

PROGRESS EXCHANGE 2013

DISCOVER. DEVELOP. DELIVER.

Progress Developer Studio for OpenEdge

Copyright © 2013 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.

OpenEdge® is a registered trademark of Progress Software Corporation.

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

Printed in the USA

October, 2013
Version 1.0



WORKSHOP OVERVIEW

Welcome to Exchange and to this workshop on Progress Developer Studio for OpenEdge (PDS OE). The goal of this workshop is to introduce you to Progress Developer Studio for OpenEdge and to provide you with the knowledge needed to gain the most from this Integrated Development Environment (IDE).

This workshop is composed of a number of labs that show you how to use the primary features of Progress Developer Studio for OpenEdge. The labs explain the features of Progress Developer Studio for OpenEdge at length, including the application development using AppBuilder, GUI for .NET, WebSpeed, and REST. The discussions in this workshop have been designed based on what you need to know to get started and on frequently asked questions by the users of Progress Developer Studio for OpenEdge.

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

- Use Progress Developer Studio for OpenEdge as your primary tool for ABL development
- Use the OpenEdge editor to write application code
- Use AppBuilder and GUI for .NET for developing ABL desktop applications
- Use WebSpeed and REST for Web and Cloud based applications development

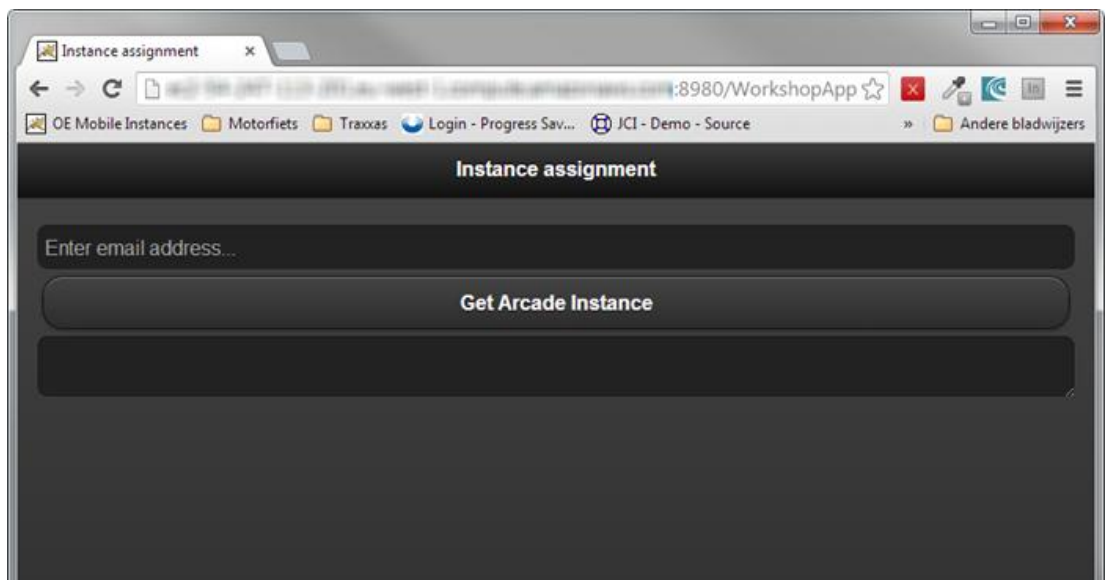
Enjoy the workshop!



CONNECTING TO YOUR MACHINE



1. Open the browser (Google Chrome, Mozilla Firefox, or Apple Safari).
2. Depending on your session, select the appropriate link and paste it in your Browser, and hit **Enter**.
 - Morning workshop, use this URL <http://23.23.210.136:8980/WorkshopApp>
 - Afternoon workshop, use this URL <http://54.225.237.144:8980/WorkshopApp>
3. Enter your email address.
4. Click **Get Arcade Instance**.

It gives you a DNS for a running Arcade instance, which you can use to connect to an RDP session.



WORKSHOP LABS

This workshop is composed of several short presentations and labs on various developer studio capabilities. Completing all of the labs is not the goal of the workshop. The goal of the workshop is to get you to the point where you can go back to your office and be ready to start using Progress Developer Studio for OpenEdge.

The labs contain Tips , that offer suggestions related to the task at hand, and Notes , that provide additional details related to the current task. Tips and Notes provide additional information on a topic, but are not mandatory to complete the labs.

Also, pay attention to the Cautions  to avoid possible problems.



CONTENTS

1	Lab 01: Configuring your workspace and customizing the project.....	1-8
1.1	Overview	1-8
1.2	Prerequisites	1-8
1.3	Starting Progress Developer Studio for OpenEdge and creating a workspace.....	1-8
1.4	Creating a Database connection	1-11
1.5	Creating an OpenEdge Project	1-16
1.6	Resetting a perspective	1-20
1.7	Adding a directory to a project.....	1-21
1.8	Setting project properties	1-22
1.9	Closing and Opening projects (Optional)	1-23
1.10	Accessing Online Help (Optional).....	1-24
2	Lab 02: Application development using Progress Developer Studio for OpenEdge – AppBuilder	2-27
2.1	Overview	2-27
2.2	Prerequisites	2-28
2.3	Opening the OpenEdge AppBuilder perspective.....	2-28
2.4	Creating a SmartWindow	2-29
2.5	Designing the SmartWindow	2-32
2.6	Working with SmartObjects	2-34
2.6.1	Creating a SmartDataObject.....	2-35
2.6.2	Creating a SmartDataBrowser	2-37
2.6.3	Add SmartFilter to filter customer records (Optional)	2-41
2.7	Designing the order entry form (Take-home)	2-47
2.8	Linking Get Customer and Order Entry buttons to the respective frames (Take-home).....	2-52
3	Lab 03: Application development using Progress Developer Studio for OpenEdge – WebSpeed	3-61
3.1	Overview	3-61
3.3	Creating a WebSpeed project	3-62
3.4	Creating the SpeedScript file.....	3-63
3.5	Configuring the WebSpeed server and associating the WebSpeed module.....	3-65
3.6	Running the SpeedScript file on a WebSpeed server	3-68
3.7	Adding an AppServer facet to the WebSpeed project.....	3-69
3.8	Working with the SpeedScript editor	3-72
4	Lab 04: Application development using Progress Developer Studio for OpenEdge – REST	4-79



4.1	Overview	4-79
4.2	Prerequisites	4-79
4.3	Adding a REST facet to the WebSpeed project	4-79
4.4	Working with the REST Editor – Mapping ABL operations with HTTP verbs	4-81
4.5	Publish and invoke the REST service	4-85
4.6	Working with other HTTP verbs (Take-Home)	4-88
4.7	Invoke the REST service for remaining HTTP verbs (Take-home)	4-90
5	Lab 05: Application development using Progress Developer Studio for OpenEdge – GUI for .NET	5-94
5.1	Overview	5-94
5.2	Prerequisite	5-94
5.3	Creating Custom Project	5-95
5.4	Assigning a database connection to a project	5-101
5.5	Creating and designing a form	5-102
5.6	Creating Inherited controls (Take-home)	5-109
5.7	Adding controls to Toolbox	5-111
5.8	Creating User control (Take-home)	5-113
5.9	Designing the SalesReport form to display outlets data	5-115
5.10	Working with the source editor - Adding methods and events	5-120
5.11	Working with OpenEdge Debugger (Optional)	5-128
5.12	Working with UltraChart (Take-home)	5-131
6	Lab 06: Embedding an ABL window into the MDI form (Take-home)	6-134
6.1	Overview	6-134
6.2	Prerequisites	6-134
6.3	Creating an MDI form	6-134
6.4	Embedding the ABL windows into an MDI form	6-138
7	Configuring Apache WebServer (not required for the Workshop)	7-142



1 LAB 01: Configuring your workspace and customizing the project

1.1 Overview

This lab covers setting up the workshop environment and creating database connections and starting database servers. It includes the following:

- Creating a workspace and a project
 - Configuring and customizing your project
 - Setting up the project properties and PROPATH
 - Closing and re-opening the project
-

1.2 Prerequisites

The virtual machine that you will use has been set up for PDS OE with Workgroup DB and OE Ultra Controls .NET installed in the C:\Progress\OpenEdge directory.

The database required for this workshop, workshop.db is created with its schema and is placed in the C:\OpenEdge\WRK\ExchangeDB\workshop.db directory.

The files used for this workshop are placed in the C:\OpenEdge\WRK\PDSOEWorkshopFiles directory.

1.3 Starting Progress Developer Studio for OpenEdge and creating a workspace

Before you create a project, you must start Progress Developer Studio for OpenEdge.

1. Select **Start->All Programs->Progress->OpenEdge 11.3 ->Developer Studio - Clean**.

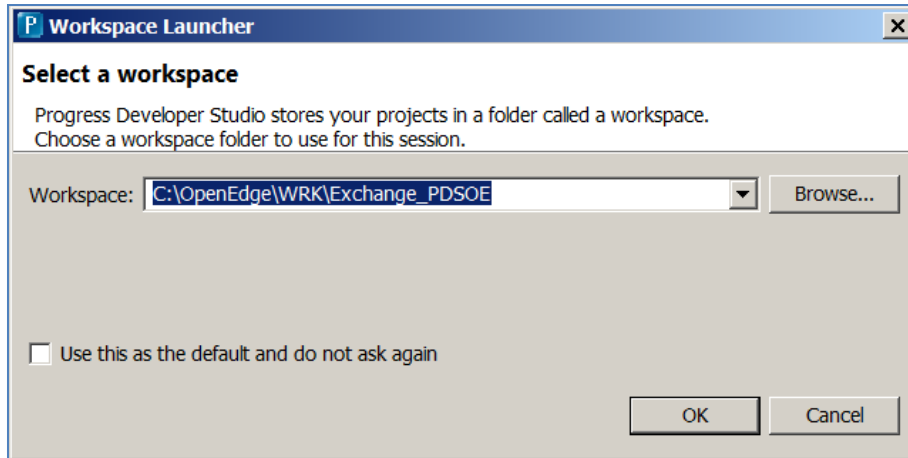
The **Workspace Launcher** dialog box appears.

2. Set the **Workspace** to C:\OpenEdge\WRK\Exchange_PDSOE. A new directory **Exchange_PDSOE** is created that sets up the workspace environment.



Tip: Workspaces can be physically located anywhere on your system or on a network drive. However, a workspace can be used by only one developer at a time even if it is on the network drive. So, we suggest that you set up your workspaces on your local computer.



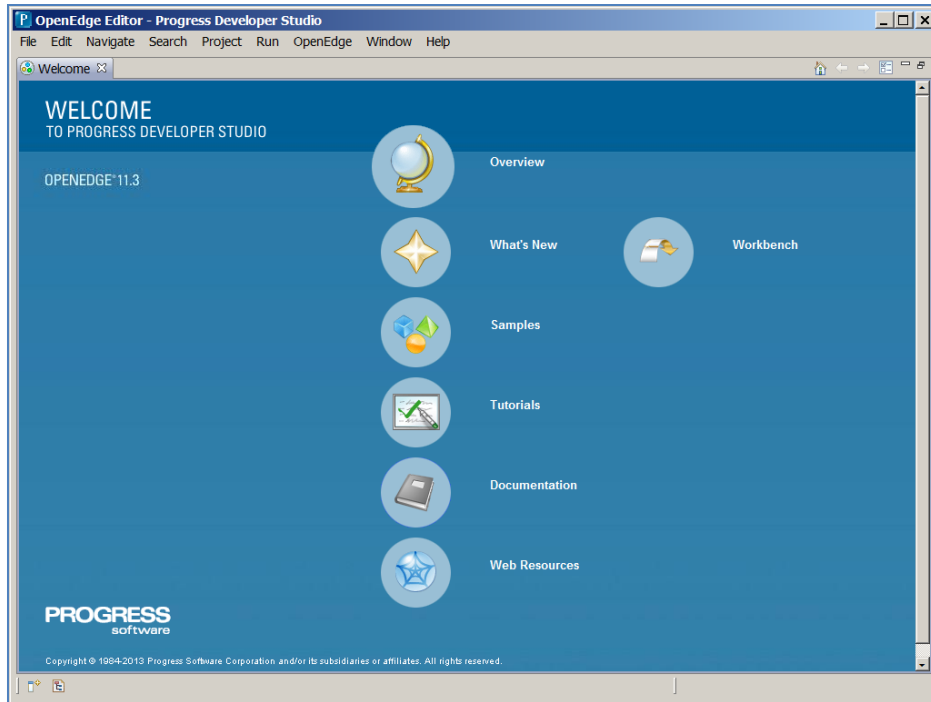



Tip: As shown in the above image, you can set up the currently configured workspace as the default workspace by selecting the **Use this as the default and do not ask again** check box. It stops developer studio from prompting for a workspace every time you open the tool and automatically opens the workspace that is set as default. You can use workspaces other than default at any time by selecting **File** → **Switch Workspace**. If you frequently work with multiple workspaces, you can leave this check box cleared so that every time you start developer studio, you have the flexibility of starting the required workspace. This option, along with many other workspace options, can be modified using the Workspace preferences, which will be covered later in the labs.

3. Click **OK**. As Progress Developer Studio for OpenEdge opens, the splash screen shows the plug-in information as they are being loaded.

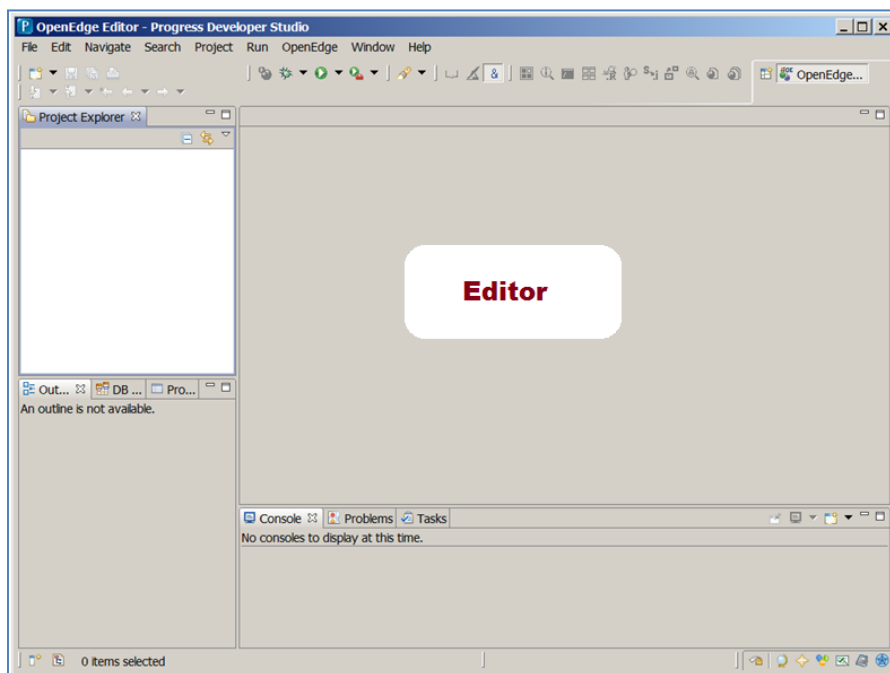
The **Welcome** page appears. It provides links to the resources of Progress Developer Studio for OpenEdge. Each link opens up related links to that particular resource.





 **Tip:** **Welcome** page gives links to all product related help. Samples and Tutorials lists items group by topic. If you select a topic and click **Open**, it takes you to the Progress communities' link of that topic. Documentation has link to the product documentation page, where latest release documents can be found.

4. Select **Workbench** . The **OpenEdge Editor** perspective opens.





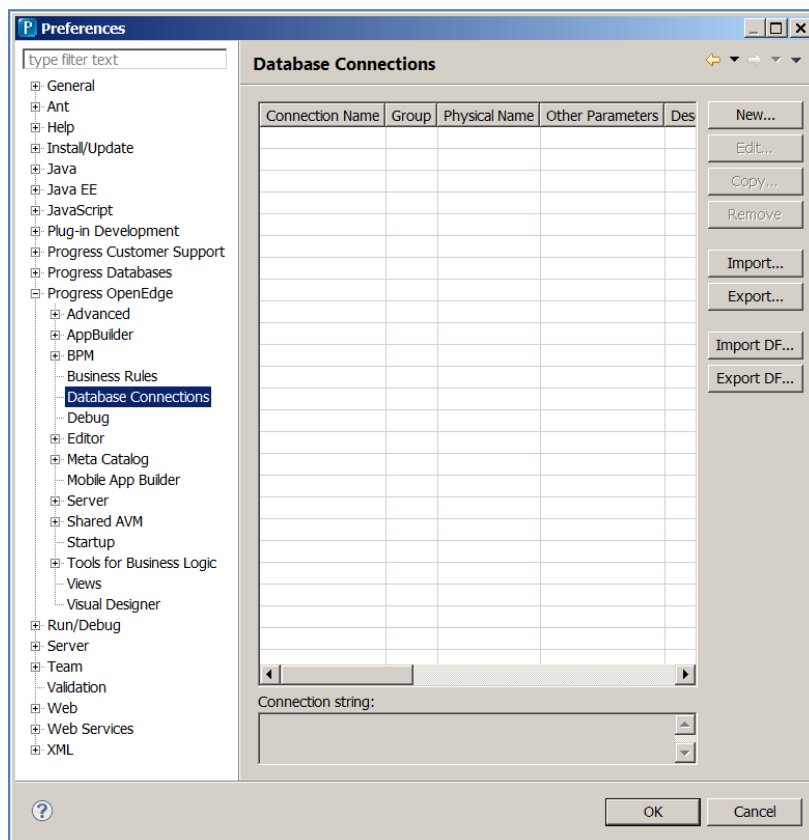
Tip: If you want to move an existing Workspace to a different physical directory, copy the workspace and move it to the new location, start PDS OE, and enter the new path for the workspace in the **Workspace Launcher** dialog box. You also need to change any physical paths that you have set such as the PROPAT. These changes can be minimized by using Configuration Variables (see the online help for information on Configuration Variables). You also need to change the working directory, go to the OpenEdge project properties -> Progress OpenEdge and select the **Browse** button.

1.4 Creating a Database connection

You can define database connection profiles for the workspace and make connections from your projects in Progress Developer Studio for OpenEdge to OpenEdge databases.

The **Database Connections** wizard allows adding both ABL and SQL database connections. This wizard creates a database connection profile. The next section of the lab shows how projects can connect to databases by selecting database connection profiles defined for the workspace.

1. From the Workbench menu bar, select **Window**→**Preferences**. The **Preferences** dialog box appears.
2. Expand the **Progress OpenEdge** node and select the **Database Connections** node. The **Database Connections Preference** page appears.



3. Select **New...**. The **Add Connection Profile** dialog box appears. This first page of the wizard allows you to enter the connection information for an ABL connection to the OpenEdge RDBMS.

Add Connection Profile

Add OpenEdge Database Connection

Enter a unique name for the database connection

Connection name:

Physical name: **Browse...**

Optional

Description:

Logical name:

Host name: Service/Port:

User ID: Password:

Aliases: Group:

Other parameters:

Test Connection

< Back **Next >** **Finish** **Cancel**

4. Specify **Exchange_db** in **Connection name**.
5. Click **Browse...** next to **Physical name**. Browse to the **C:\OpenEdge\WRK\ExchangeDB** folder and select the **workshop** file. Click **Save**.

The **Physical name** field should now display the **C:\OpenEdge\WRK\ExchangeDB\workshop.db** path.

6. Specify the other values as follows:

Host name: localhost
Service/Port: 6210
User ID: pdsoe
Password: pdsoe



Note: The Service/Port field accepts either a service number or a port number. In case you use a service name, make sure that it is defined in the services file. For Windows, the services file is located in the C:\WINDOWS\system32\drivers\etc\services directory.



Add Connection Profile

Add OpenEdge Database Connection

If a User ID was specified, enter the valid password for the user ID.

Connection name: Exchange_db

Physical name: C:\OpenEdge\WRK\ExchangeDB\workshop.db Browse...

Optional

Description:

Logical name:

Host name: localhost Service/Port: 6210

User ID: pdsoe Password: ●●●●

Aliases: Group:

Other parameters:

Test Connection

? < Back Next > Finish Cancel

- Click **Next**. The **Define a SQL connection** dialog box opens.

Add Connection Profile

Define a SQL connection

Define SQL Connection

Add new SQL connection Use existing SQL connection

Existing SQL connections

Name	URL	Driver



Caution: If you do not need a SQL connection, clear the **Define SQL Connection** check box. However, we recommend that you configure a SQL connection since it is used by a number of the tools in PDS OE and they might not work properly without a SQL connection.

- Confirm that the **Add new SQL connection** option button is selected and click **Next**. The **Add SQL Connection Profile** dialog box opens; here you define the SQL connection.



You will notice that all the information that you entered for the ABL connection is automatically filled in for the SQL connection.

