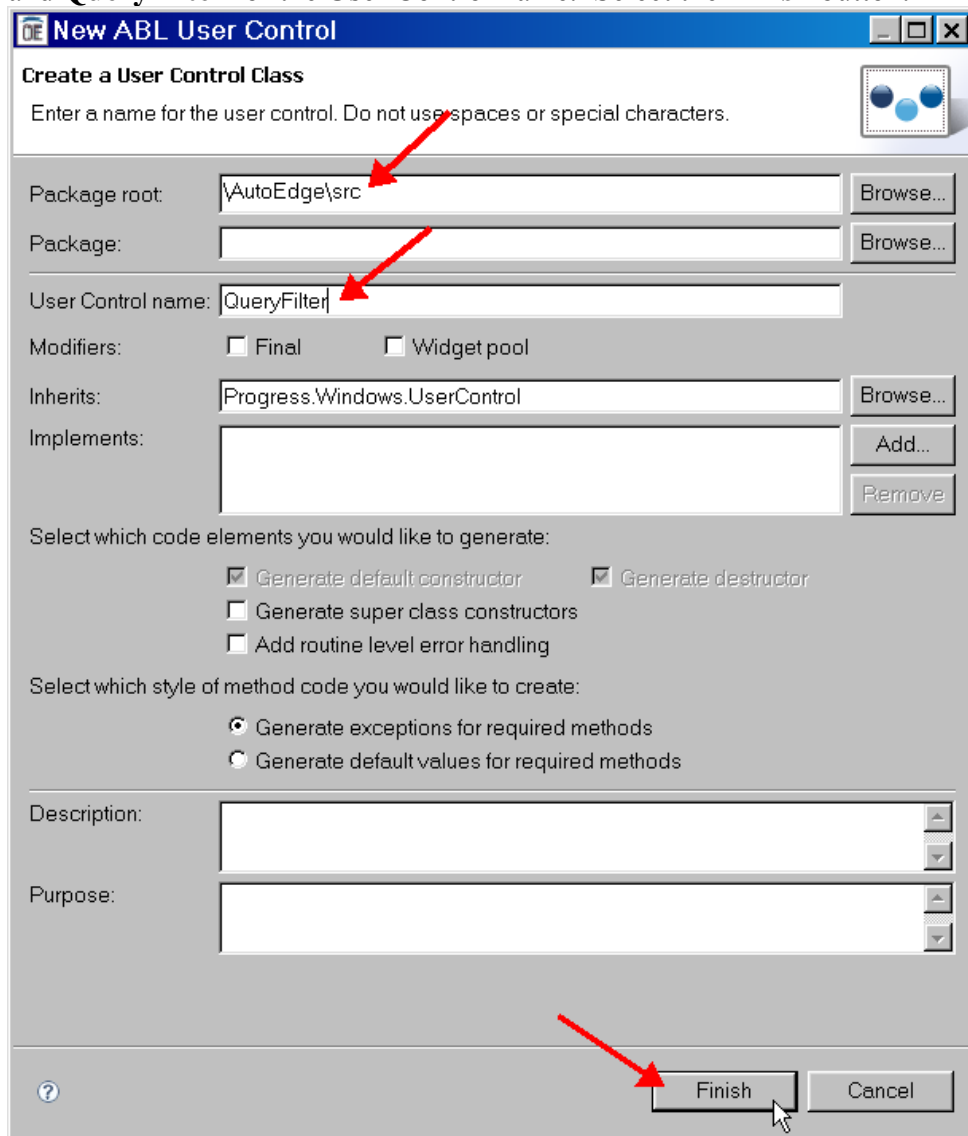


continued on next page

Lab 10 – Using the Visual Designer, continued

Creating User Controls, continued

2. In the New ABL User Control wizard, enter **\AutoEdge\src** for the Package root and **QueryFilter** for the User Control name. Select the **Finish** button.



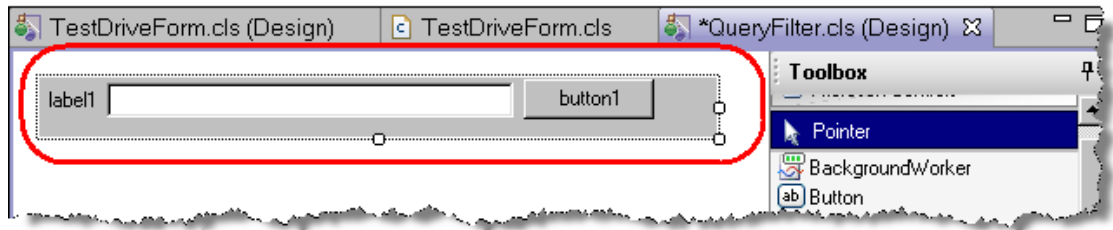
3. Expand the Microsoft Controls group in the Toolbox.