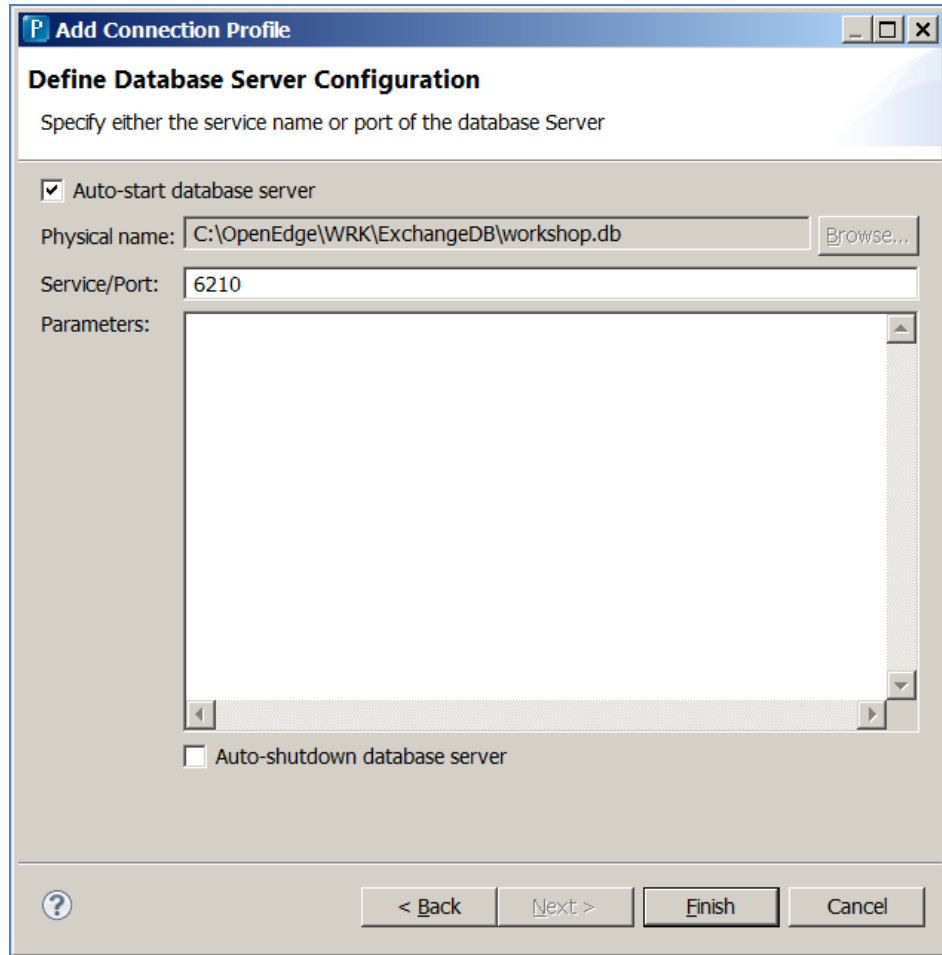


Note: When working with SQL, the user who creates the database has access to the database. By default, the database is created with windows logon ID. If you had used a different user ID to create the database then that user ID should be entered in the User ID field. here, instead of the windows logon ID. For example, the SQL connection to workshop.db needs to use the User ID of the user that created the database (in this case, PDS OE).

When using PDS OE, you may choose to not use a password for the database. It is acceptable depending on your requirement specially if are using PDS OE as only a development tool. For more information on SQL database security, including the creation and usage of user IDs and passwords, see the OpenEdge SQL documentation.

9. Click **Next**. The **Define Database Server Configuration** dialog box opens. It allows a database broker to be started for the connection profile. Continue with the defaults.

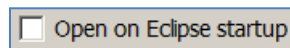




Note: The options on this page are to configure auto-start and auto-shutdown of a database server. Ignore these options in cases when a database broker is already started for a database.

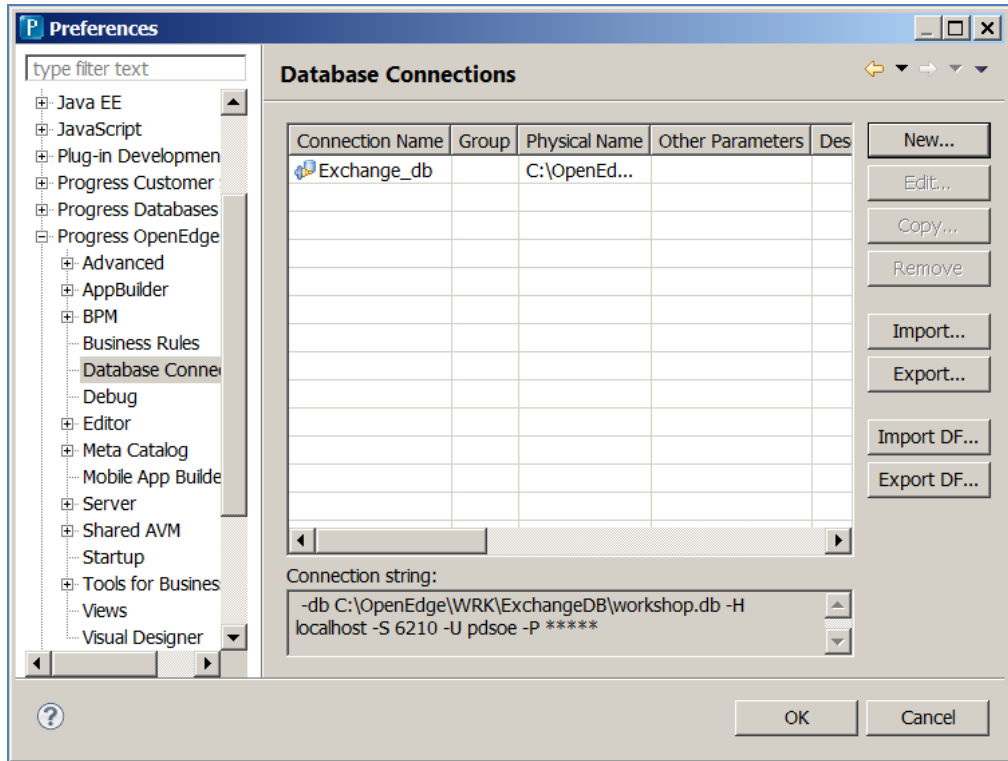


Tip: If you do not want the database broker to be started using this option, clear the **Open on Eclipse startup** check box in the **Add SQL Connection Profile** dialog box, so that the broker does not run when PDS OE initially starts up.



10. Select the **Auto-shutdown database server** check box.
11. Click **Finish**. The **Database Connections** preference page appears again displaying the newly created database connection profile in the table grid.



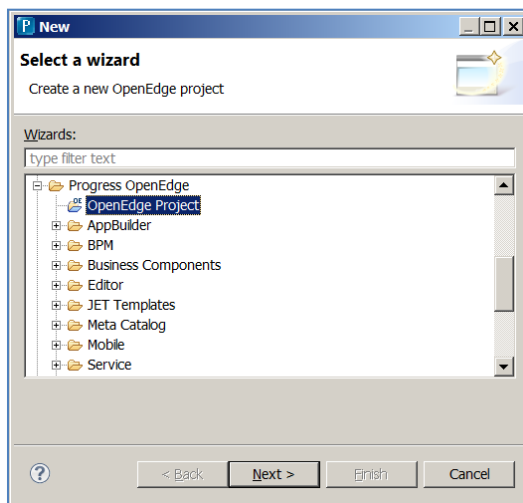


12. Click **OK** to close the **Preferences** dialog box.

1.5 Creating an OpenEdge Project

This section shows you how to create a new project.

1. From the Workbench menu, select **File**→**New**→**Other**. The **New** dialog box appears.
2. Double-click the **Progress OpenEdge** node and select **OpenEdge Project**.

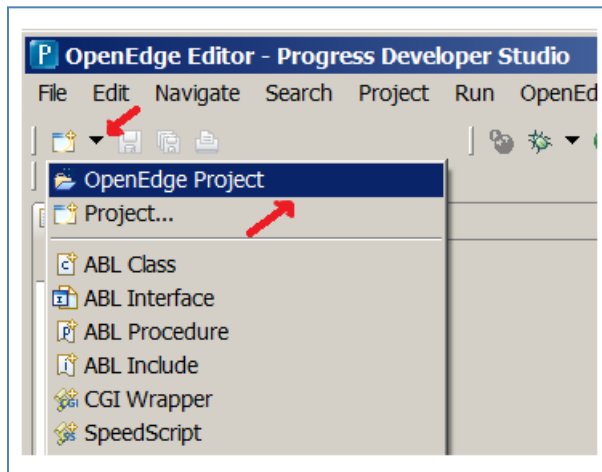




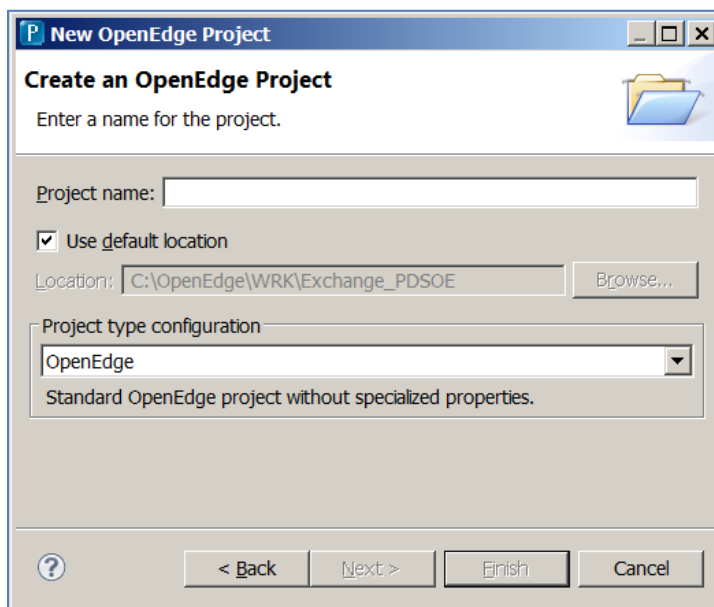
Tip: A common feature in the PDS OE user interface is a fill-in at the top of a selection that allows you to filter the list to be displayed. For example in the above screen, you can enter text in the Wizards empty text box. If you enter O, it displays the list of wizards that start with O. This filter helps to locate the selection that you are searching for.



Tip: You can directly open the **OpenEdge Project** dialog box by selecting **File**→**New**→**OpenEdge Project** or selecting the drop-down arrow button next to **New** and selecting **OpenEdge Project** as shown below:



3. Click **Next**. The **New OpenEdge Project** dialog box appears.

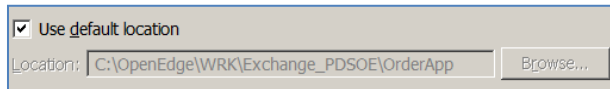


4. Enter **OrderApp** in **Project name**.

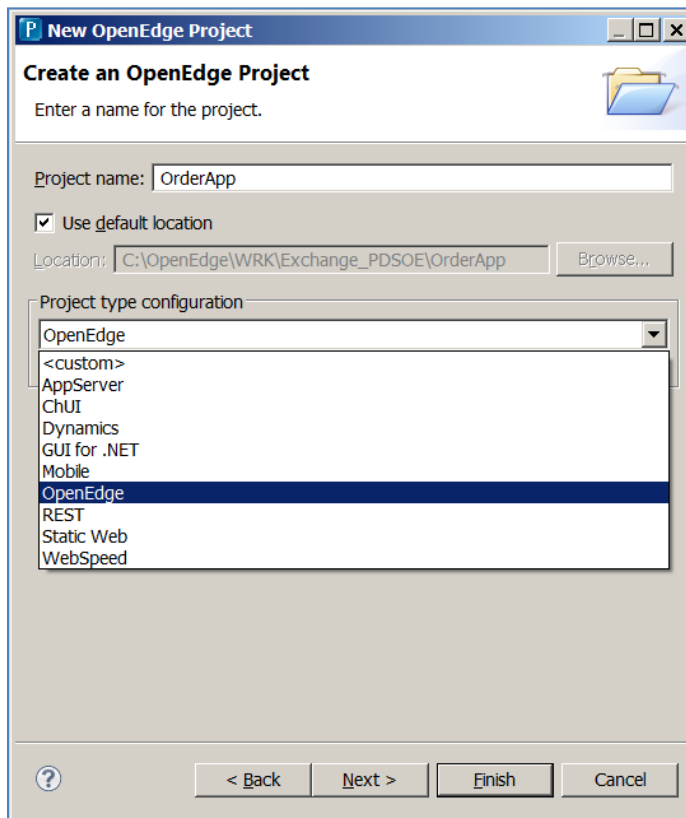




Tip: The directory path for the new project is shown in the **Location**. This is the location where the project and its associated files are physically stored. By default, the framework creates a subdirectory with the same name as the project in the workspace directory. You can choose to create the project in a different directory by clearing the **Use default location** check box and editing the **Location**.



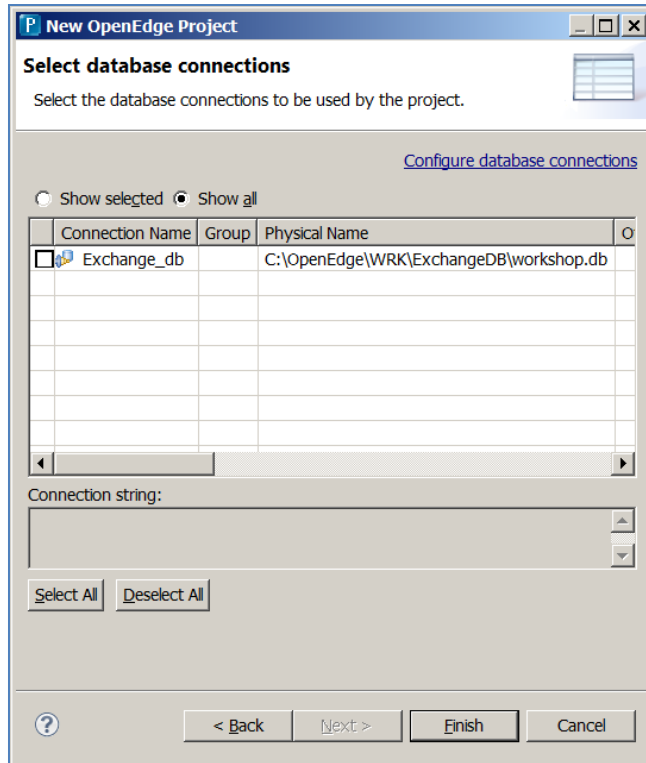
5. Open Edge is selected as the **Project type configuration**. Continue with the defaults.



Tip: Depending on the application to be developed, you should select the project type configuration. It adds the related facets, and the next pages in this wizard will depend on the configurations needed for the selected project type.

6. Click **Next** until the **Select database connections** dialog box appears.





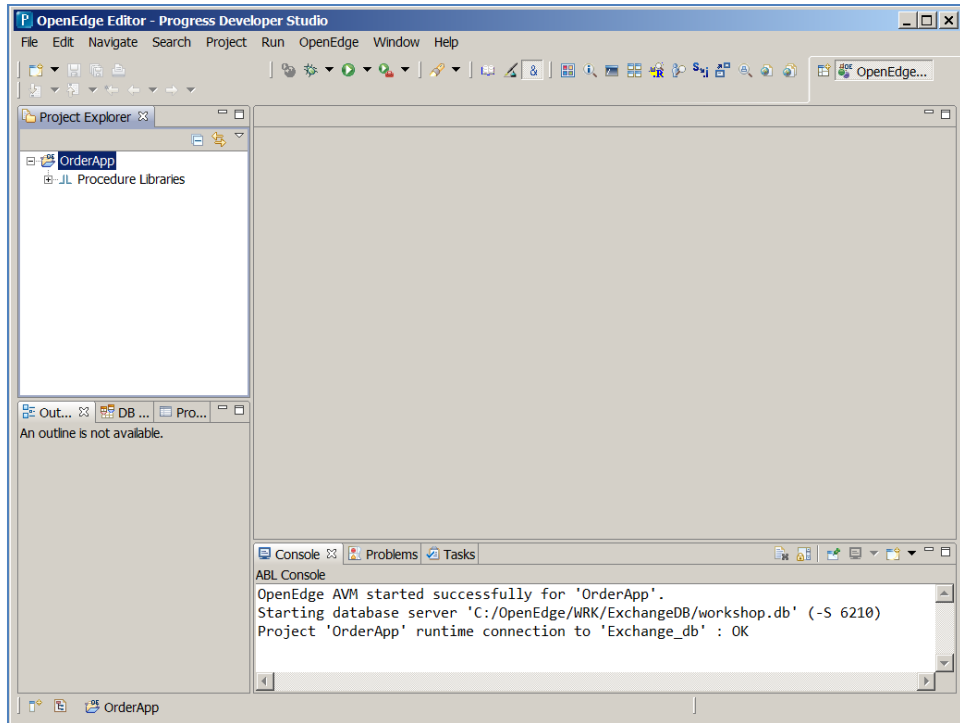
Caution: Add only the database connections that are needed. If you have a large number of projects that are always active and they connect to a number of databases then the time to start these connections will increase the time to start PDS OE. Also consider that there are limitations to the number of allowed database connections depending on the OpenEdge products that are installed and configured.

7. Select the **Exchange_db** check box and click **Finish**. The project is created. The **Powered by Progress** splash screen appears showing that a Progress session is started for the project.



Once you go back to the editor, you will see the project listed in the **Project Explorer** view:






The console view shows a message that the OrderApp project successfully connected to the AVM.



Note: The Workbench contains a number of views. On the left are the **Project Explorer** and the **Outline** views. The tab for the **DB Structure** view can also be seen. The bottom center shows views that can be selected by the folder tabs, for **Console**, **Problems**, and **Tasks**. The middle center is empty and is where the editor area is located.

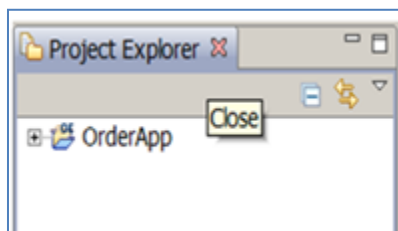


Tip: It can be useful at times to clear the messages in the **Console** view to focus just on the latest messages. Click  on the **Console** view to clear all the messages.

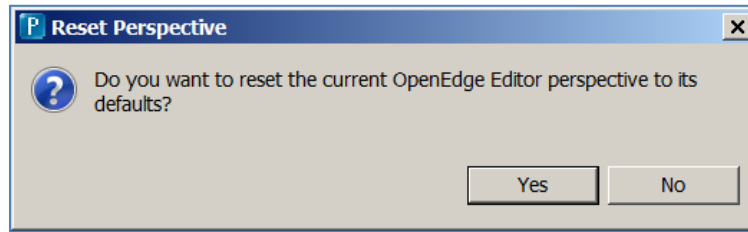
1.6 Resetting a perspective

Sometimes, the views get closed or moved around the workspace. A fast way to put a perspective back to its default layout is to reset the perspective.

1. Each view has its own **Close** button on its label tab. When you move your cursor over it, the **X** changes color to show it has focus. The pop-up text shows the function of X as seen below. Close the **Project Explorer** view and the **Outline** view.




- From the Workbench menu, select **Window**→**Reset Perspective**. The following message is displayed:

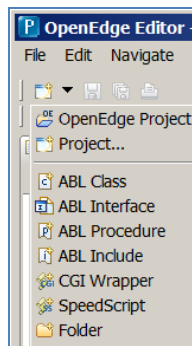


- Click **Yes**. The selected perspective is reset to the default configuration. In this case, the **Project Explorer** view and **Outline** view are visible again on the left side of the page.

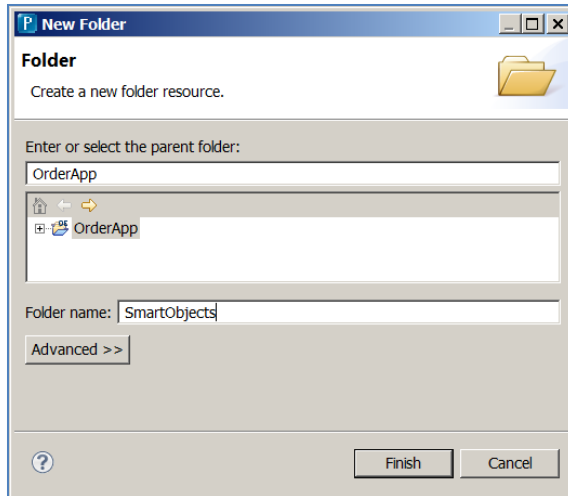
1.7 Adding a directory to a project

A common task is to add a directory to a project. Directories can be used to organize resources within a project.

- In the **OpenEdge Editor** perspective, from the workbench menu bar, select the drop-down list next to **New** .



- Select **Folder** from the list. The **New Folder** dialog box is displayed.
- Select **OrderApp** as the parent folder and enter **SmartObjects** in **Folder name**.

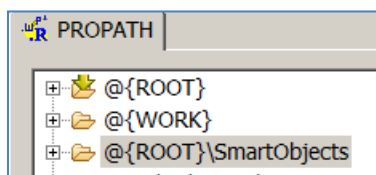


4. Click **Finish**. The SmartObjects folder is created in the OrderApp project directory.

1.8 Setting project properties

You can view the properties of a project and modify the PROPATH so that files from other folders can access files inside this folder.

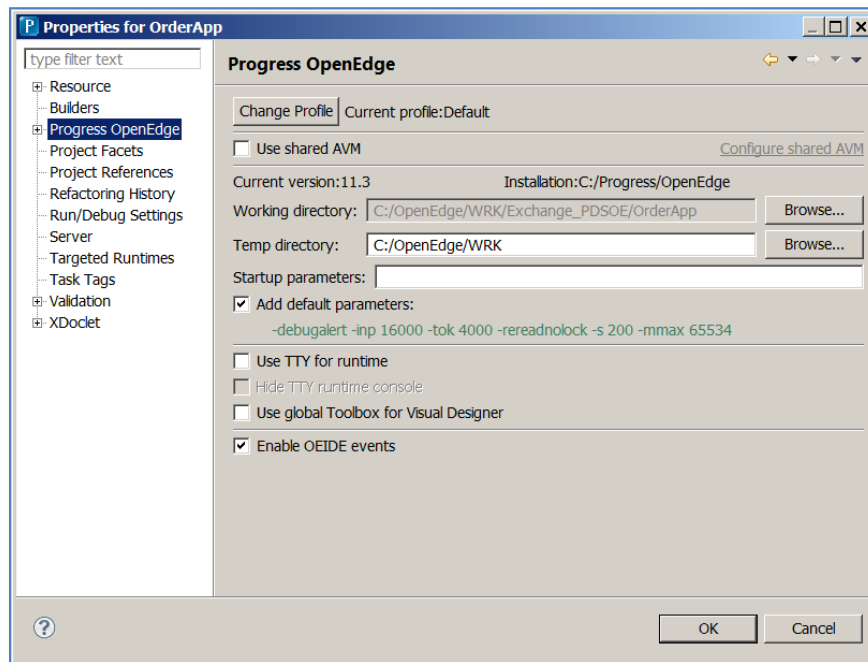
1. In the **Project Explorer** view, select the **OrderApp** project.
2. From the Workbench menu, select **Project**→**Properties**. The **Properties for OrderApp** dialog box appears. Since project properties are specific to a project, the project name appears in the title of the dialog box.
3. Double-click **Progress OpenEdge** and select **PROPATH**, click **Add Workspace Directory....** The **Select Propath Directory** dialog box is displayed.
4. Select **SmartObjects** and click **OK**. By adding SmartObjects folder to PROPATH, all other files in the project can directly access the files in SmartObjects folder without providing fully qualified name or full path. Whenever any operation is being done, AVM first searches the PROPATH. (This is similar to JAVA CLASSPATH).
5. Use **Move Up** and **Move Down** to locate the newly added **@{ROOT}\SmartObjects** entry in **PROPATH** after the **@{WORK}** entry.