continued on next page

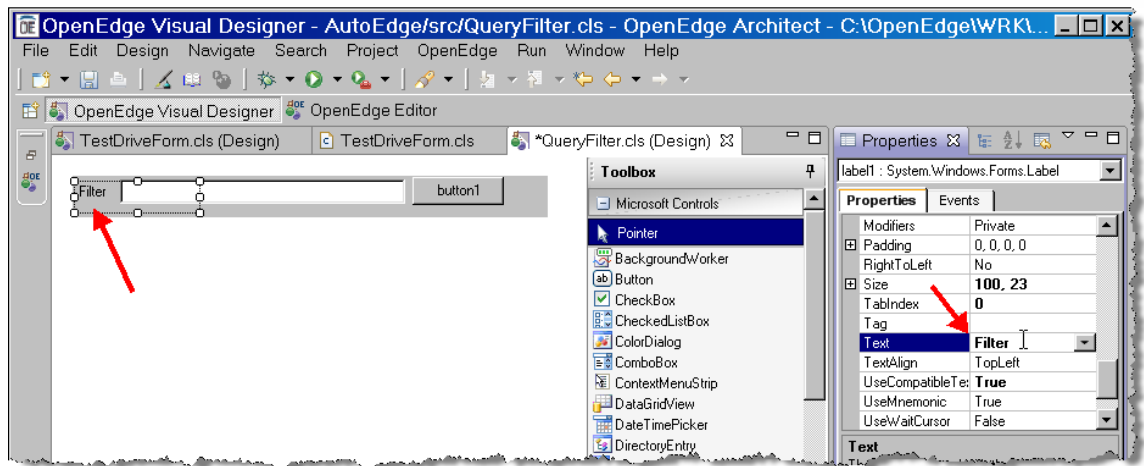
Lab 10 – Using the Visual Designer, continued

Creating User Controls, continued

4. Place and align a **Label**, **TextBox**, and **Button** control in the form and resize as shown below.



5. Click on the **label1** control in the form. In the Properties view, scroll to find the Text property and enter **Filter** for the property.



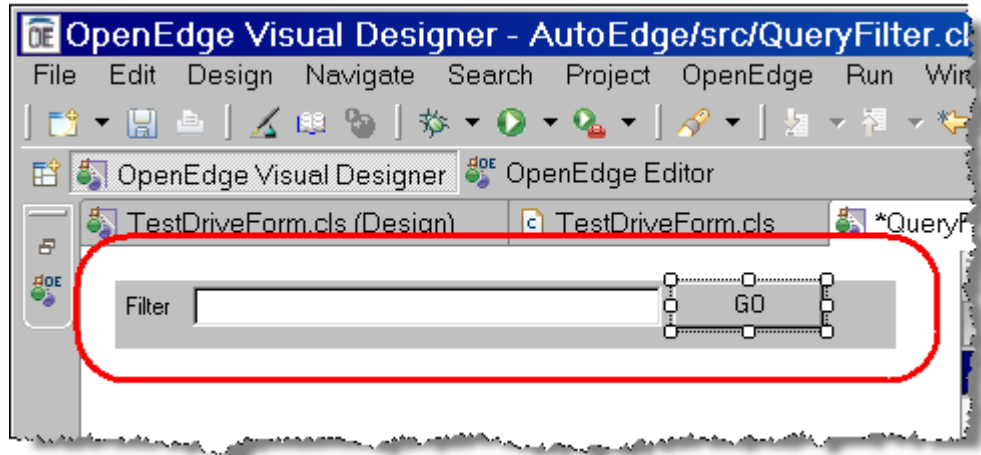
6. Select the **TextBox** control in the form and set the Name property to **FilterBox**.
7. Select the **Button** control in the form and set the Text property to **GO** and set the Name property to **FilterButton**.

continued on next page

Lab 10 – Using the Visual Designer, continued

Creating User Controls, continued

8. The User Control should look like the following:



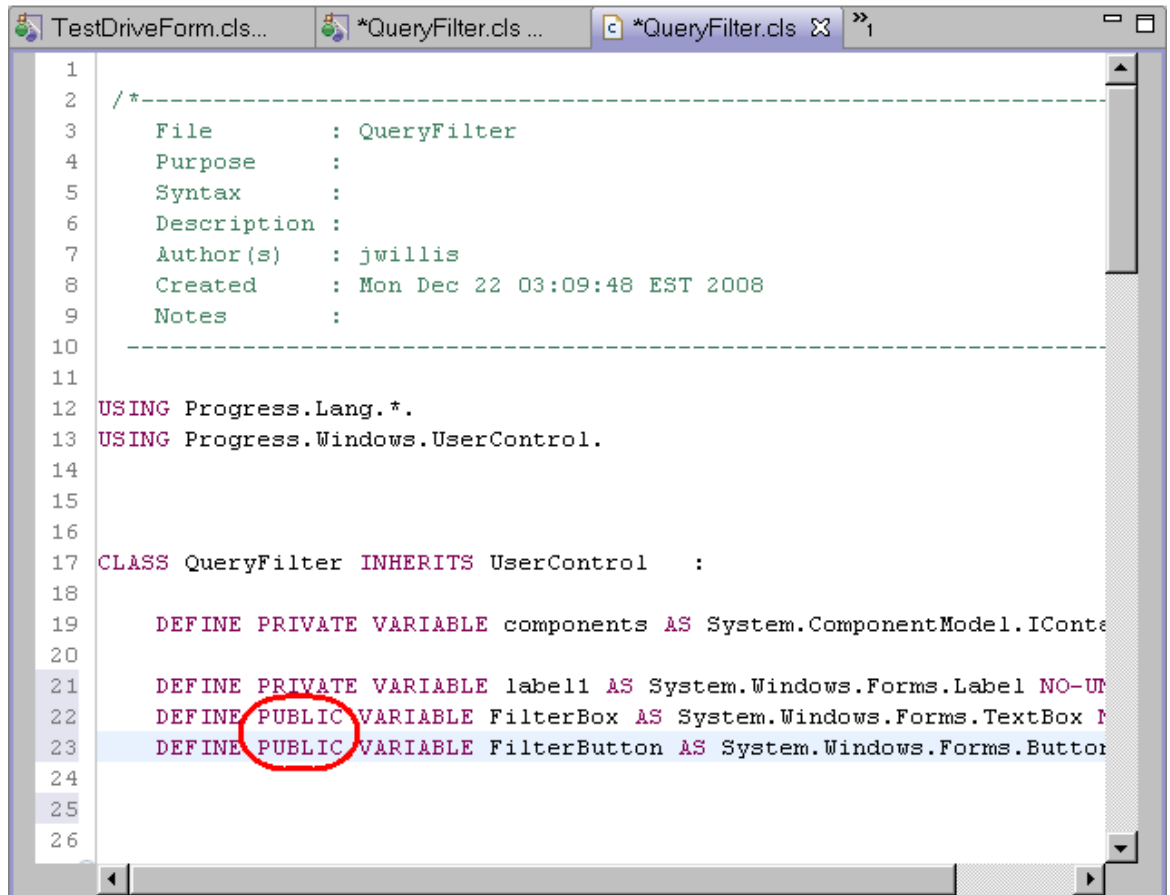
9. Right-click on the form and select the View Source option.

continued on next page

Lab 10 – Using the Visual Designer, continued

Creating User Controls, continued

10. Change the FilterBox and FilterButton variables to **PUBLIC**.



```
1
2  /*-----
3      File       : QueryFilter
4      Purpose    :
5      Syntax     :
6      Description:
7      Author(s)  : jwillis
8      Created    : Mon Dec 22 03:09:48 EST 2008
9      Notes      :
10     -----
11
12     USING Progress.Lang.*.
13     USING Progress.Windows.UserControl.
14
15
16
17     CLASS QueryFilter INHERITS UserControl :
18
19         DEFINE PRIVATE VARIABLE components AS System.ComponentModel.IConte
20
21         DEFINE PRIVATE VARIABLE label1 AS System.Windows.Forms.Label NO-UP
22         DEFINE PUBLIC VARIABLE FilterBox AS System.Windows.Forms.TextBox P
23         DEFINE PUBLIC VARIABLE FilterButton AS System.Windows.Forms.Button
24
25
26
```