6. Double-click **Progress OpenEdge** on the left section of the page again 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 the AVM startup parameters.





 **Note:** The 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 values for the parameters can be changed or removed by clearing the **Add default parameters** check box.

 **Caution:** The `-ide` startup parameter lets the AVM know that it is being used in conjunction with PDS OE. This parameter must be included for PDS OE to function properly.

 **Caution:** While you could connect to a database with startup parameters on this page, we strongly discourage it. It is difficult to diagnose connection problems if they occur, as opposed to using the Database Connections properties which are designed for defining connections to databases. Details on creating database connections are covered in the *Creating a Database Connection* section.

7. Click **OK** to apply the changes. The **Project Properties** dialog box closes.

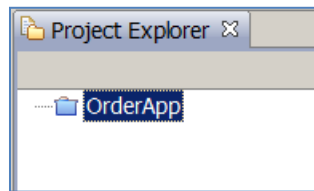
1.9 Closing and Opening projects (Optional)

Workspaces use a varied amount 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 a project include freeing up resources used by the project such as database connections. Closing unused projects will also improve performance of some tasks. For example, searches will complete more quickly if you close projects that you do not want to search.

1. Select the **OrderApp** project in the **Project Explorer** view.



2. Right-click the **OrderApp** project and select **Close Project** from the context menu. The project and all its nodes are closed. If there were database connections for this project, they would also be closed.

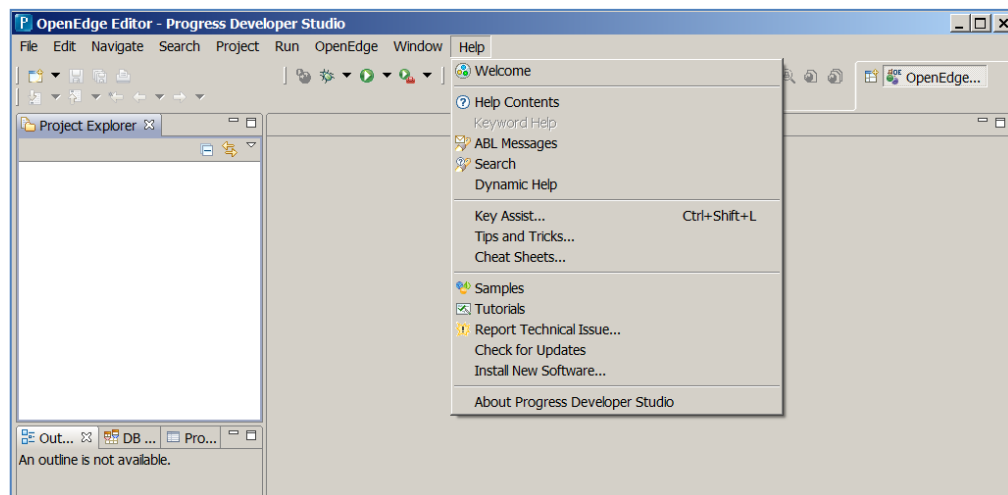


The **OrderApp** project still appears in the view but is grayed out and has a different icon as shown in the screen shot above. The nodes under a closed project cannot be expanded or viewed when the project is closed.

3. Right-click the **OrderApp** project and select **Open Project** from the context menu. The project is opened again.

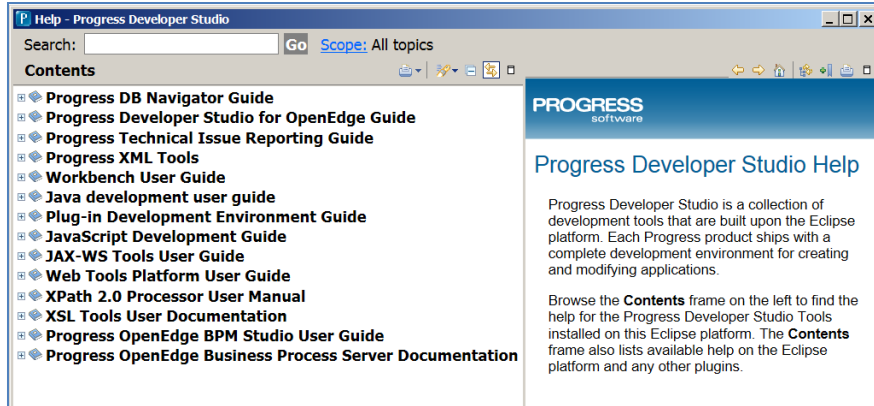
1.10 Accessing Online Help (Optional)

Progress Developer Studio for OpenEdge comes with an extensive online help. It includes help on Eclipse Framework (Workbench User Guide) and Progress Developer Studio for OpenEdge specific help. If you add additional functionality through the plug-ins, then additional topics appear under the help contents. You can specify the topics you want help on. It speeds up the searching and reduces the number of unwanted matches. From the Workbench menu, select **Help->Help Contents** to open the Progress Developer Studio for OpenEdge Help page.



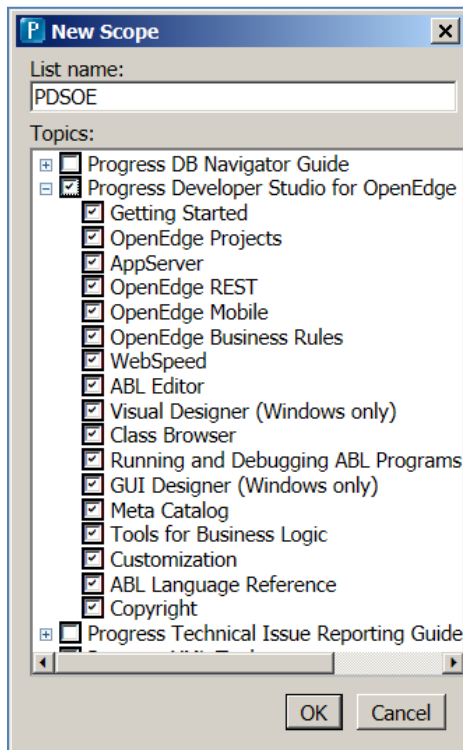
The Progress Developer Studio for OpenEdge help page opens:





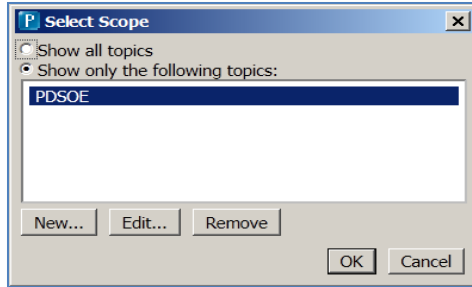
You can set up the scope of the search for the helps that you would frequently need to search for.

1. Select **Scope**. The **Select Scope** dialog box opens.
2. Click **New**. The **New Scope** dialog box opens.
3. Set the **List name** to PDSOE.
4. Expand the **Progress Developer Studio for OpenEdge Guide** node. Select the check boxes that you would frequently search for. You can select multiple nodes to be included in the search. You can also select the **Progress Developer Studio for OpenEdge Guide** check box to search across the PDS OE documentation.



5. Click **OK**. It takes you back to the **Select Scope** dialog box.





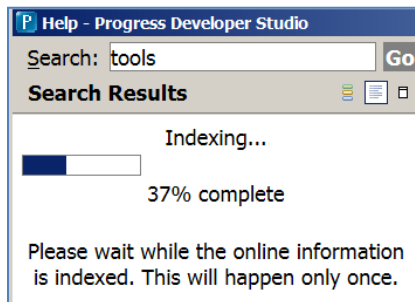
6. Confirm that **Show only the following topics** check box is selected and that **PDSOE** is highlighted, and click **OK**. The help window now shows the scope set to **Progress Developer Studio for OpenEdge Guide**.

Scope: PDSOE



Note: The search scope is retained, so the next time you start this workspace and open help, the search scope is still set to PDS OE.

7. Enter a keyword such as *tools* in **Search**, and click **GO**. The first time you access help, indexing is performed.



After the indexing is complete, the search results are displayed. The next time, you search a keyword, it completes much faster.

8. Close the **Help** page.

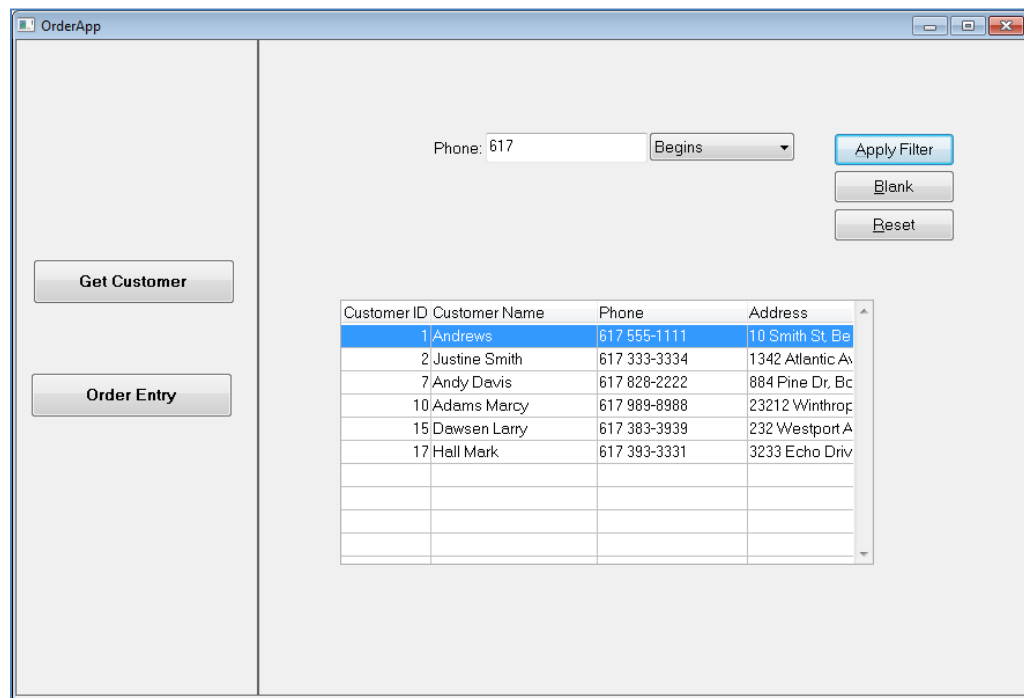


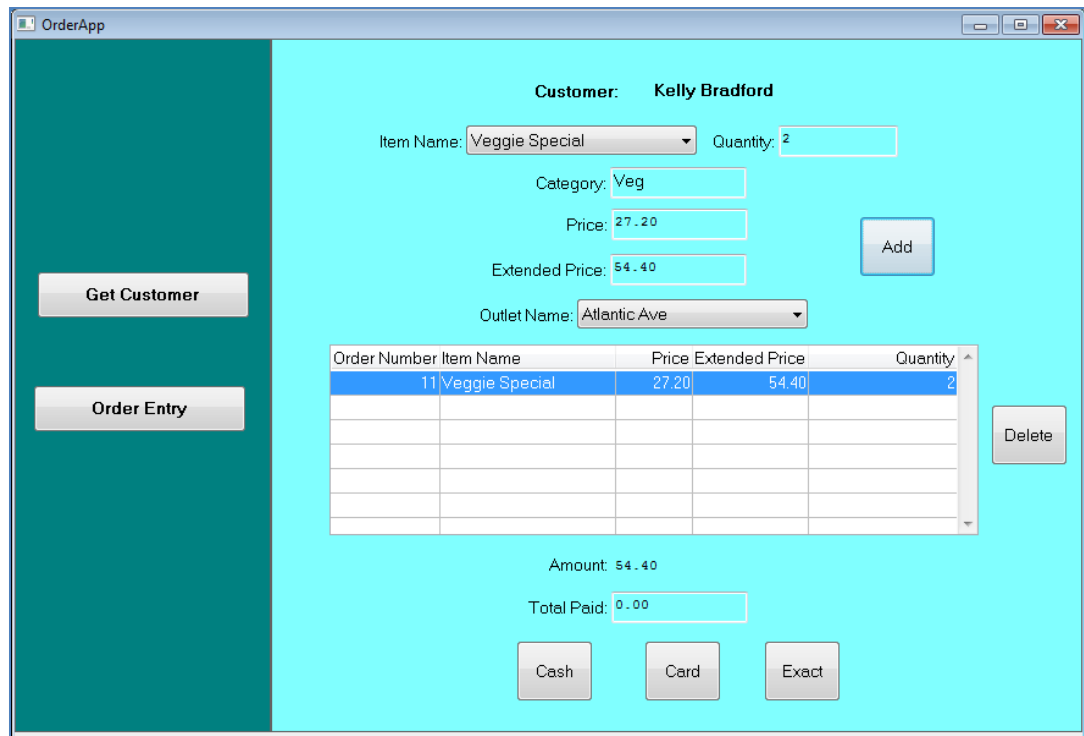
2 LAB 02: Application development using Progress Developer Studio for OpenEdge – AppBuilder

2.1 Overview

The sections of this lab show you how to design, develop, and run an ABL application. You will develop an Order Entry application using AppBuilder.

Here is how your application will finally look:





2.2 Prerequisites

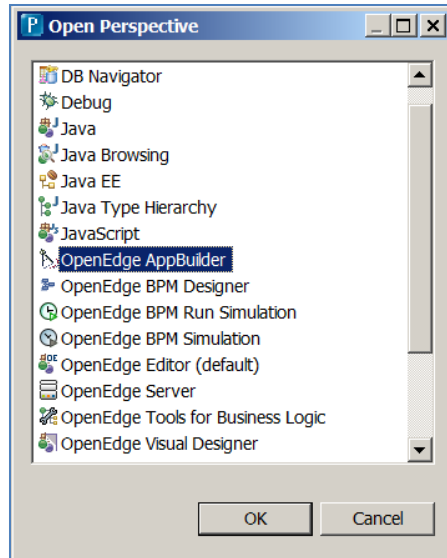
Complete **Lab 01: Configuring your workspace and customizing the Project** prior to working on this lab.

2.3 Opening the OpenEdge AppBuilder perspective

Open the AppBuilder perspective in Progress Developer Studio for OpenEdge.

1. From workbench menu, select **Window**→**Open Perspective**→**Other...**. The **Open Perspective** dialog box appears and displays a list of all available perspectives.





2. Select **OpenEdge AppBundle** and click **OK**.
3. Observe that the **AppBundle** perspective is opened. Two new views; **ABL Messages** and **ABL Cue Cards** are displayed. We will discuss these in the coming sections.



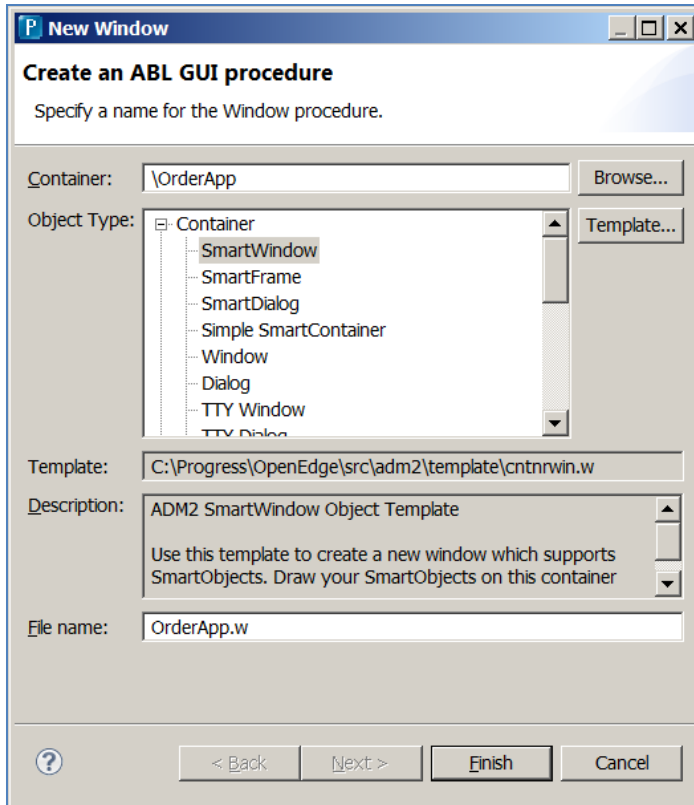
Note: Working in a related perspective has many advantages for the developers. **Project Explorer** context menu displays the related file options in the list for easy selection. Related views for the development work are provided. Views are arranged in such a way that the right editor is provided as required.

2.4 Creating a SmartWindow

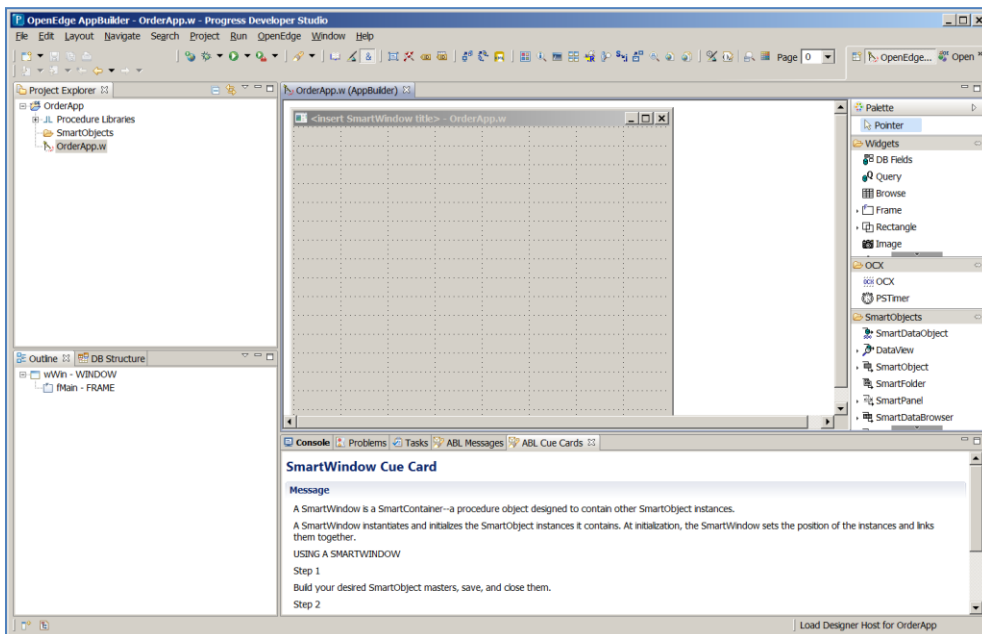
A SmartWindow is a blank container where you can add your SmartObjects and establish the appropriate SmartLinks to connect them. In this section, we will create a SmartWindow and edit its properties so that it could contain your SmartObject instances.


1. Select the **OrderApp** project from the **Project Explorer** view. Right-click and select **New**→**ABL UI Design** from the context menu. The **New Window** dialog box appears.
2. Select **Container**→**SmartWindow** in **Object Type**. Enter **OrderApp.w** in **File name**.





3. Click **Finish** to create a SmartWindow as shown in the image below:




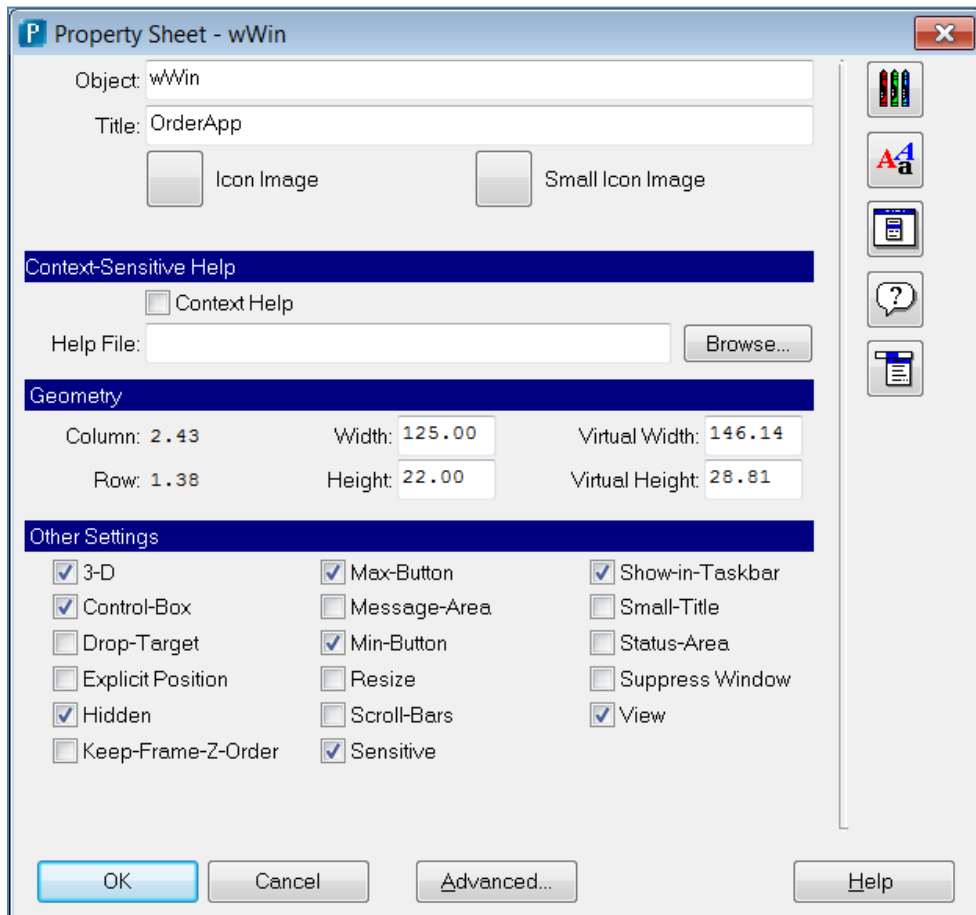
 **Note:** The **ABL Cue Card** view displays help on a specific SmartObject type. When you create a new SmartObject master, the Cue Card associated with the SmartObject type is also opened. The Cue Card provides a basic definition of the SmartObject type, along with the information on creating and using the object.