11. Save the QueryFilter file and close both the ABL and Visual Designer editors for the control.

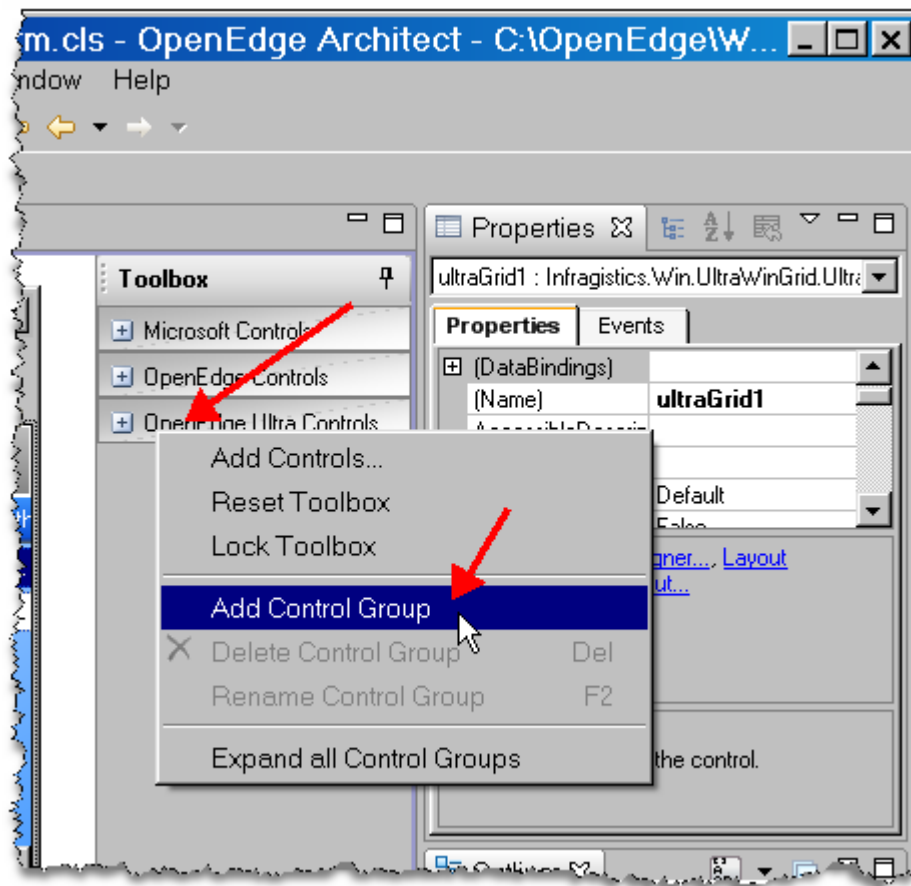
continued on next page

Lab 10 – Using the Visual Designer, continued

Working with User Controls

The following section shows how to create a Toolbox entry for the new User Control, place it in a form, and write code for it.

1. Collapse all the control groups in the Toolbox.
2. Right-click on one of the control groups and select the **Add Control Group** option.