This view lists the details of all the active Cue Cards in your current AppBuilder session. The ABL Cue Cards is the default view and is available only with the AppBuilder perspective.

To retrieve the ABL Cue Cards view that you close, choose **Help>ABL Cue Cards** from the main menu.

4. Select **Property sheet** option  from **Toolbar**.
5. Enter **OrderApp** in **Title**.
6. Change the values for **Width** and **Height** properties to 125 and 22 respectively. Confirm by opening the **Properties Sheet** window again.



 **Note:** The **Property Sheet** lets you customize the various properties. If you have many frames and controls designed in a window, it is difficult to select an item in a window. The **Outline** view shows an outline of all the controls.

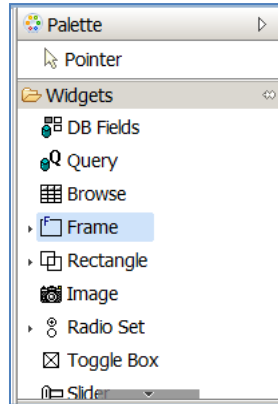
7. Click **OK** to close the **Properties Sheet** dialog box.
8. Click **File→Save** to save the changes.



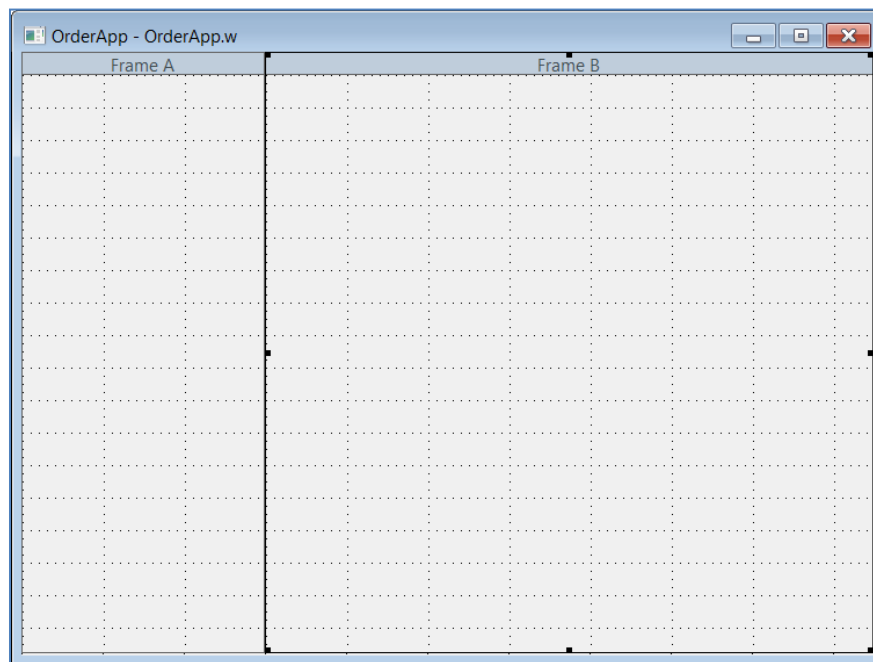
2.5 Designing the SmartWindow

This section shows you how to design your user interface.

1. From the **Widgets** on the right section, select **Frame**.




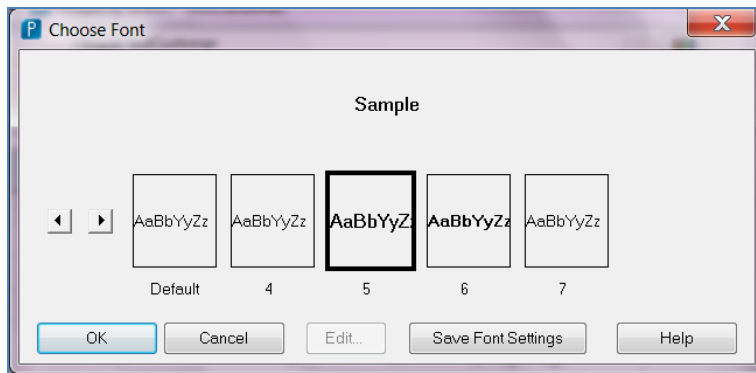
2. Click anywhere on the **OrderApp** window.
3. Repeat Step 1 to add another frame to **OrderApp** window.
4. Click **Pointer** in **Palette**.
5. Drag frames as shown in image below. You have now divided the window into two different frames; you will now design each frame.




6. Double-click the **Frame A** area. The **Property Sheet-FRAME-A** dialog box appears.
7. Clear the **Title Bar** check box in **Other Settings**.



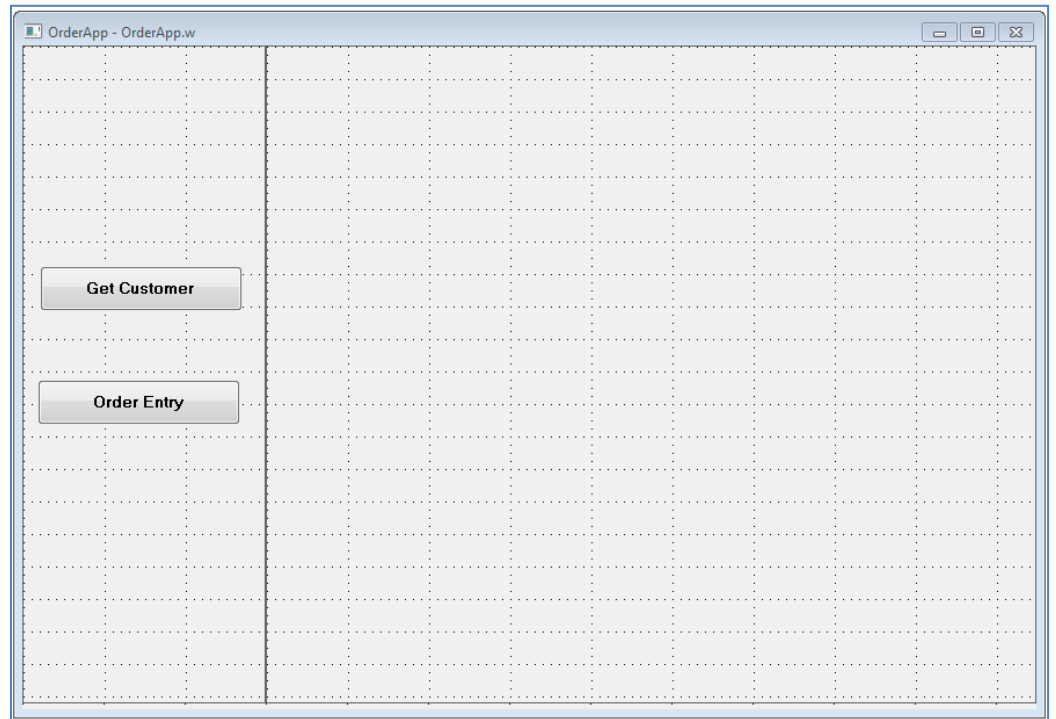
8. Specify **frameMenu** in **Object**. Click **OK** to close the **Property Sheet** dialog box.
9. Repeat for Frame B. Specify **frameCustomer** in **Object**.
10. Click **File**→**Save** to save the changes.
11. Select **Button** in the **Widgets** category from **Palette**.
12. Click **frameMenu** area. Button 1 gets added to Frame A.
13. Double click **Button 1** to open the **Property Sheet** dialog box.
14. Specify 25.0 in **Width** and 1.50 in **Height**.
15. Select  to change the font of Button 1. The **Choose Font** dialog box appears.



16. Select  and select the font with bold letters. Font choices may vary in different systems. Select the choice with bold letters.
17. Click **OK** to close the **Choose Font** dialog box.
18. Click **OK** to close the **Property Sheet** dialog box for **Button 1**.
19. Right click **Button 1** and select **Duplicate** from the context menu.
20. Arrange the buttons one after the other in the **frameMenu** area.
21. Double-click each button to open the respective **Property Sheet** dialog box and modify the values as provided in table below. Click **OK**.

	Object	Label
Button 1	butCustomer	Get Customer
Button 2	butOrdEntry	Order Entry

Your screen will look like this:



22. Click **File**→**Save** to save the changes. You have completed designing the frameMenu and it appears on the left side of the main frame.

2.6 Working with SmartObjects

This section shows you how to create SmartObjects. SmartObjects are external procedures that encapsulate standard UI and functionality. The set of SmartObjects provided with the Application Development Model (ADM) are useful for typical database applications. For example, most database applications view data, so the ADM provides a SmartViewer object. The main purpose of a SmartViewer is to display data.

Each of these SmartObjects represents a part of a component of a typical database application.

Each SmartObject knows how to interact with other SmartObjects. For example, the SmartDataObject and SmartBrowser work together in predefined ways. The SmartDataObject tells the browser what records to display in the browser. In the SmartDataObject, we define the query to be processed and the linked SmartBrowser will display the resultant data.

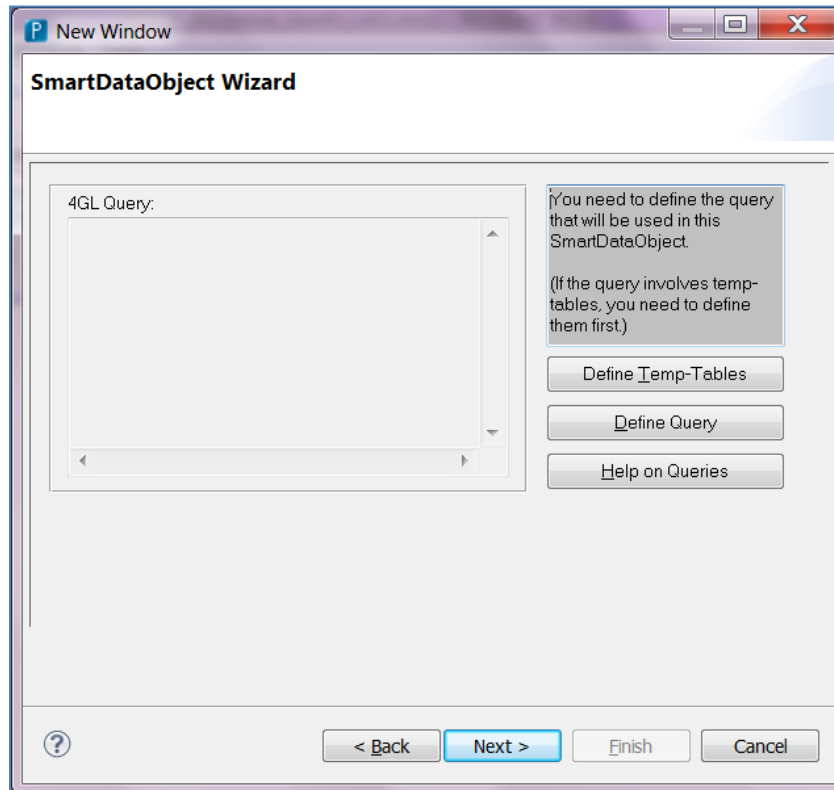


Note: SmartObjects need a database connection. If there is no database connection assigned to the current project, you will be asked to connect to a database.



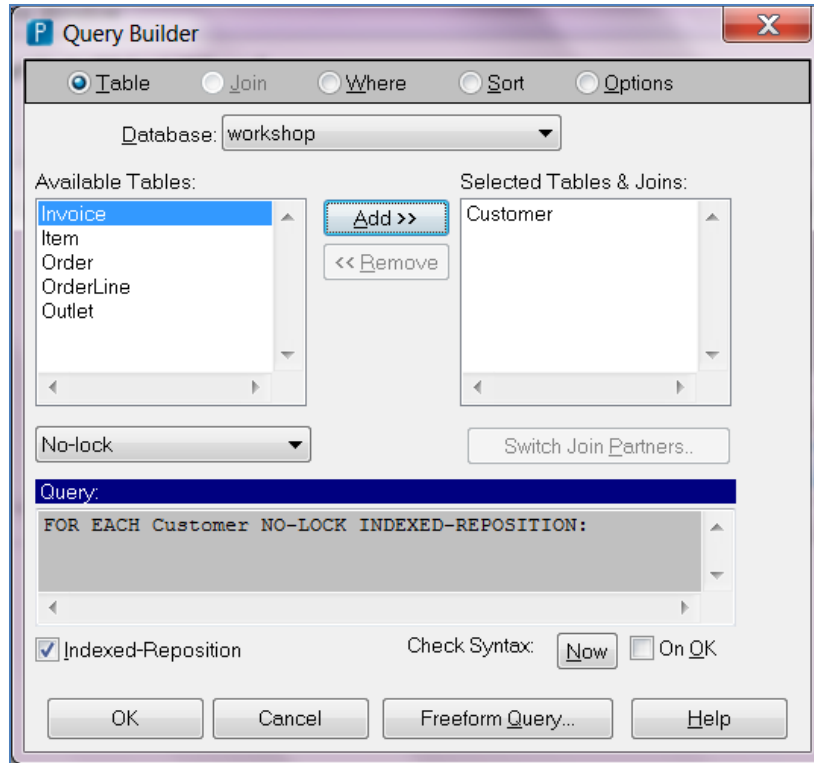
2.6.1 Creating a SmartDataObject

1. Select the **SmartObjects** folder from the **Project Explorer** view. Right-click and select **New→ABL UI Design**.
2. Select **SmartDataObject** in **Object type**. Enter **dGetCust.w** in **File name**.
3. Click **Next** until you reach the following screen where you have to define a query for the customer table.



4. Click **Define Query**. The **Query Builder** dialog box opens and it lists the tables from the database that you are connected to.
5. Select **Customer** from the **Available Tables** section and click **Add**. The query is defined for the customer table and shown in the **Query** section.

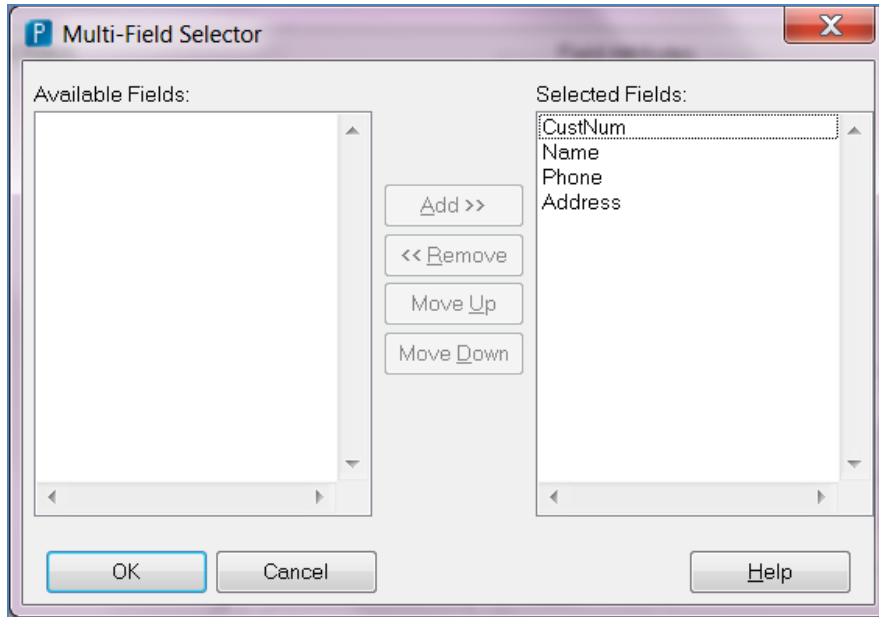




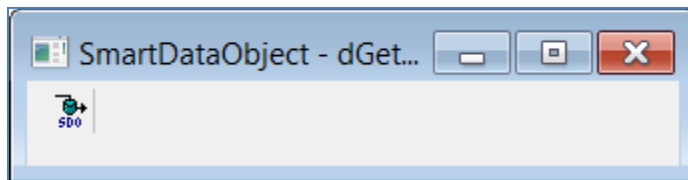
Tip: The **Query Builder** dialog box has five modes: **Table**, **Where**, **Join**, **Sort**, and **Options**. The **Table** mode allows you to add and remove tables to the selected query, The **Where** mode allows you to generate a WHERE clause to specify search criteria in ABL for the selected query. The **Join** mode allows you to create custom joins in ABL for the selected query. The **Sort** mode allows you to generate a SORT clause in ABL for the selected query. The **Options** mode allows you to modify the query attributes listed below and query tuning parameters for the selected query.

6. Click **OK** to close the **Query Builder** dialog box. The **4GL Query** section displays your new query.
7. Click **Next**.
8. Click **Add fields**. The **Column Editor** dialog box appears.
9. Click **Add**. The **Multi-Field Selector** dialog box appears.
10. Select an entry from the **Available Fields** and click **Add**. Similarly, move all the entries to the **Selected Fields** section.





11. Click **OK** to close the **Multi-Field Selector** dialog box. All the customer fields selected are listed for **Fields in SmartDataObject**.
12. Click **OK** to close the **Column Editor** dialog box.
13. Click **Next** and click **Finish**. You have now successfully created a SmartDataObject for the customer table to display all fields. The SmartDataObject is created and opened in Editor.



14. Expand the **SmartObjects** folder in **Project Explorer** view and observe that the dGetCust.w SmartDataObject file is created.

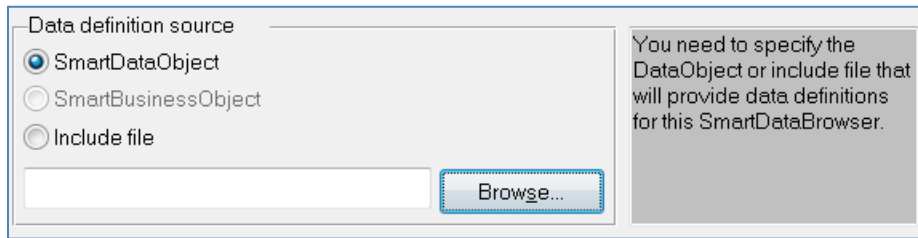
2.6.2 Creating a SmartDataBrowser

Now, you will create a SmartDataBrowser which uses the above SmartDataObject that you created.

1. Select the **SmartObjects** folder from the **Project Explorer** view. Right-click and select **New**→**ABL UI Design**.
2. Select **SmartDataBrowser** in **Object type**. Make sure **Container** displays **\OrderApp\SmartObjects**.
3. Enter **bGetCust.w** in the **File name**.



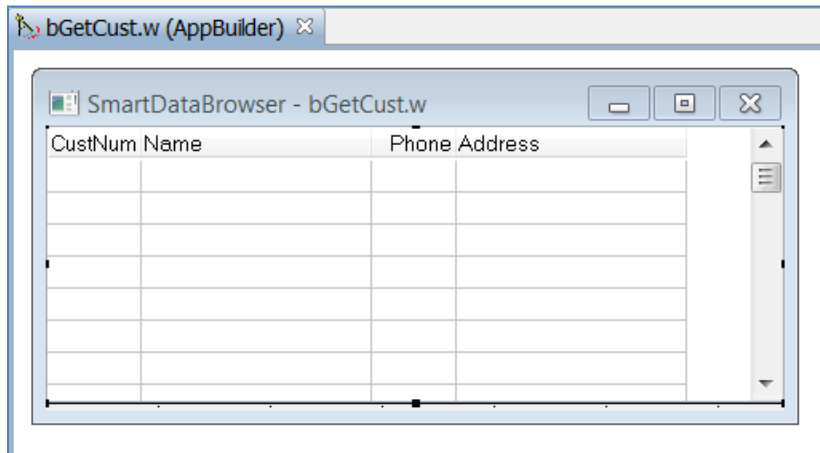
4. Click **Next** until you reach the **Data definition source** section as shown in the screen below:



5. Click **Browse....** The **Open SmartObject** dialog box appears.
6. Click **Browse...** and browse for **dGetCust.w SmartDataObject** created in C:\OpenEdge\WRK\Exchange_PDSOE\OrderApp\SmartObjects\dGetCust.w and click **Open**.
7. Click **Next**. Click **Add Fields** next to the **Fields to display** section.
8. Select an entry from the **Available Fields** and click **Add**. Similarly, move all the entries to the **Selected Fields** section.
9. Click **OK**. All the fields are added to the **Fields to display** section.

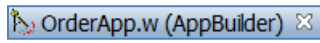


10. Click **Next** and click **Finish** to create the SmartDataBrowser.

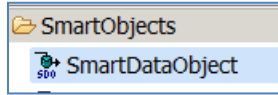


11. Open **OrderApp.w** file from the **Project Explorer** view. Select the **OrderApp.w (AppBuilder)** tab in the editor.





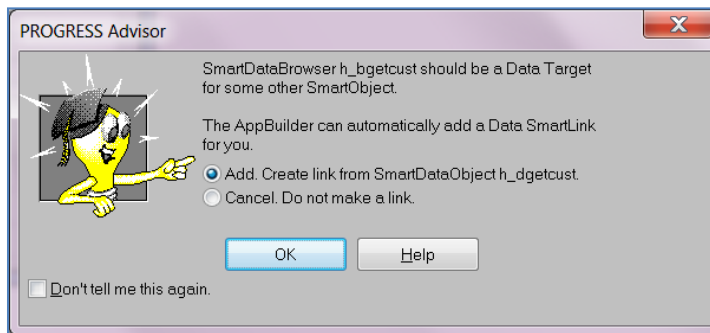
12. Double-click **SmartDataObject** in **Palette**. The **Open SmartObject** dialog box appears.