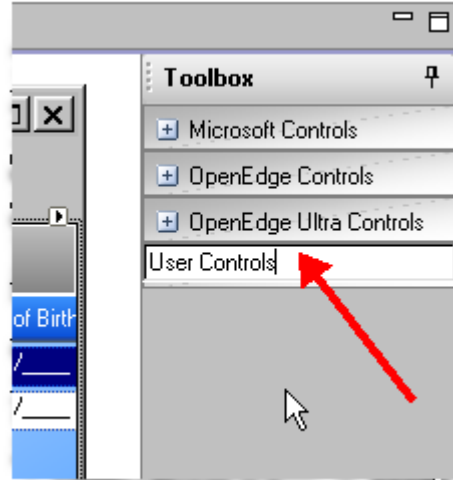


continued on next page

Lab 10 – Using the Visual Designer, continued

Working with User Controls, continued

3. Type **User Controls** and press **Enter**.

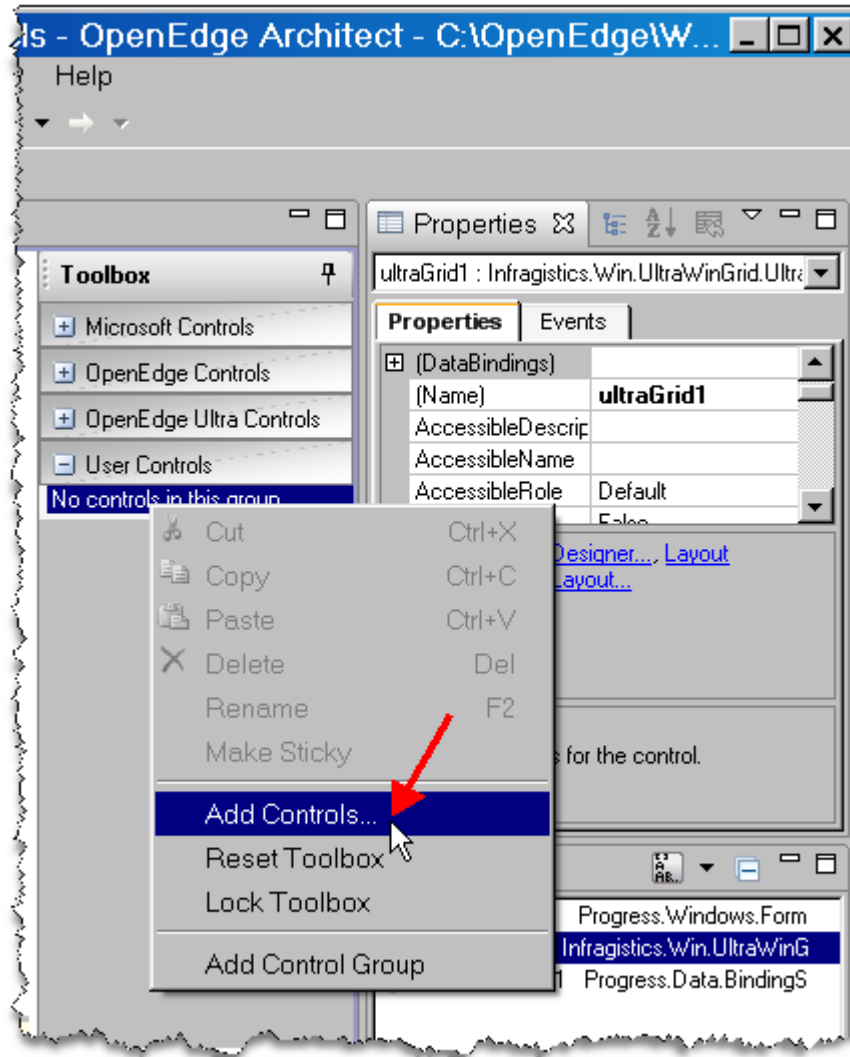


continued on next page

Lab 10 – Using the Visual Designer, continued

Working with User Controls, continued

4. Right-click on the new control group and select the **Add Controls** option.

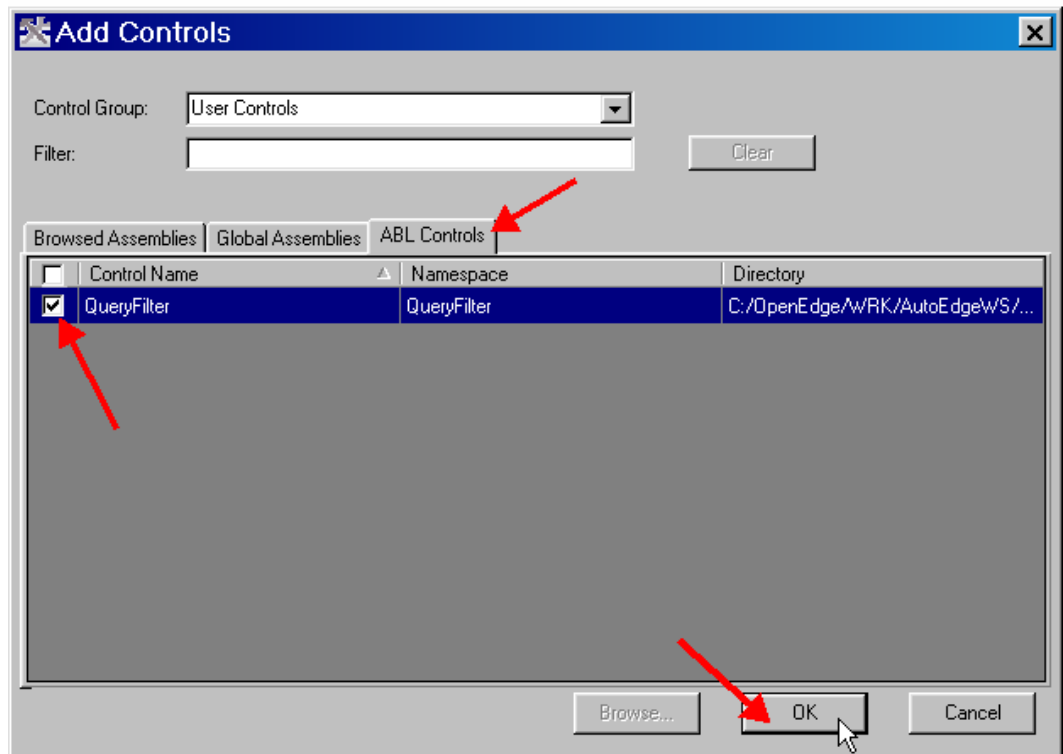


continued on next page

Lab 10 – Using the Visual Designer, continued

Working with User Controls, continued

5. Select the **ABL Controls** tab. Check the box next to the **QueryFilter** control and select the **OK** button.

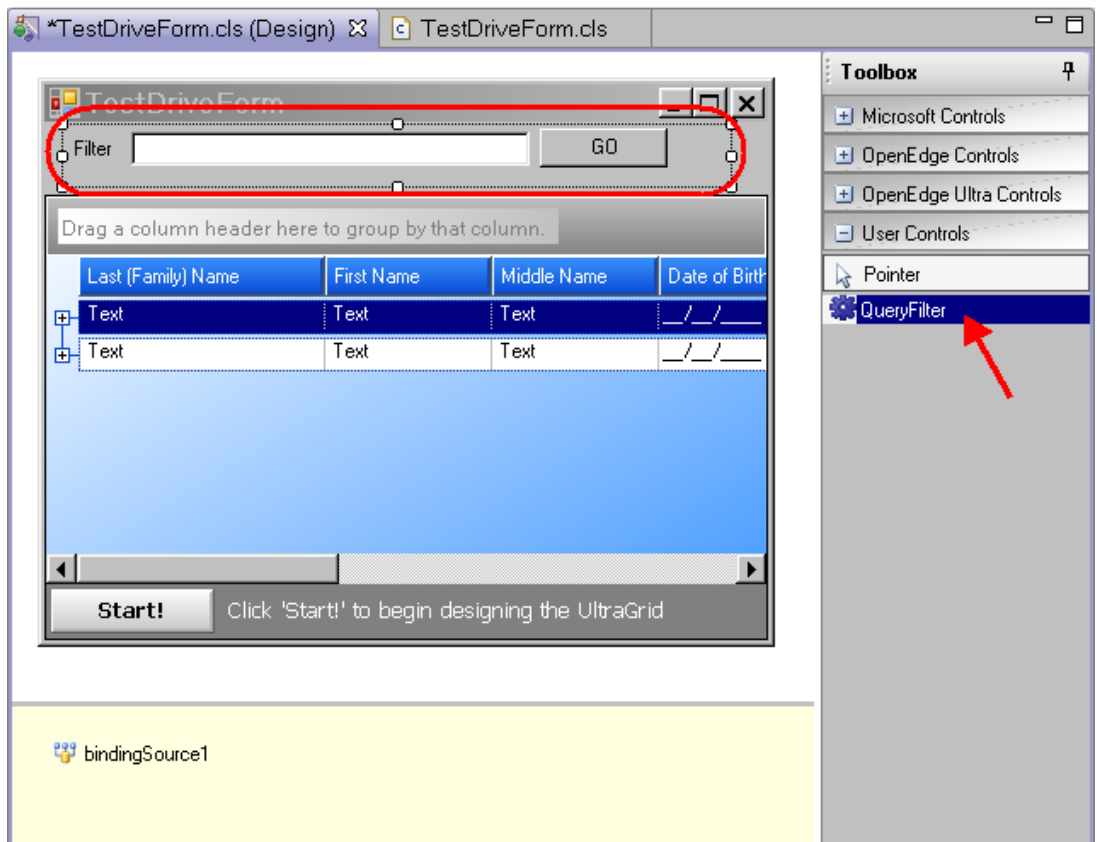


continued on next page

Lab 10 – Using the Visual Designer, continued

Working with User Controls, continued

6. Select the QueryFilter control in the Toolbox and place and position it in the form as shown below:

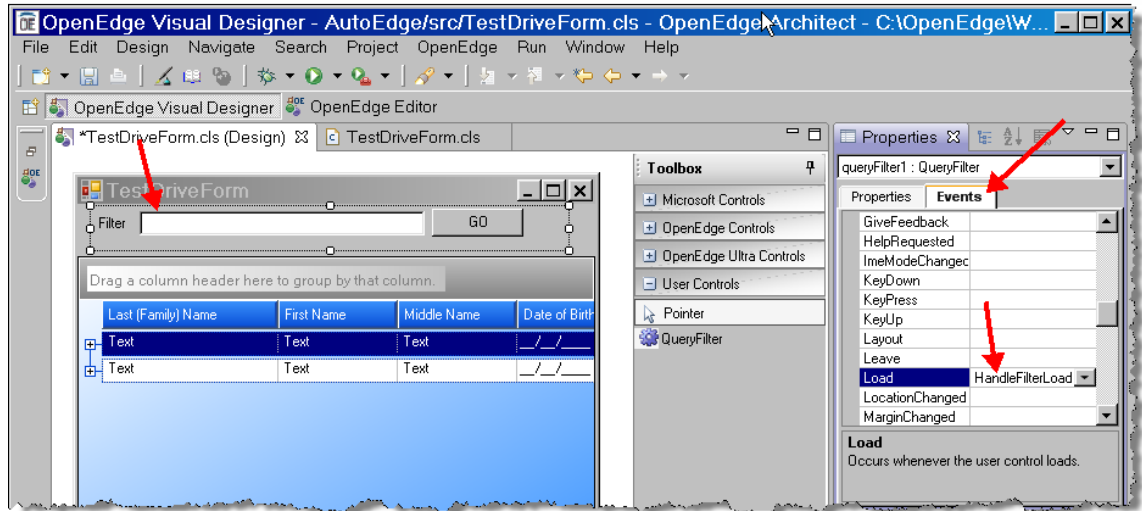


continued on next page

Lab 10 – Using the Visual Designer, continued

Working with User Controls, continued

7. While the placed control is selected in the form, select the Events tab in the Properties view. Scroll to find the Load event and type **HandleFilterLoad** for its method value. Press the **Enter** key.