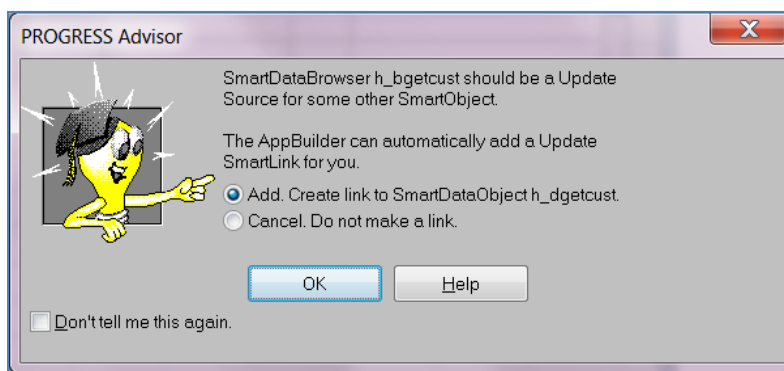
13. Click **Browse...** and select **dGetCust.w** file from the C:\OpenEdge\WRK\Exchange_PDSOE\OrderApp\SmartObjects path. Click **Open**.
14. Click anywhere in **frameCustomer** (frame B). The SmartDataObject is added to the frame.



15. Repeat Step 12 to 14 to add **bGetCust.w** to **frameCustomer**. The **PROGRESS Advisor** dialog-box appears.

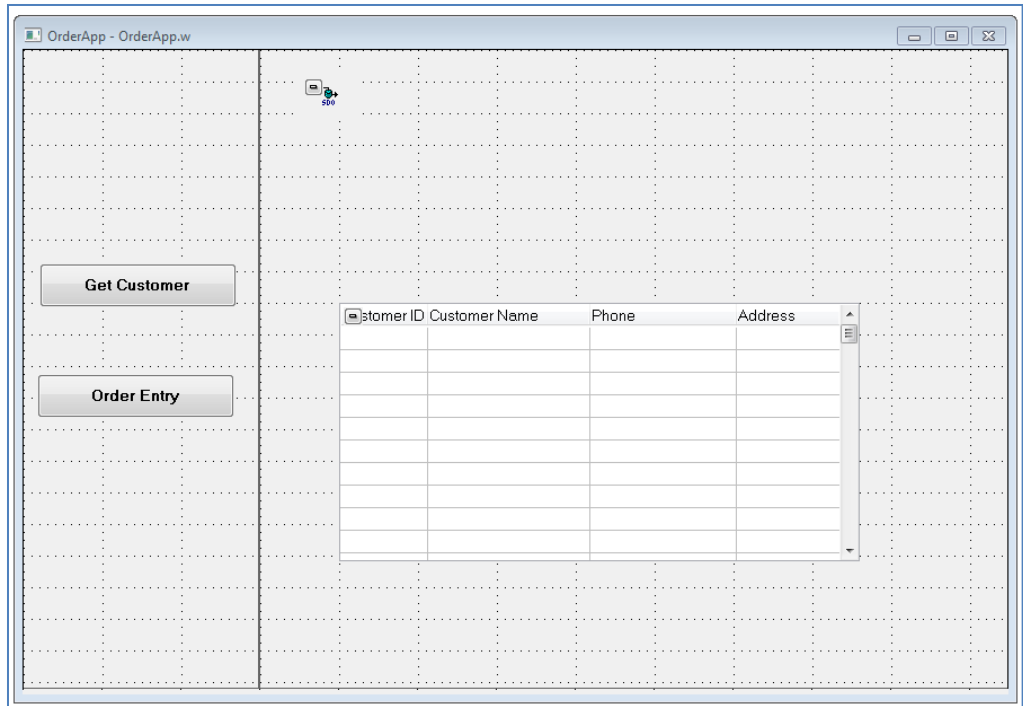


16. Click **OK**.

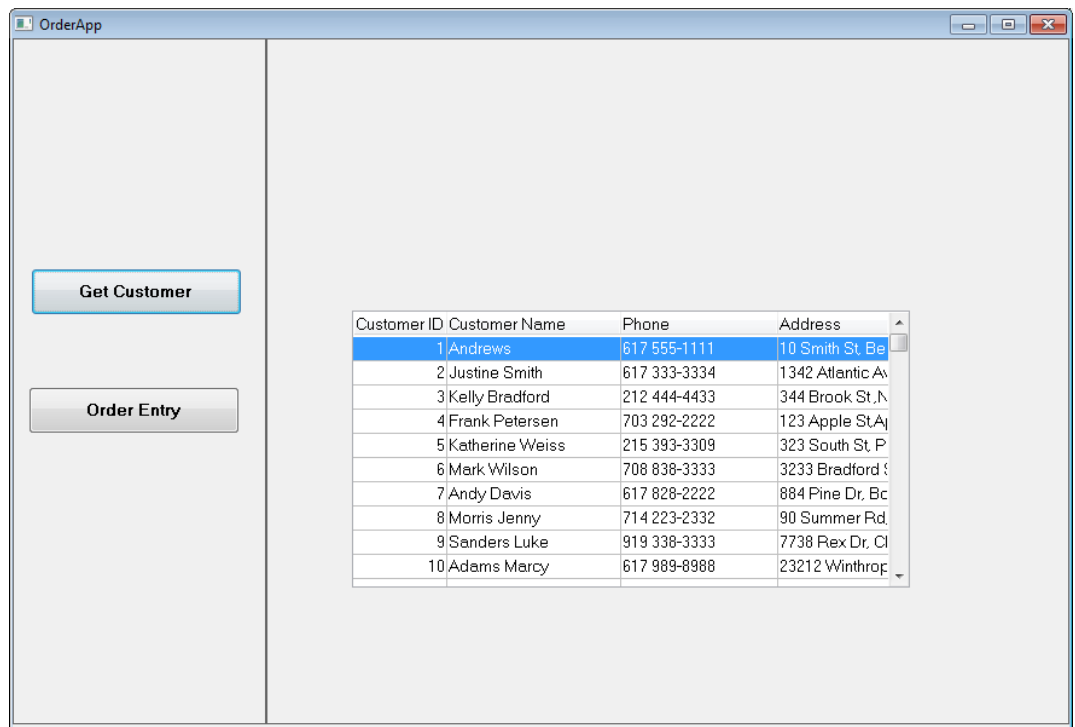


17. Click **OK** to add. A table is added to the editor.





18. Click **File**→**Save** to save the changes.
19. Right-click anywhere on the **OrderApp** window and select **Run**. The OrderApp dialog box is displayed. All customer records that you added are displayed in the table.



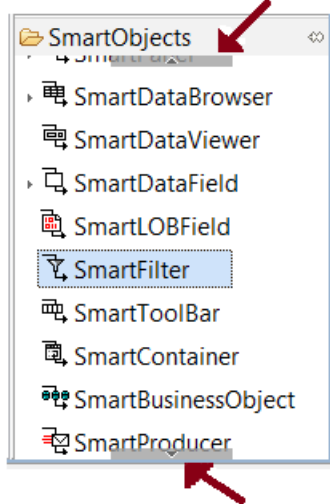
20. Close the **Run** dialog box.



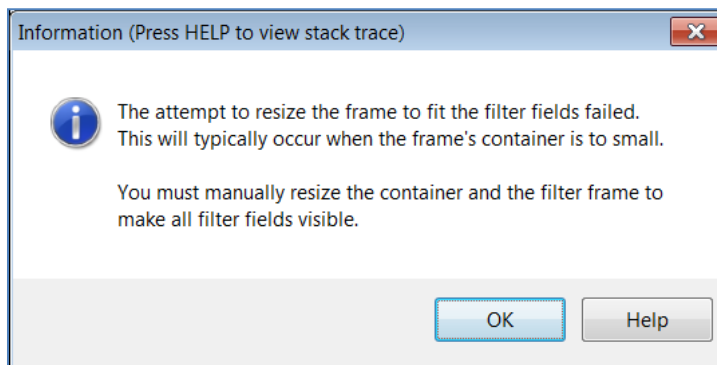
2.6.3 Add SmartFilter to filter customer records (Optional)

You will now add a SmartFilter to filter the customer records based on phone numbers and get the corresponding customer records.

1. In **Palette**, click the down arrow to go to the next items in **SmartObjects** category.

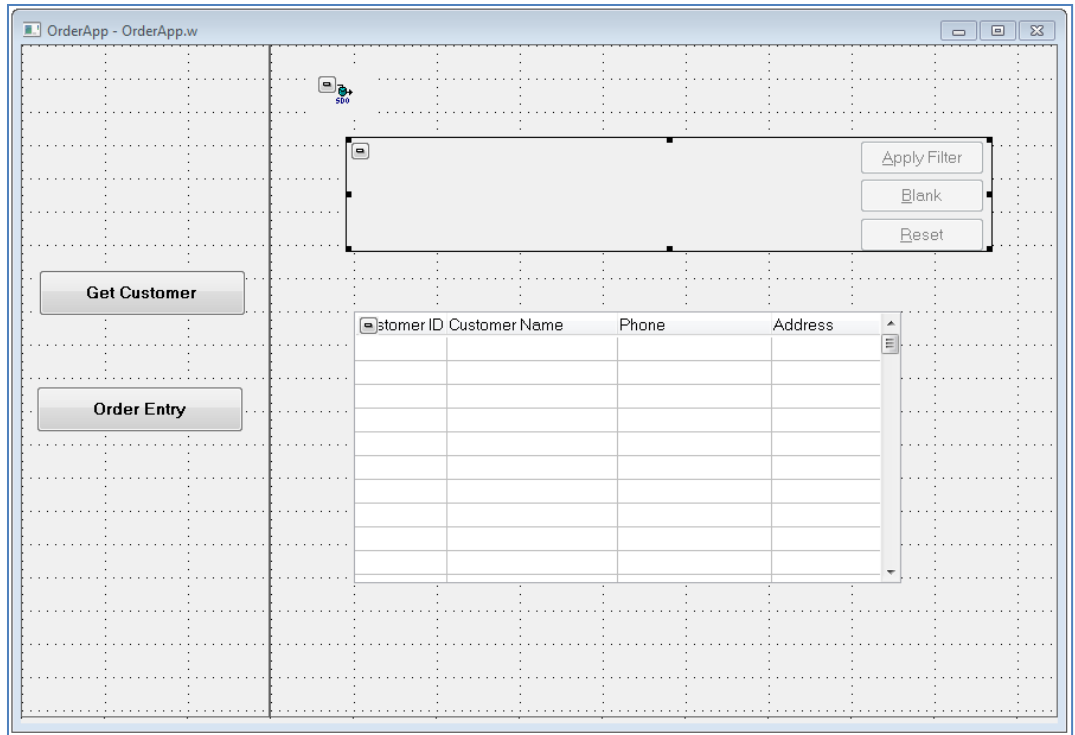


2. Select **SmartFilter** and click **frameCustomer** (frame B). Depending on the size you specified for **frameCustomer**, you may see the following message. Click **OK** and resize the filter frame manually.

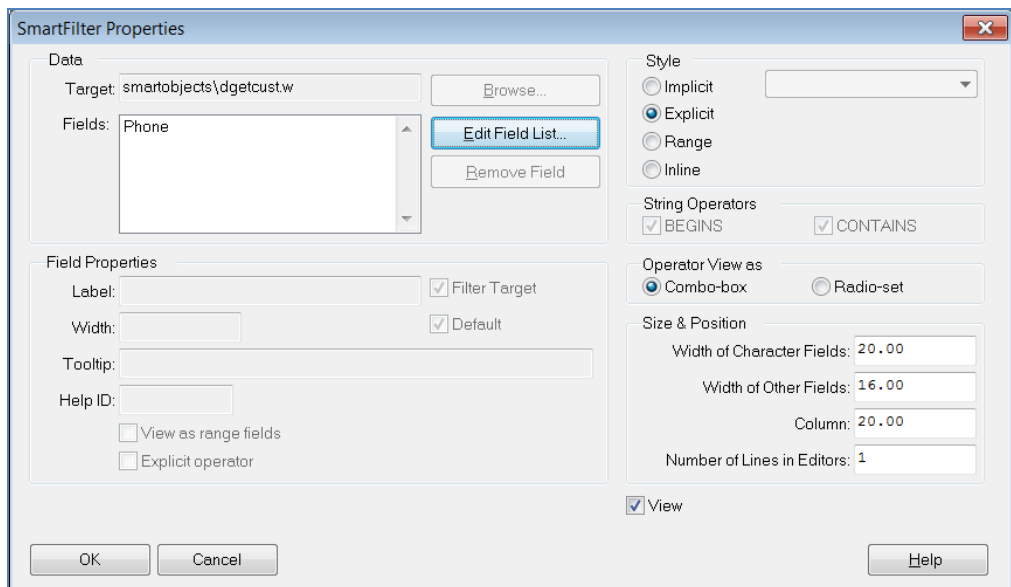


3. Click **OK** to close the message. The PROGRESS Advisor message appears. Click **OK** to add create link to smart object.
4. Adjust the filter and the customer table as shown in image below:



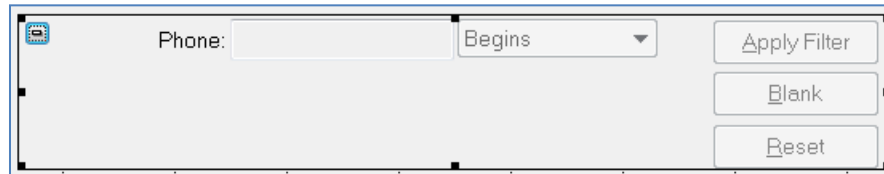


5. Right-click the filter and select **Instance Properties...**. The **SmartFilter Properties** dialog box appears.
6. Click **Edit Field List...**. Select the Phone entry from the **Available Fields** and click **Add**. It moves to **Selected Fields**.
7. Click **OK** to close the **Multi-Field Selector** dialog box. The **Fields** section displays Phone as shown in the image below:

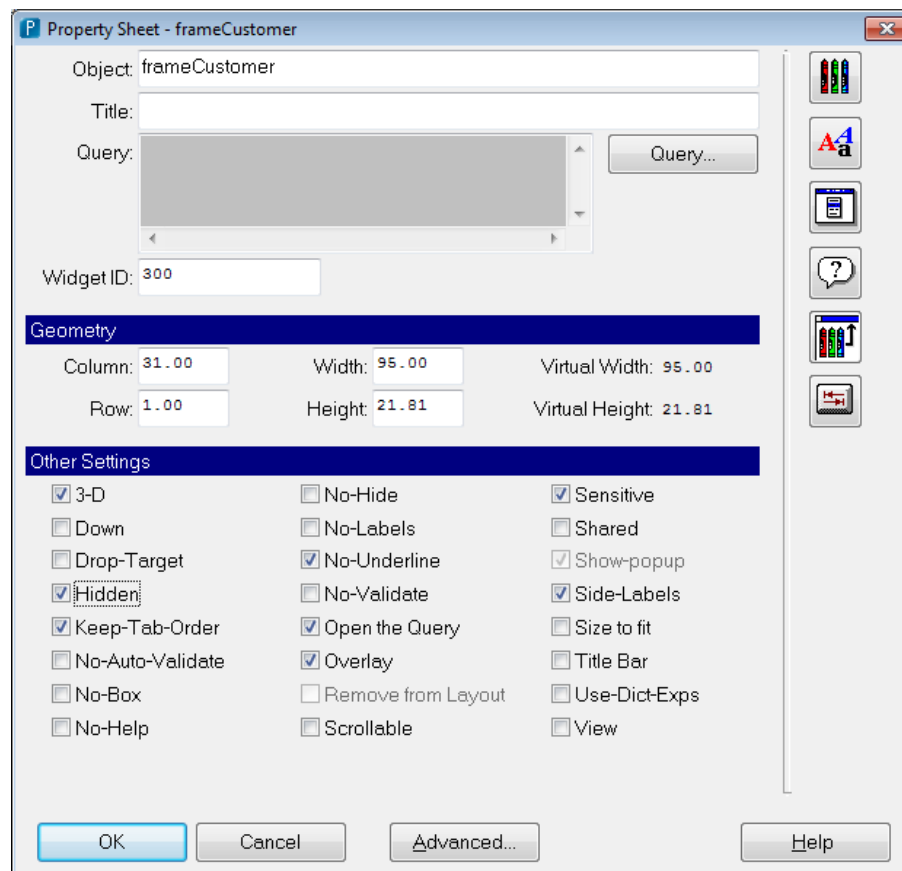


8. Click **OK** to close the **SmartFilter Properties** dialog box.  **PROGRESS**

- Click **OK** to close any error message if displayed. Drag filter control and adjust as shown in below image so that all controls are displayed properly.

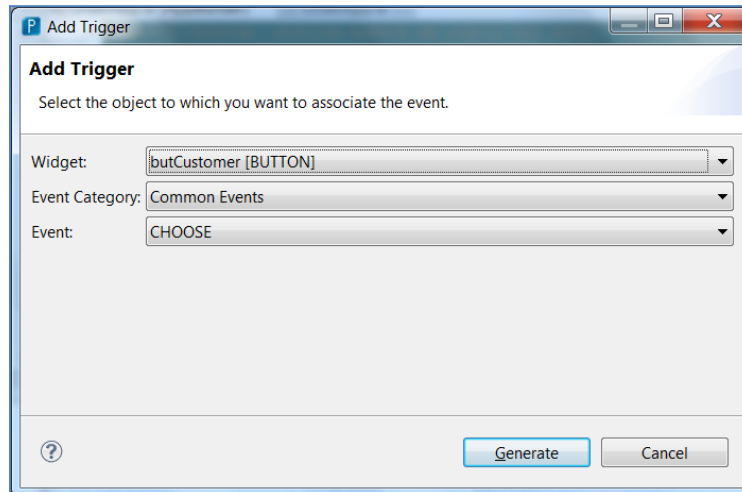


- Click **File**→**Save** to save the changes.
- Double-click **frameCustomer** to open the properties. Clear the **View Property** check box and select the **Hidden property** check box. This hides the frame at runtime.

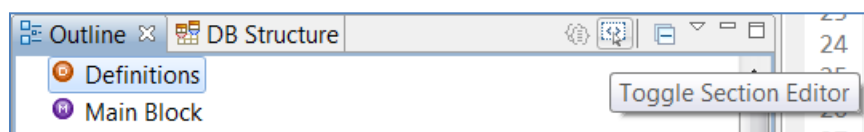


Now we will add a trigger to the **Get Customer** button so that frameCustomer is shown on clicking the button.

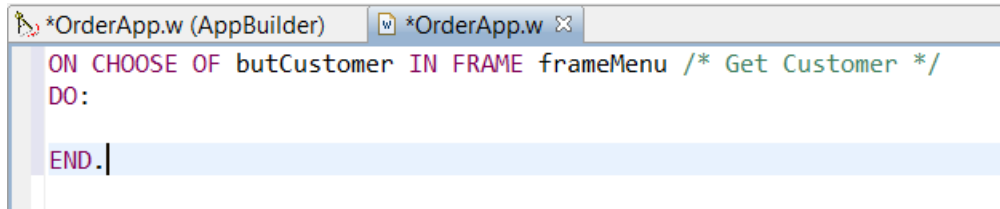
12. Select and right-click the **Get Customer** button in the OrderApp and click **Add Trigger**. The **Add Trigger** dialog box appears.



13. By default, the **CHOOSE** event is selected. Click **Generate**. The source code view of OrderApp.w opens. Observe that the trigger code is generated.
14. From the **Outline** view, expand the **Triggers** section and select **ON CHOOSE OF butCustomer** node.
15. Click **Toggle Section Editor** in the **Outline** view.



16. Observe that only the trigger code appears in the OrderApp.w source code.



```
*OrderApp.w (AppBuilder) *OrderApp.w X
ON CHOOSE OF butCustomer IN FRAME frameMenu /* Get Customer */
DO:
END.
```



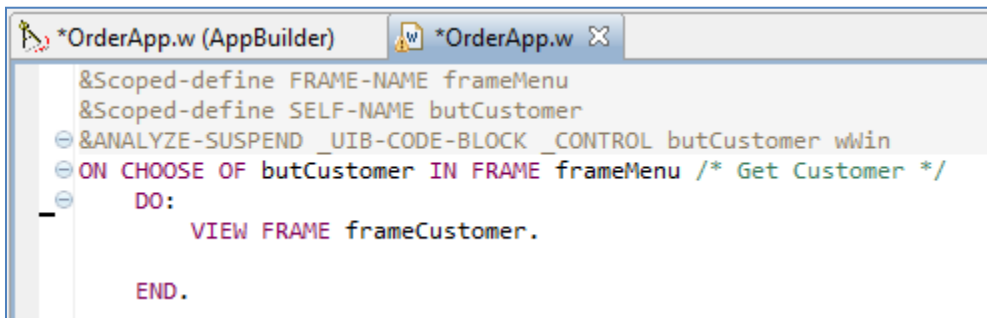
Note: The **Section Editor** mode enables you to view a block of code of an AppBuilder (.w) procedure file while navigating to specific places in the code in the **Outline** view, instead of displaying the entire code of the procedure file. The **Outline** view is not available with the design view. The **Outline** view lists all the structural elements of a procedure file only when the file is opened with the ABL Editor.

17. Add the following line inside the **DO** loop to view the customer frame when the user clicks the **Get Customer** button.

```
VIEW FRAME frameCustomer.
```

18. Click **Toggle Section Editor** in the **Outline** view once more to view the complete code.

19. Click **Ctrl+I** and observe that the code is indented to the right automatically.



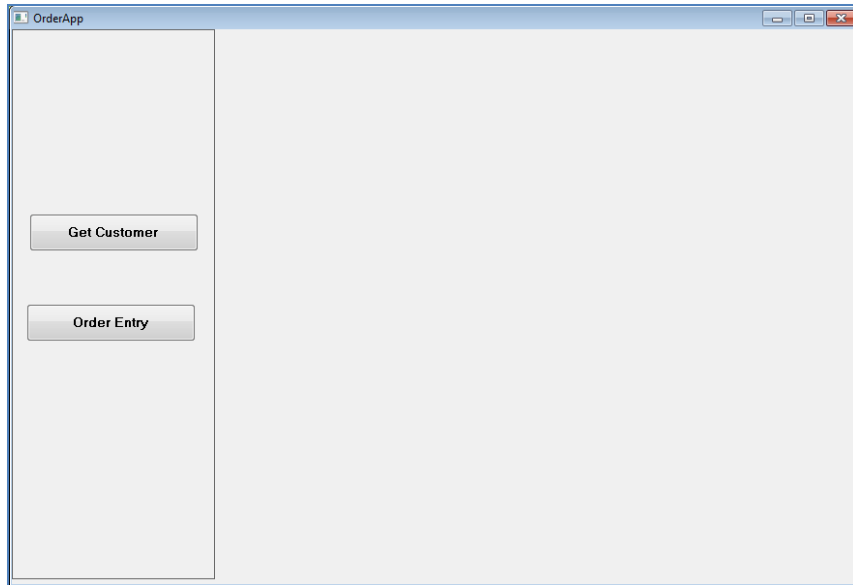
```
*OrderApp.w (AppBuilder) *OrderApp.w X
&Scoped-define FRAME-NAME frameMenu
&Scoped-define SELF-NAME butCustomer
&ANALYZE-SUSPEND _UIB-CODE-BLOCK _CONTROL butCustomer wwin
ON CHOOSE OF butCustomer IN FRAME frameMenu /* Get Customer */
DO:
    VIEW FRAME frameCustomer.
END.
```



Note: The ABL Editor can help you improve readability by automatically indenting lines as you type, when you paste text, or on demand. A separate option (tabular formatting) lets you automatically left-align like elements of statements within a code block, further improving readability. After copy pasting code or typing the code, press **Ctrl+I** or select **Correct Indentation** from editor context menu to correct indentation in the file.

20. Click **File**→**Save** to save the changes.
21. Right-click anywhere in the editor and select **Run**. The **OrderApp** dialog box opens.
22. Observe that by default frameCustomer is not shown.

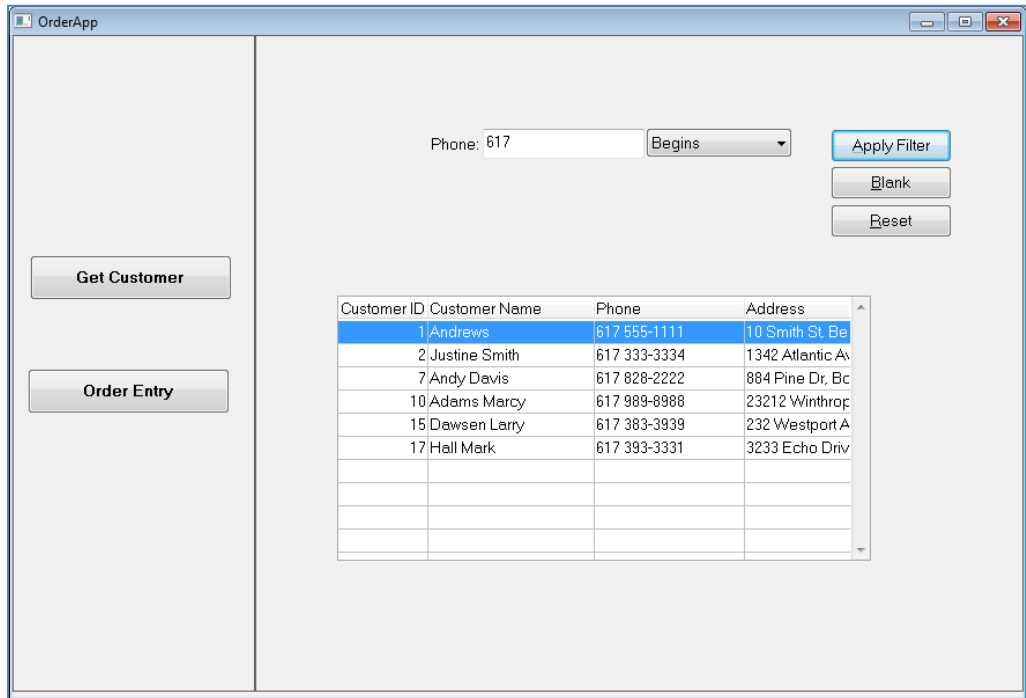




23. Click **Get Customer** and observe that frameCustomer is displayed.
24. Enter first three digits of any phone number in the **Phone** field. Select filter criteria as **Begins** and click **Apply Filter**. Observe that the list is filtered to display the customer records that match the filter criteria.



This completes development of initial screen along with the logic through which you can retrieve existing customer details. When you click **Get Customer**, all the customer records are listed and you can filter the records by phone numbers.



25. Close the **Run** window. Select **File**→**Close All**. All the files opened in the editor are closed.

2.7 Designing the order entry form (Take-home)

This section is a take-home section for you to develop a whole application at your leisure. It is not mandatory to perform this section in this workshop.

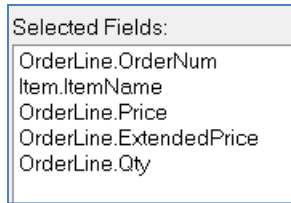
In this section, we will design an order entry form through which you can add items to the list and perform the billing function. Similar to the previous section, you will create a SmartDataObject called dOrderLine.w and SmartDataBrowser using OrderLine to create item tables to retrieve the ordered item details.

1. Create an ABL UI Design file, select **SmartDataObject** in **Object Type**, and **dOrderLine.w** in **File name**.
2. Open the **Query Builder** dialog box and add **OrderLine** and **Item** tables. Select the **Join** option button and join the two tables with condition **Item.ItemNum = OrderLine.ItemNum**:

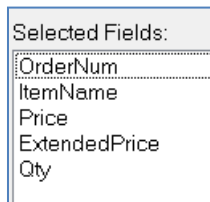
```
Query: EACH workshop.OrderLine NO-LOCK,
      EACH workshop.Item WHERE Item.ItemNum = OrderLine.ItemNum NO-LOC
```

3. Click **Add** in the **Column Editor** dialog box and add the fields in following order:

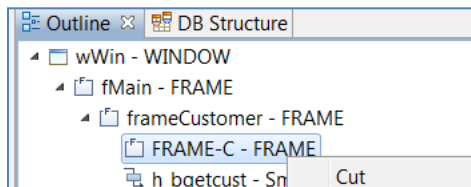




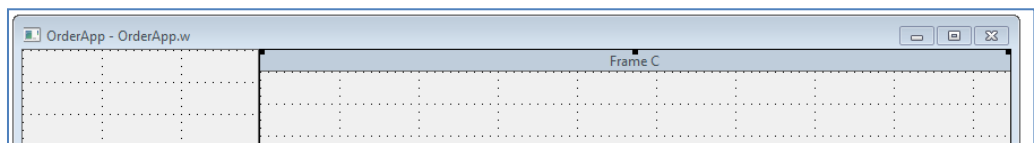
4. Create an ABL UI Design file, select **SmartDataBrowser** as **Object Type**, and **bOrderLine.w** as **File name**.
5. Define the data definition source **smartobjects\dorderline.w** in **SmartDataObject**.
6. Click **Add** in the **Column Editor** dialog box and add the fields in following order:



7. Double-click **OrderApp.w** from the **Project Explorer** view. The file is highlighted if it is already opened in the editor, if not the file opens in the editor.
8. Select **Frame** in **Palette** under the **Widgets** category. Click **frameCustomer**. A new frame is added.
9. From the **Outline** view, right-click **Frame-C** and select **Cut**.



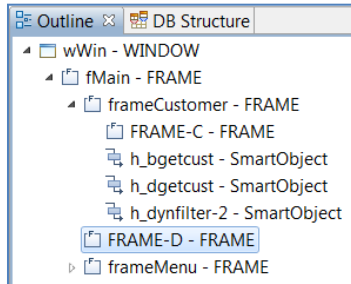
10. Right-click **wWin** in the **Outline** view and select **Paste**. It adds a new frame under **wWin**.
11. Resize the new frame and place it on top of **frameCustomer**.



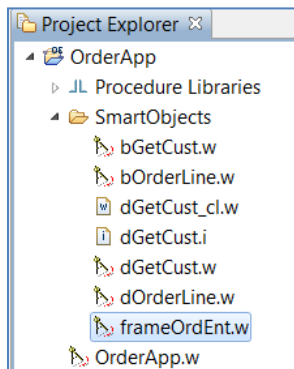
12. Double-click the frame added in the editor. The **Property Sheet – Frame–C** dialog box appears. Specify **frameOrdEntry** in **Object**. Clear the **Title Bar** check box under **Other Settings**.
13. Select **OK** to close the **Property Sheet** dialog box.



- Right-click **frameOrdEntry** and select **Duplicate**. It adds a new frame to the window with same properties. Observe **FRAME-D** listed in the **Outline** view.



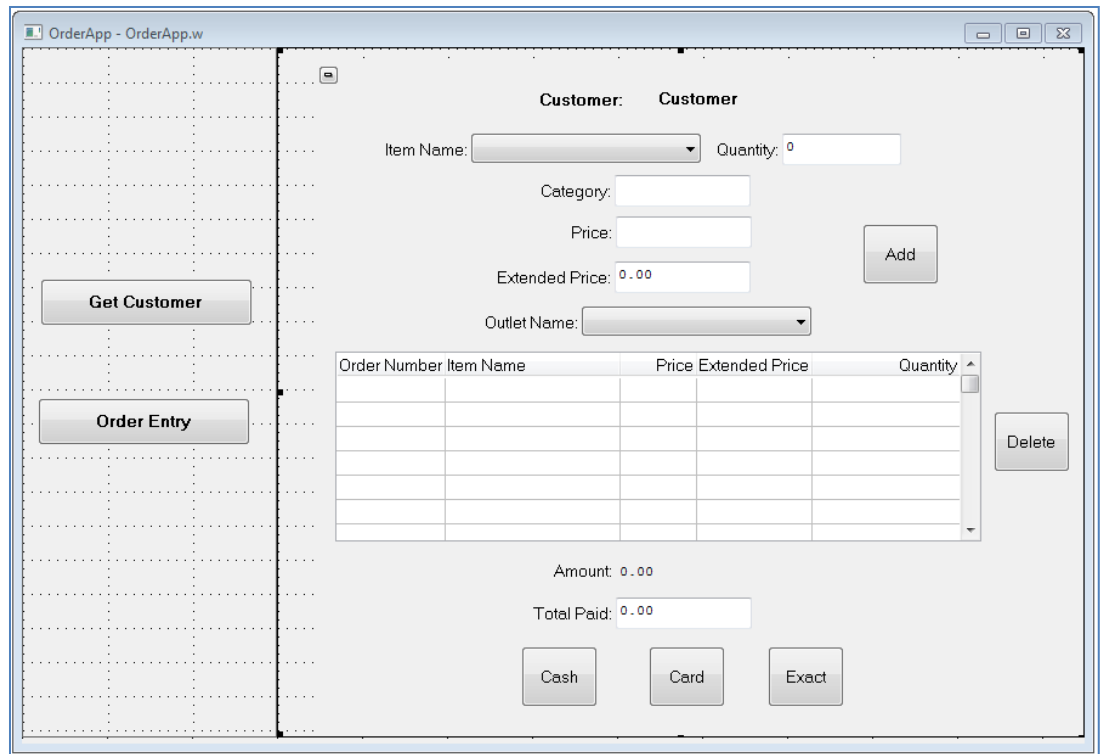
- Right-click **FRAME-D** in the **Outline** view and select **Properties**. The **Property Sheet – Frame – D** dialog box appears. Specify **frameWelcome** in **Object**.
- Select **OK** to close the **Property Sheet** dialog box. Click **File→Save** to save the changes.
- Copy the **frameOrdEnt.w** file from C:\OpenEdge\WRK\PDSOEWorkshopFiles\WorkshopFiles folder and paste in the **OrderApp\SmartObjects** folder in **Project Explorer** view.



- Copy the **Images** folder from C:\OpenEdge\WRK\PDSOEWorkshopFiles\WorkshopFiles and paste in the **OrderApp** folder in **Project Explorer** view.
- Close and reopen the **OrderApp.w** file.
- From the **Outline** view, expand the **wWin→fMain** entry, right-click **frameOrdEntry**, and select **Move-to-Top**.
- Select **SmartDataObject** from the **SmartObjects** section in the **Palette**.
- Browse for the **SmartObjects\frameOrdEnt.w** file. The **PROGRESS Advisor** dialog appears. Click **OK** to add a SmartLink to SmartDataObject.
- Confirm that **frameWelcome**, **frameCustomer** and **frameOrdEntry** should be of the same size and appear overlapping and also that you have added the SmartObject to **frameOrdEntry**.

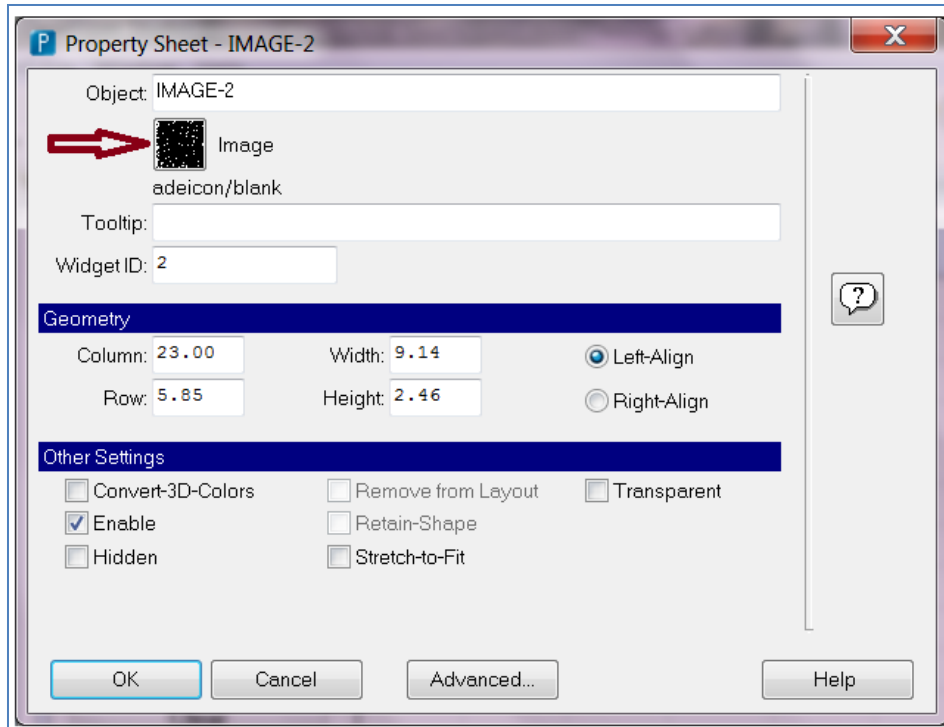


24. When you click **Run**, you will see that the order entry SmartFrame is added to frameOrdEntry.

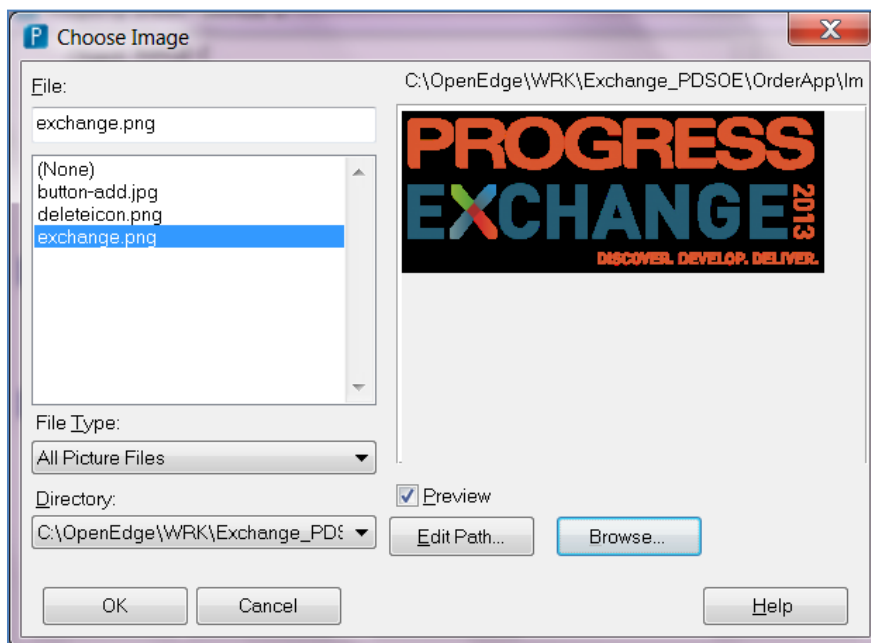


25. Click **File**→**Save** to save the changes.
26. From the **Outline** view, right-click **frameWelcome – FRAME** and select **Move-to-Top**.
27. Select **Image** from the **Widgets** section in **Palette**. Click **frameWelcome**. The Image control is added to the frame.
28. Double-click the image, the **Property Sheet – IMAGE-1** dialog box opens.
29. Click **Image**. The **Choose Image** dialog box appears.





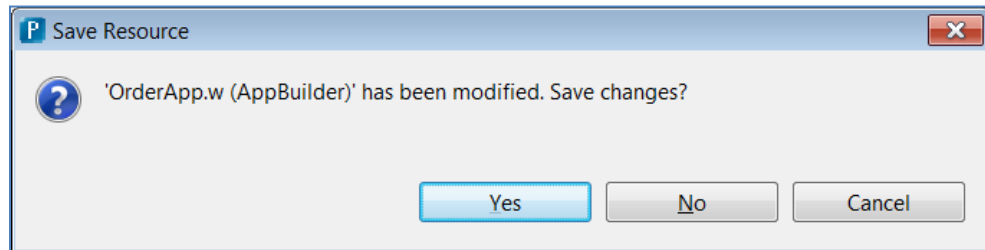
30. Browse for **exchange.png** in the C:\OpenEdge\WRK\Exchange_PDSOE\OrderApp\Images folder under the OrderApp project.



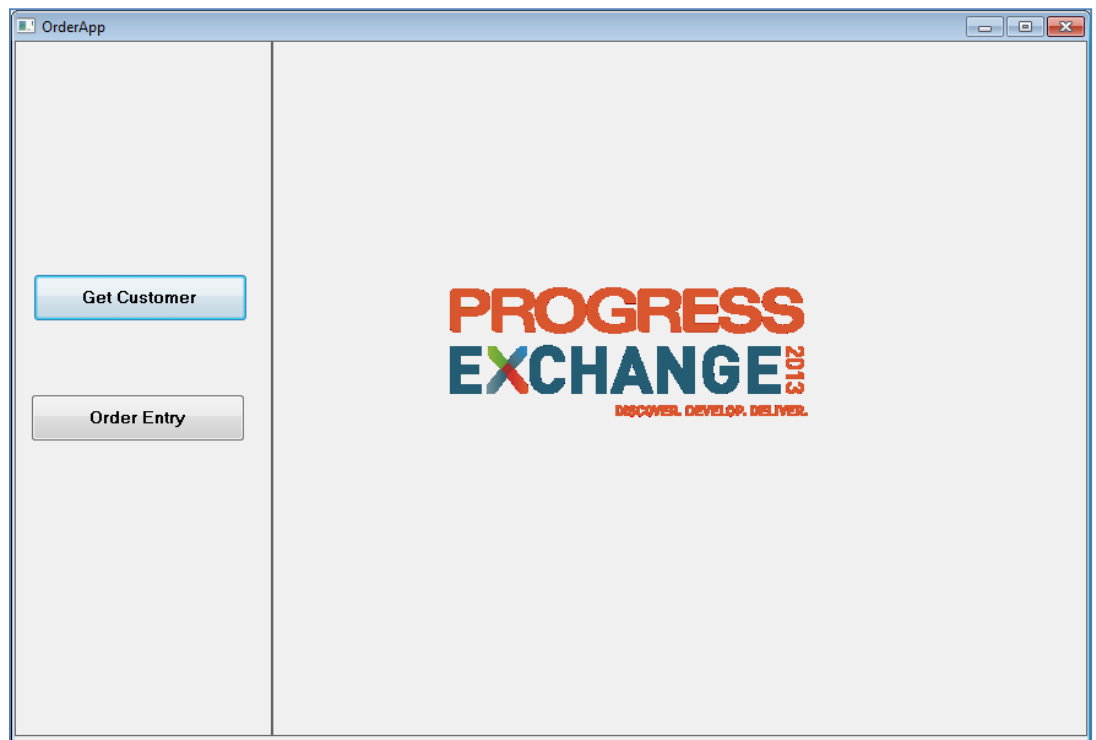
31. Click **OK** to apply the selection of image.
32. Enter **WelcomeImage** in **Object**.



33. Select the **Transparent** check box in **Other Settings**. Click **OK** to close the **Image Property Sheet** dialog box.
34. Resize the image to view it completely. Right-click and select **Run** to run the file.
35. Click **Yes** in the **Save Resource** dialog, if asked for, to save the changes.



The Welcome frame displays the following image:



36. Close the **Run** window.

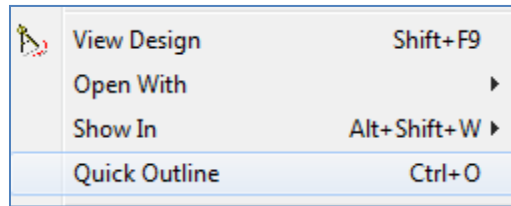
2.8 Linking Get Customer and Order Entry buttons to the respective frames (Take-home)

This section is a take-home section for you to develop a whole application at your leisure. It is not mandatory to perform this section in this workshop.