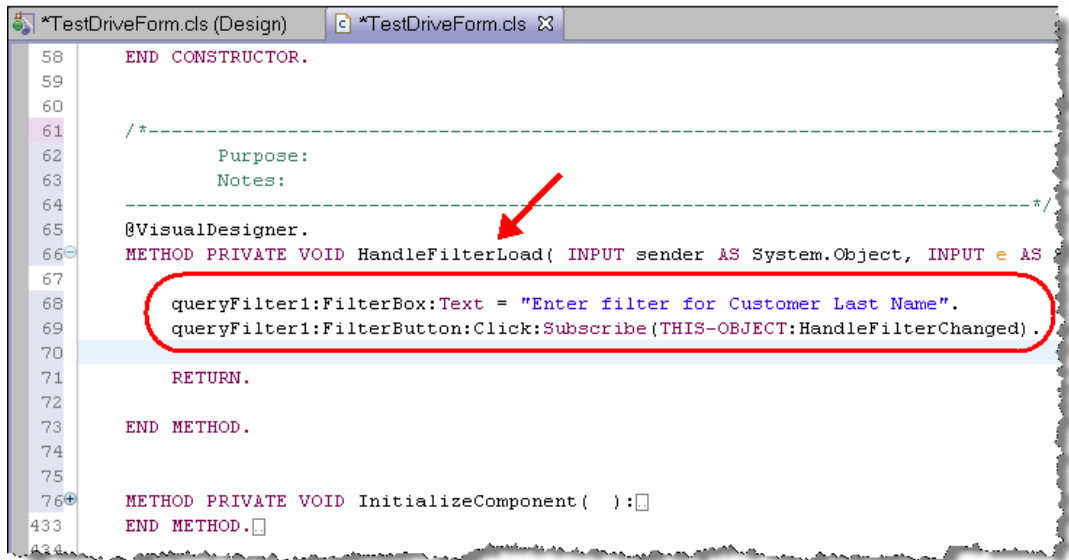


continued on next page

Lab 10 – Using the Visual Designer, continued

Working with User Controls, continued

8. Architect will create a new method using the method name you just entered. You also will be taken to the editor for the form. Enter the following code in the new method to set the Text property for the FilterBox and subscribe to the FilterButton's click event. You can copy the code from the box below.



```
58     END CONSTRUCTOR.  
59  
60  
61     /*-----  
62         Purpose:  
63         Notes:  
64     -----*/  
65     @VisualDesigner.  
66     METHOD PRIVATE VOID HandleFilterLoad( INPUT sender AS System.Object, INPUT e AS  
67  
68         queryFilter1:FilterBox:Text = "Enter filter for Customer Last Name".  
69         queryFilter1:FilterButton:Click:Subscribe(THIS-OBJECT:HandleFilterChanged).  
70  
71     RETURN.  
72  
73     END METHOD.  
74  
75  
76     METHOD PRIVATE VOID InitializeComponent( ):□  
433     END METHOD.□  
434
```

Program: TestDriveForm.cls

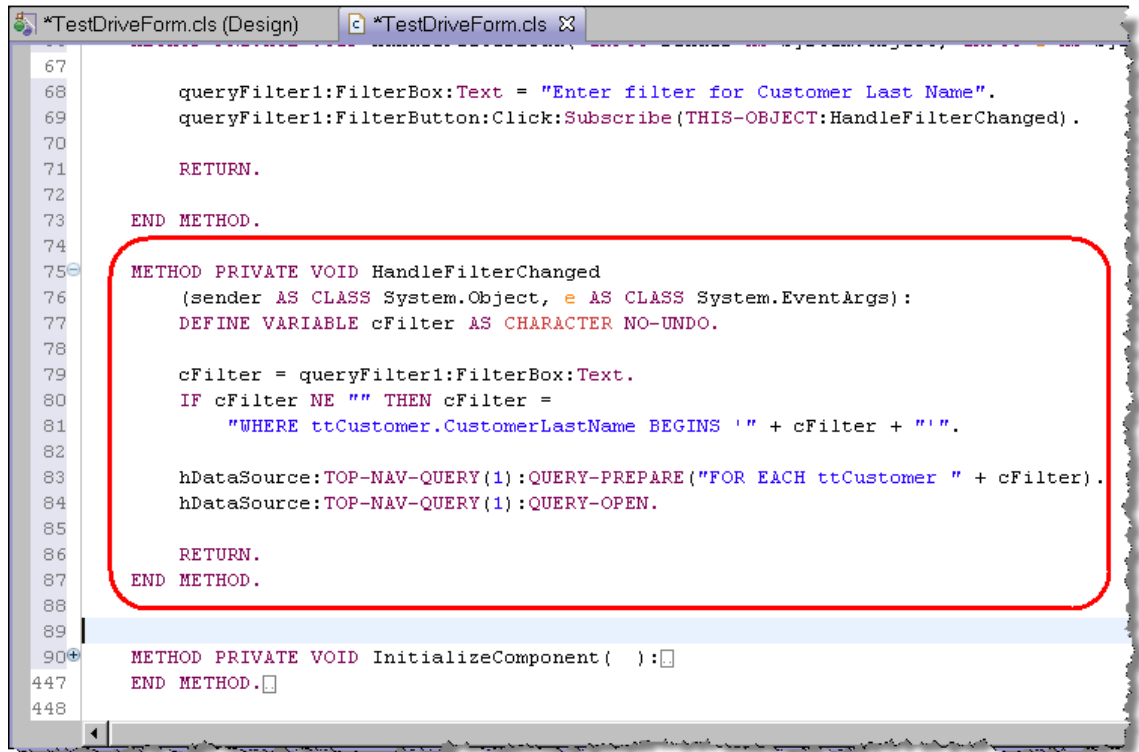
```
queryFilter1:FilterBox:Text = "Enter filter for Customer Last Name".  
queryFilter1:FilterButton:Click:Subscribe(THIS-OBJECT:HandleFilterChanged).
```

continued on next page

Lab 10 – Using the Visual Designer, continued

Working with User Controls, continued

9. Create a new method as shown below to handle the FilterButton's click event. You can copy the code from the box below.



```
67
68     queryFilter1:FilterBox:Text = "Enter filter for Customer Last Name".
69     queryFilter1:FilterButton:Click:Subscribe(THIS-OBJECT:HandleFilterChanged).
70
71     RETURN.
72
73 END METHOD.
74
75 METHOD PRIVATE VOID HandleFilterChanged
76 (sender AS CLASS System.Object, e AS CLASS System.EventArgs):
77     DEFINE VARIABLE cFilter AS CHARACTER NO-UNDO.
78
79     cFilter = queryFilter1:FilterBox:Text.
80     IF cFilter NE "" THEN cFilter =
81         "WHERE ttCustomer.CustomerLastName BEGINS '" + cFilter + "'".
82
83     hDataSource:TOP-NAV-QUERY(1):QUERY-PREPARE("FOR EACH ttCustomer " + cFilter).
84     hDataSource:TOP-NAV-QUERY(1):QUERY-OPEN.
85
86     RETURN.
87 END METHOD.
88
89
90+ METHOD PRIVATE VOID InitializeComponent( ):
447     END METHOD.
448
```

Program: TestDriveForm.cls

```
METHOD PRIVATE VOID HandleFilterChanged
(sender AS CLASS System.Object, e AS CLASS System.EventArgs):
DEFINE VARIABLE cFilter AS CHARACTER NO-UNDO.

cFilter = queryFilter1:FilterBox:Text.
IF cFilter NE "" THEN cFilter =
    "WHERE ttCustomer.CustomerLastName BEGINS '" + cFilter + "'".

hDataSource:TOP-NAV-QUERY(1):QUERY-PREPARE("FOR EACH ttCustomer " + cFilter).
hDataSource:TOP-NAV-QUERY(1):QUERY-OPEN.

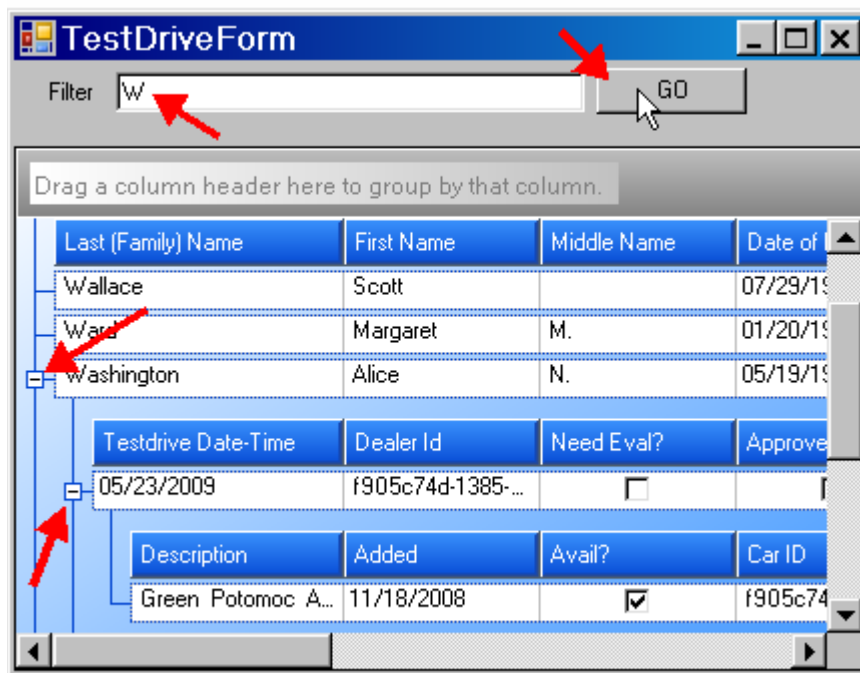
RETURN.
END METHOD.
```

continued on next page

Lab 10 – Using the Visual Designer, continued

Working with User Controls, continued

10. Save the file.
11. Select the **TestDriveForm.cls (Design Form)** tab.
12. Run the program using the Generic Run launch configuration.
13. Test the new functionality by entering **W** in the Filter box and selecting the **GO** button. Make sure the filter worked by showing only customers with a last name that starts with W. Also, select some of the nodes.



14. Close the application.
 15. Close Architect.
-