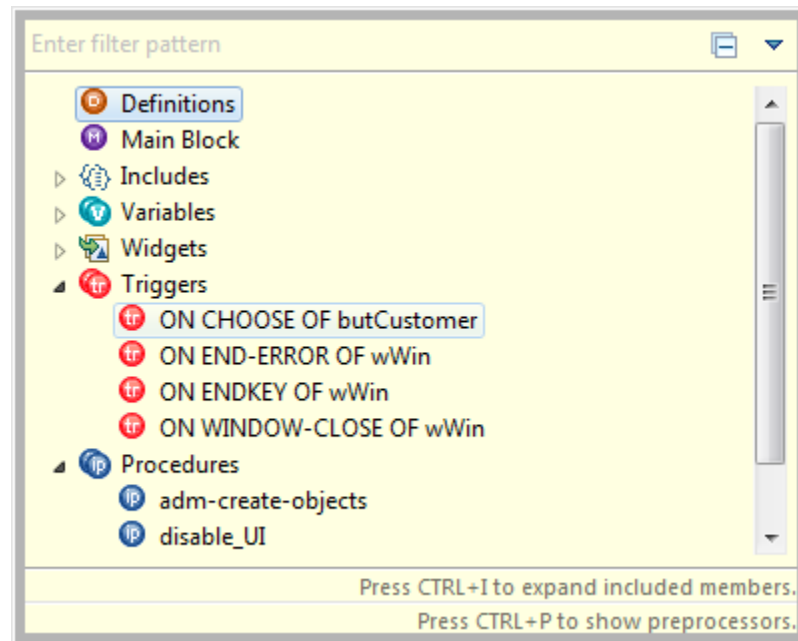
This section shows how to edit code in the source editor. You will add triggers to the buttons and the corresponding code to be executed for each trigger.




1. Right-click **OrderApp** and select **View Source** to view the source code editor.
2. Right-click anywhere in the editor and select **Quick Outline**. The **Quick Outline** view opens.



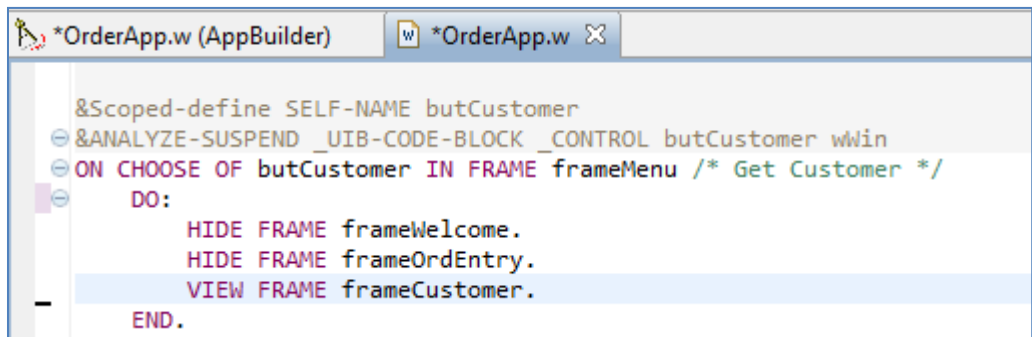
3. Select **ON CHOOSE OF butCustomer** under **Triggers** node.



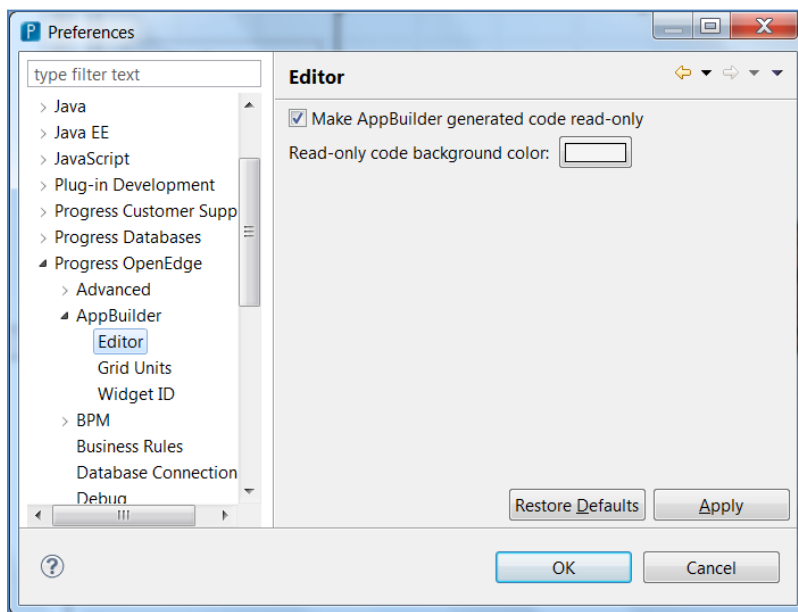
 **Tip:** **Quick Outline** is a pop-up window that shows a tree view of the ABL code file which is currently open in the ABL Editor. You can click a node in the tree view to navigate to a particular section of the file. It is similar to the **Outline** view, but it is more convenient to work with.

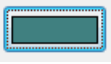
4. Add following lines in the trigger to hide the other two frames created. Click **Ctrl+I** to indent code.

```
HIDE FRAME frameWelcome.
HIDE FRAME frameOrdEntry.
```



5. Click **File**→**Save** to save the changes.
6. Select **Window**→**Preferences** to open the **Preferences** dialog box.
7. Select **Progress OpenEdge**→**AppBuilder**→**Editor**. By default, the **Make AppBuilder generated code read-only** check box is selected.



8. Click the color box for read-only code background color to any dark color like . Click **Apply** and **OK** to close the **Preferences** dialog box.
9. Observe that read-only code is shown in selected color.



```

OrderApp.w (AppBuilder) OrderApp.w
294
295 &Scoped-define FRAME-NAME frameMenu
296 &Scoped-define SELF-NAME butCustomer
297 &ANALYZE-SUSPEND _ITB-CODE-BLOCK CONTROL butCustomer
298 ON CHOOSE OF butCustomer IN FRAME frameMenu /* Get Cu
299 DO:
300     HIDE FRAME frameWelcome.
301     HIDE FRAME frameOrdEntry.
302     VIEW FRAME frameCustomer.
303
304     END.

```

10. Try typing in the read-only section and observe that it is not allowed.



Note: You can make the sections of the AppBuilder-generated code read-only and foldable in the ABL Editor. This ensures that users do not modify the AppBuilder-generated code while editing the ABL procedure (.w) file in the ABL Editor. Editing the AppBuilder-generated code might corrupt the AppBuilder procedure (.w) file, and display errors when trying to open in the ABL GUI Designer.

The Find/Replace option does not work with the read-only or auto-generated code of an AppBuilder procedure file. You cannot rename an internal procedure or function name which is a part of the auto-generated or read-only code section.

11. Add the **CHOOSE** trigger for the **Order Entry** button.

12. Add the following code to hide other frames and view the Order Entry frame.

```

HIDE FRAME frameCustomer. /*Hide frameCustomer*/
gOrderNum = NEXT-VALUE (NextOrderNum). /*Generate Order Number*/
VIEW FRAME frameOrdEntry IN WINDOW wWin. /*View frameOrdEntry*/
Run viewFrame in h_frameordent. /*Initialize default values in Order
frame*/

```

13. From the **Outline** view, expand the **Includes** section and select {src/adm2/widgetprto.i}. It marks the section in the source code view.



Note: The **Outline** view shows the structure of the code in the ABL Editor buffer that currently has focus, and provides an easy way to navigate to specific places in the code. The elements inside the inactive preprocessor regions appear in gray in the **Outline** view. This is similar to include nodes but without the include decorator.

The **Outline** view supports the following navigation techniques:

- Click an element to position the cursor at the declaration of that element. Black elements are declared in the current file.
- Double-click a black include file name to open that include file.
- Double-click an element labeled in gray type to open the Include file and position the cursor at the declaration of that element. Gray elements are declared in include files.

14. Press enter after the {src/adm2/widgetprto.i} include statement. Copy and paste the following code.



```

HIDE FRAME frameOrdEntry.
HIDE FRAME frameCustomer.

/*Create a global shared variable*/
&IF DEFINED(gCust) <> 1 &THEN
DEFINE NEW GLOBAL SHARED VARIABLE gCust AS CHARACTER.
DEFINE NEW GLOBAL SHARED VARIABLE gCustNum AS INTEGER.
&ENDIF

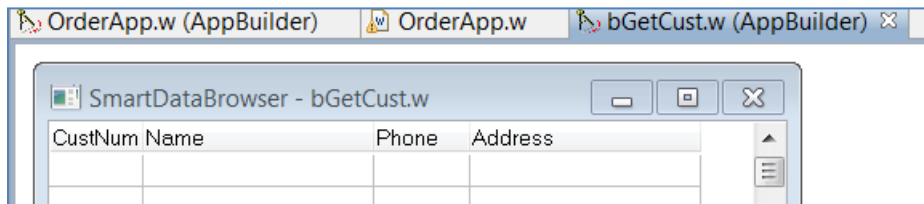
&IF DEFINED(gOrder) <> 1 &THEN
DEFINE NEW GLOBAL SHARED VARIABLE gOrder AS INTEGER.
&ENDIF

&IF DEFINED(gOrderNum) <> 1 &THEN
DEFINE NEW GLOBAL SHARED VARIABLE gOrderNum AS INTEGER NO-UNDO.
&ENDIF

&IF DEFINED(gCust) <> 1 &THEN
DEFINE NEW GLOBAL SHARED VARIABLE gCust AS CHARACTER.
DEFINE NEW GLOBAL SHARED VARIABLE gCustNum AS INTEGER.
&ENDIF

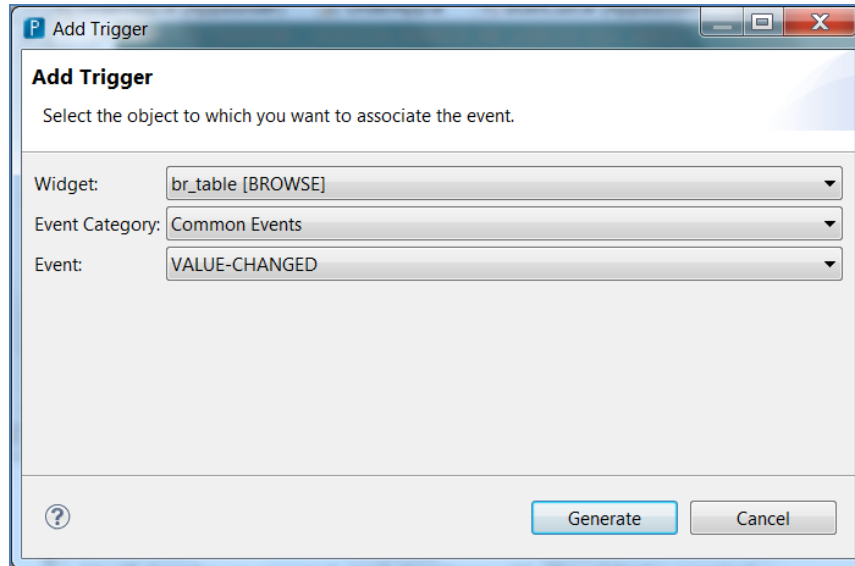
```

15. Save the changes.
16. Right-click the source code editor and select **View Design**. The **Design** view of OrderApp opens.
17. Double-click **bGetCust.w** under the **SmartObjects** folder in **Project Explorer** view. The design view of SmartDataBrowser opens.



18. Right-click and select **Add Trigger**. The **Add Trigger** dialog box appears. Select **br_table [BROWSE]** in **Widget** and **VALUE-CHANGED** in **Event**.





19. Click **Generate**. Observe that the source view opens and the cursor is placed inside trigger code.
20. Add the following code after the Do loop:

```
gCust = {fnarg WidgetValue 'Name'} .
gCustNum = {fnarg WidgetValue 'CustNum'} .
```

It appears as follows:

```
ON VALUE-CHANGED OF br_table IN FRAME F-Main
DO:
  {src/adm2/brschngc.i}
  gCust = {fnarg WidgetValue 'Name'} .
  gCustNum = {fnarg WidgetValue 'CustNum'} .

END.
```

21. Expand **Includes** in the **Outline** view and select `{src/adm2/widgetprto.i}`
22. Press enter after the `{src/adm2/widgetprto.i}` include statement, copy and paste the following code to define the global variables. Save the changes.

```
&IF DEFINED(gCust) <> 1 &THEN
DEFINE NEW GLOBAL SHARED VARIABLE gCust AS CHARACTER.
DEFINE NEW GLOBAL SHARED VARIABLE gCustNum AS INTEGER.
&ENDIF
```

23. Double-click the **bOrderLine.w** file under the **SmartObjects** folder in the **Project Explorer** view to open file in the editor.
24. Right-click **SmartDataBrowser** and select **Add Trigger**. Ensure that **VALUE-CHANGED** is selected.



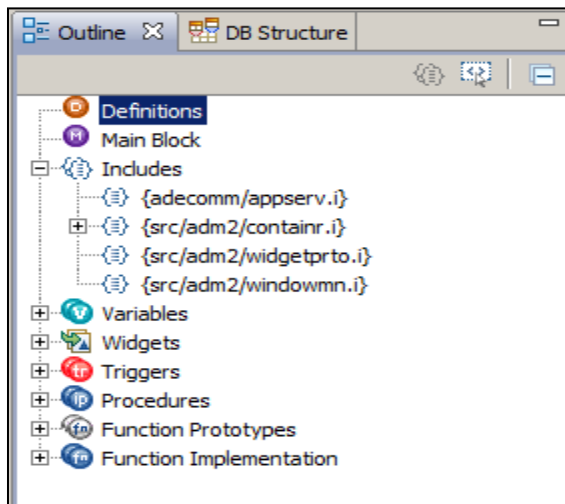
25. Copy and paste the following code in the Do loop:

```
gOrder = {fnarg WidgetValue 'OrderlineNum'} .
```

It appears as follows:

```
ON VALUE-CHANGED OF br_table IN FRAME F-Main  
DO:  
  {src/adm2/brschngc.i}  
  gOrder = {fnarg WidgetValue 'OrderlineNum'} .  
END.
```

26. Open the **Outline** view. Select `{src/adm2/widgetprto.i}` under **Includes** category as below.



27. In the source view, press enter after the `{src/adm2/widgetprto.i}` and copy and paste the following code to define global variables:

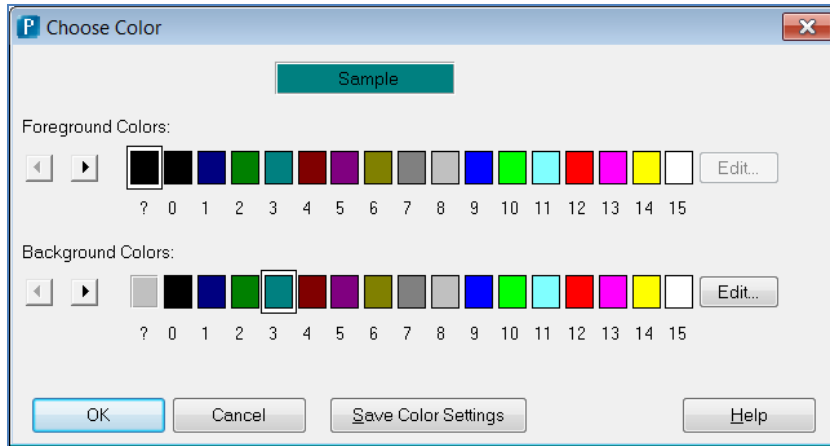
```
&IF DEFINED(gOrder) <> 1 &THEN  
  DEFINE NEW GLOBAL SHARED VARIABLE gOrder AS INTEGER.  
&ENDIF
```



28. Click **File**→**Save All** to save all changes.
29. From the **Outline** view, right-click **frameMenu** and select **Properties**. The **frameMenu Property Sheet** dialog box opens.

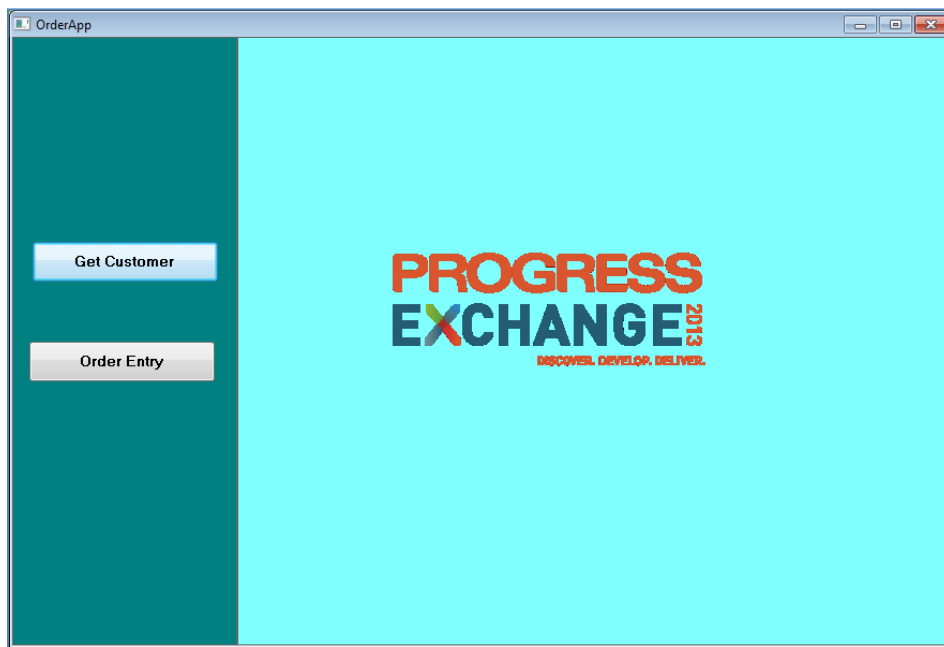


30. Click . The **Choose Color** dialog box appears.



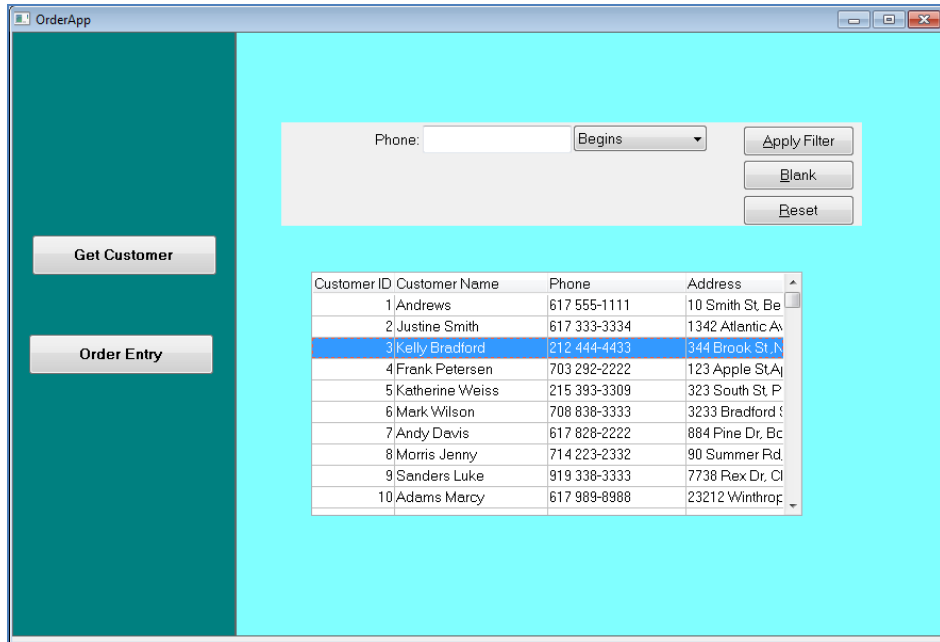


31. Select **Background Color** as  and click **OK**. Click **OK** to close the **Properties** dialog box.
32. Repeat to select the **Background Color** as  for the **frameCustomer**, **frameOrdEntry** and **frameWelcome** frames. Also, open the **frameOrdEnt.w** file and select the same background color using the **Properties** dialog box.
33. Click **File**→**Save All** to save the changes. This completes the development of OrderApp.
34. Select **OrderApp.w (AppBuilder)** in the editor. From workbench menu, select **Run**→**Run As**→**Progress OpenEdge Application**. The **OrderApp Run** dialog box displays:

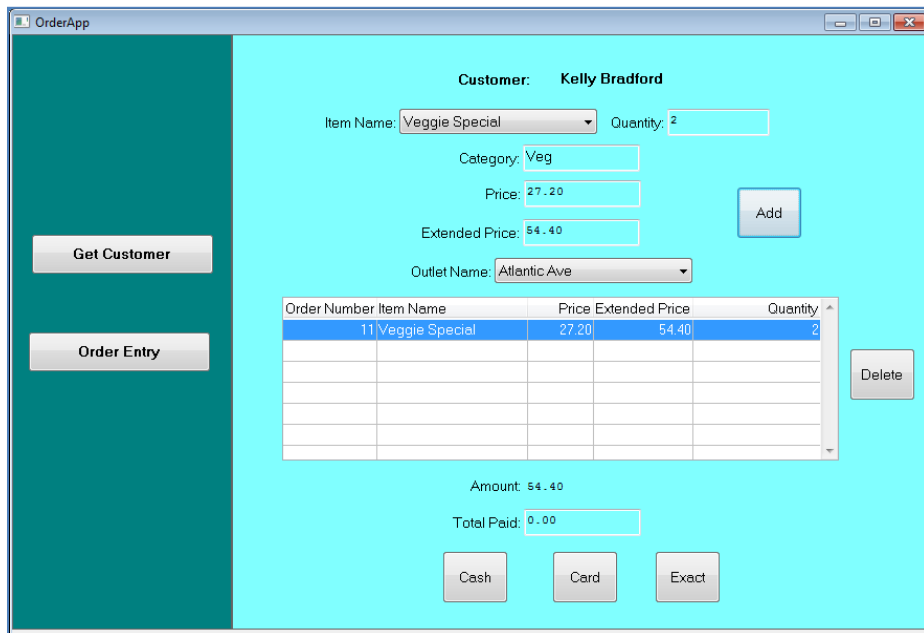


35. Click **Get Customer** and select any customer record from the list.





36. Click **Order Entry**. Observe that the selected customer name is used. Select the items, enter the quantity, and click **Add**. The selected items are added to list.



37. Enter the amount in **Total Paid**, and click **Exact**. The **Order Completed** message appears.
38. Click **File**→**Close All** to close all files in editor.



3 LAB 03: Application development using Progress Developer Studio for OpenEdge – WebSpeed

3.1 Overview

The sections of this lab show you how to create a WebSpeed project and work with the new SpeedScript editor. You will also learn how to assign a SpeedScript file to the WebSpeed server, and run the file on the server.

You will develop a web application as shown below to view the customer records and to add records to the database.

Jump to:

Results List:

1	Andrews	617 555-1111	10 Smith St, Bedford
2	Justine Smith	617 333-3334	1342 Atlantic Ave, Apt 345b, Boston, MA
3	Kelly Bradford	212 444-4433	344 Brook St, New York
4	Frank Petersen	703 292-2222	123 Apple St, Apt 34A, VA
5	Katherine Weiss	215 393-3309	323 South St, Philadelphia

◀ ◻ ◻ ◻ ◻ ▶

PROGRESS EXCHANGE 2013
DISCOVER. DEVELOP. DELIVER.

Customer Details

Name:

Phone:

Address:



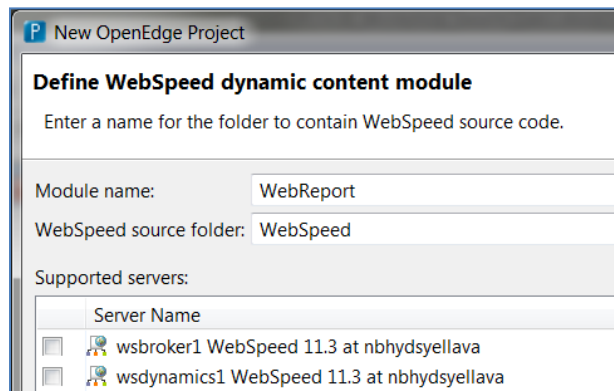
3.2 Prerequisites

Complete **Lab 01: Configuring your workspace and customizing the project** prior to working on this lab.

3.3 Creating a WebSpeed project

This section shows how to create a WebSpeed project.

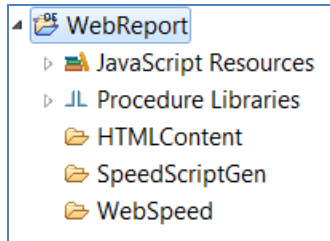
1. From the workbench menu, select **File→New→OpenEdge Project**. The **New OpenEdge Project** dialog box appears.
2. Specify **WebReport** in **Project name**.
3. Select **WebSpeed** from the drop-down list of **Project Type configuration**.
4. Click **Next**. The **Select AVM and layout options** dialog box appears.
5. Click **Next**. The **Define WebSpeed dynamic content module** dialog box appears. The **WebSpeed source folder** displays **WebSpeed**. All the files created under this WebSpeed folder will be published to selected WebSpeed server.



Tip: By default the project name appears in **Module name**. Module is a unit of files and folders which will be published to the server. Under a project, the folders that should be considered for this module can be configured from the **Modules** dialog box in the **Project** properties.

6. Click **Next** until the **Define PROPATH** dialog box appears. The WebSpeed source folder is added to the PROPATH.
7. Click **Next**. The **Select database connections** dialog box appears.
8. Select the check box next to **Exchange_db** and click **Finish** to create the project.
9. The **Open Associated Perspective** dialog box appears. Click **Yes**. The **OpenEdge Server** perspective opens. The project is created along with the respective folders.





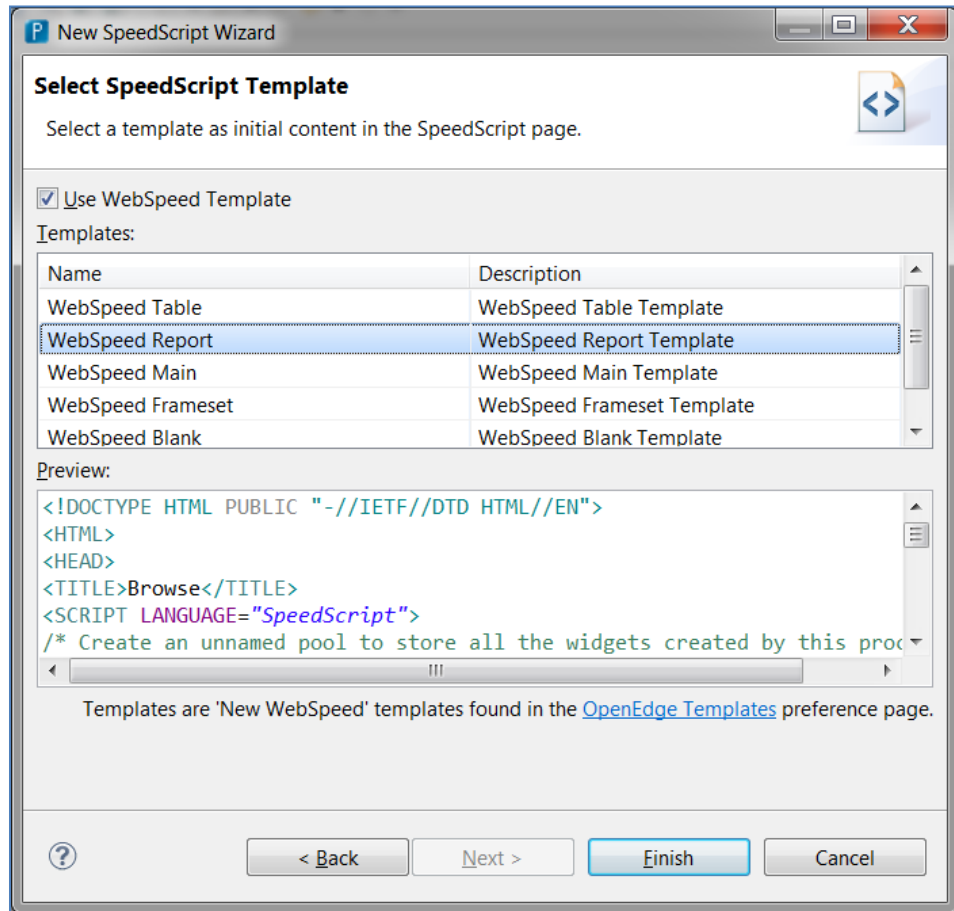
3.4 Creating the SpeedScript file

This section shows you how to create a SpeedScript file using an existing template. We will create a report that lists all the employees from the database.

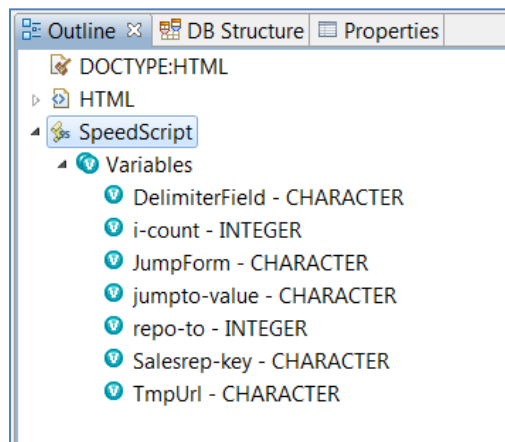
1. Right-click anywhere in the **Project Explorer** view and select **New**→**SpeedScript**. The **New SpeedScript Wizard** dialog box appears.
2. Select the **WebSpeed** folder and enter **CustomerList.html** in **File name**.
3. Click **Next**. The **Select SpeedScript Template** dialog box appears. The default WebSpeed HTML templates are listed. When you choose a template, its HTML markup appears in the **Preview** pane.
4. Select **WebSpeed Report** from the list. The **WebSpeed Report** template has all the pre-defined code for generating a report.



5. Click **Finish** to create the file.



The file is created and opens in editor. The **SpeedScript** node appears in the **Outline** view. The outline of ABL code inside SpeedScript appears as follows:



6. In the source view of CustomerList.html file, find the code with SCOPED-DEFINE preprocessor, replace all &SCOPED-DEFINE statements with the following code:




```

&SCOPED-DEFINE Query-Table      Customer
&SCOPED-DEFINE Query-Field      Name
&SCOPED-DEFINE Query-Index      CustNumIdx
&SCOPED-DEFINE Filter-Field     Phone
&SCOPED-DEFINE Display-Fields   " <TR><TD>" CustNum "</TD><TD>" Name
"</TD><TD>" Phone "</TD><TD>" Address "</TD></TR>"
&SCOPED-DEFINE Result-Rows     5

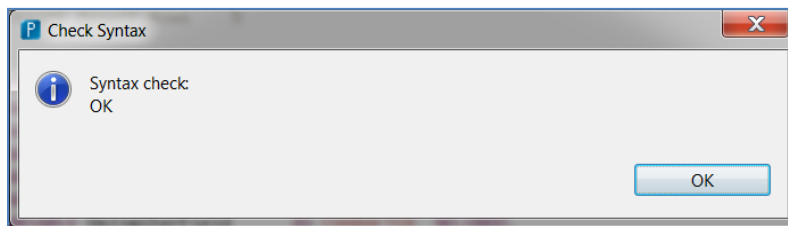
```

By default, the template code refers to tables in Sports2000 database which is a sample database shipped with the product. The code above points to the table, field, and index information corresponding to the database that we are using for this workshop.

7. Find and change the Body type background color to:

```
<BODY STYLE="background-color: #FFF8C6">
```

8. Save the changes.
9. Right-click anywhere on the editor and select **Check Syntax** from the context menu. You should see the following message.

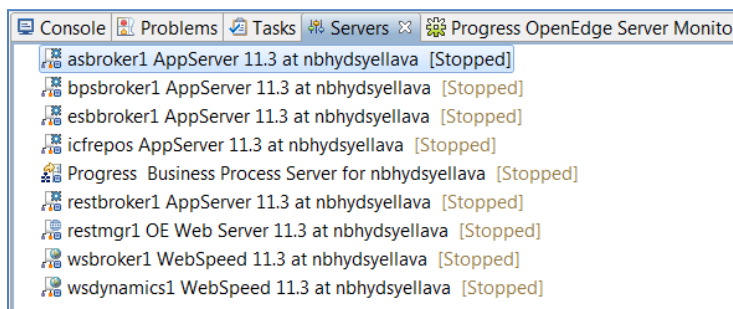


10. Click **OK** to close message.

3.5 Configuring the WebSpeed server and associating the WebSpeed module

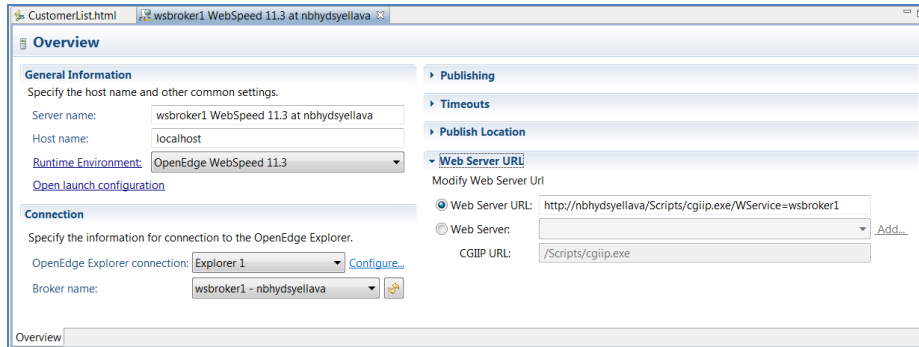
This section shows you how to work with the WebSpeed servers, associate module to the server, edit the server properties, start the server, and publish the modules.

1. Select the **Servers** view.



2. Double-click the **wsbroker1 WebSpeed** server. The **Server Configuration** dialog box opens in editor.
3. Expand **Web Server URL**.



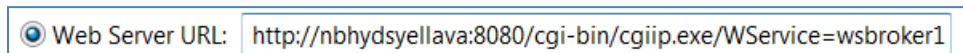


Tip: You use the Server Editor to view or modify the server properties that define the connection to OpenEdge Explorer and the broker.

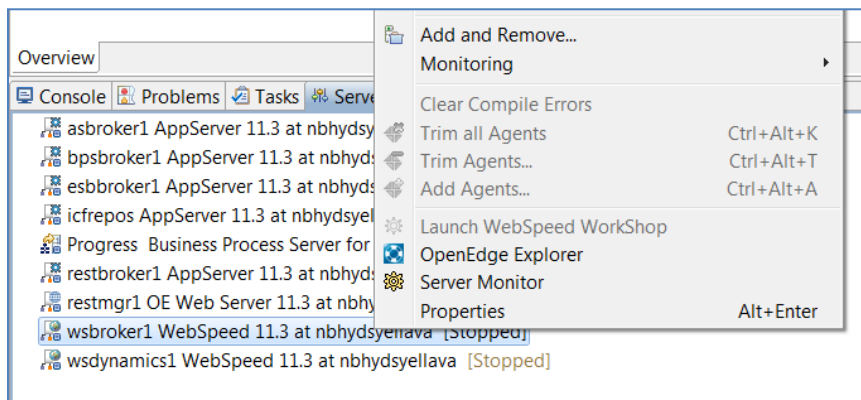
The Server Editor provides information on the following:

- General Information - Provides the host name and other common settings.
- Connection - Specifies the information on connection to OpenEdge Explorer.
- Publishing - Specifies when to publish.
- Timeouts - Specifies the time limit to complete server operations (Start and Stop).
- Publish Location - Specifies the server publish directory.

4. Add **8080** port (on which our Apache web server is running) and remove **Scripts** and add **cgi-bin** to the URL that appears by default, as shown below:

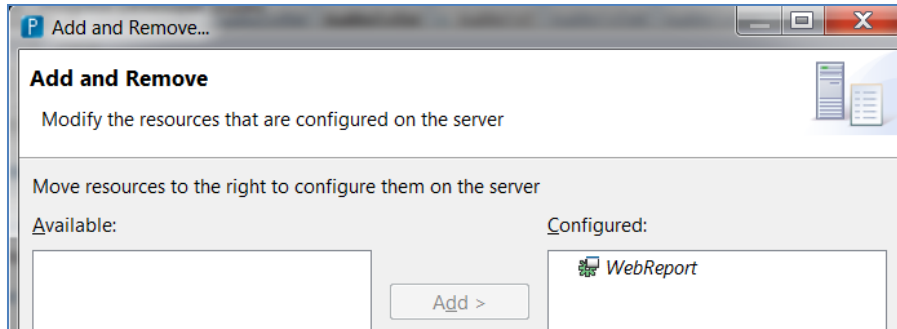


5. Click **File**→**Save**.
6. Right-click **wsbroker1 WebSpeed** in the **Servers** view and select **Add and Remove....** The **Add and Remove** dialog box appears.

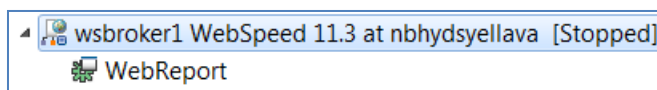


7. Select **WebReport** from the **Available** section and click **Add**.



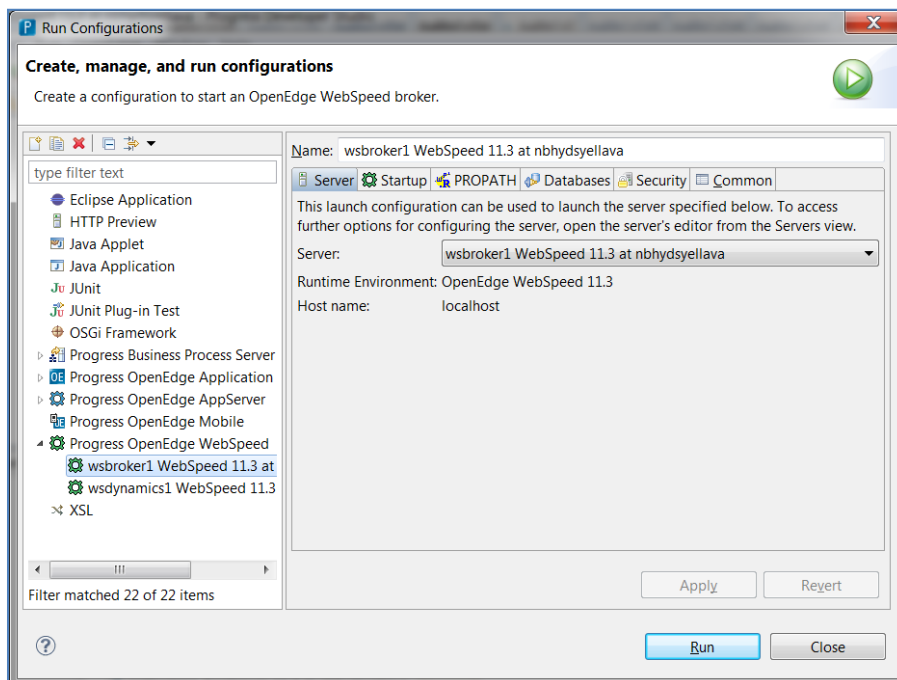


8. Click **Finish**. The **WebReport** module is added and listed under **wsbroker1** server in the **Servers** view.



We are now done with creating an html file to display all the records from customer table. We have also associated the module to our server. The next step is to start our server.

9. From the workbench menu, select **Run**→**Run Configurations**. The **Run Configurations** dialog box appears. All the **Run configurations** grouped by category are listed.
10. Expand **Progress OpenEdge WebSpeed** and select **wsbroker1 WebSpeed**.



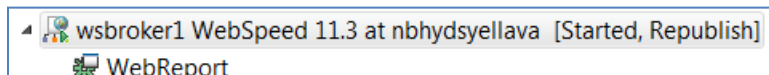


Note: A Run configuration defines the characteristics of the AVM instance under which the selected program runs. These characteristics include for example, startup parameters, PROPATH settings, and environment settings for the AVM session, database connections, and whether the program uses a dedicated instance of the AVM or the instance under which your OpenEdge project is currently running.

You can use the Run Configurations wizard to define all of a Run configuration's characteristics. Although this wizard contains a large number of fields on multiple tabs, defining a Run configuration need not be a complicated task. In fact, with a single click, you can create and run a configuration that uses default settings, and then edit any of these settings, if necessary.

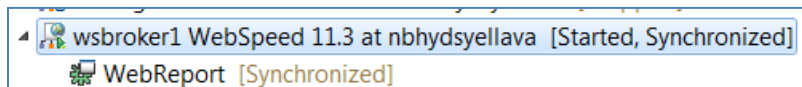
11. Select the **Databases** tab. Select the **Show All** option button to show all database connections available in the workspace and you see the **Exchange_db** connection listed.
12. Select the check box next to **Exchange_db**. Click **Apply** and **Run** to start the server.

The server is started and the status changes to **Started** in the **Servers** view.



We will now publish our code to the WebSpeed broker.

13. Right-click **wsbroker1 WebSpeed** server and select **Publish**. The server status changes to **Synchronized** which means that both our module and the server are in sync.



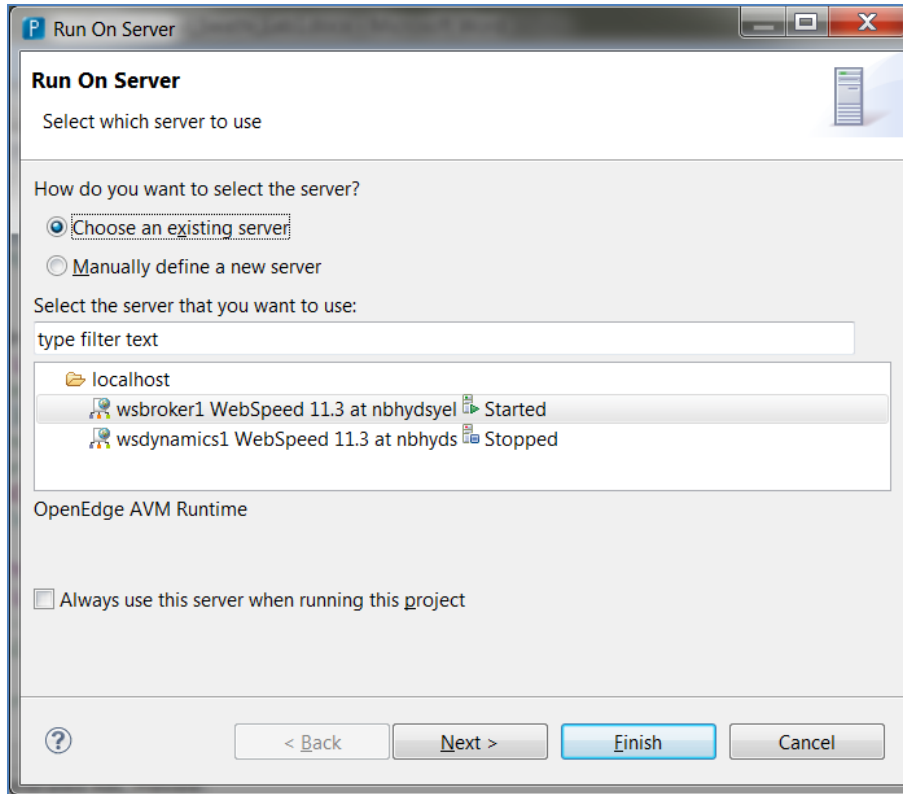
3.6 Running the SpeedScript file on a WebSpeed server

In the above section, we have seen how we can create an html file using default templates and with minimum coding to display records from a particular table.

This section shows how to run the SpeedScript file on the server.

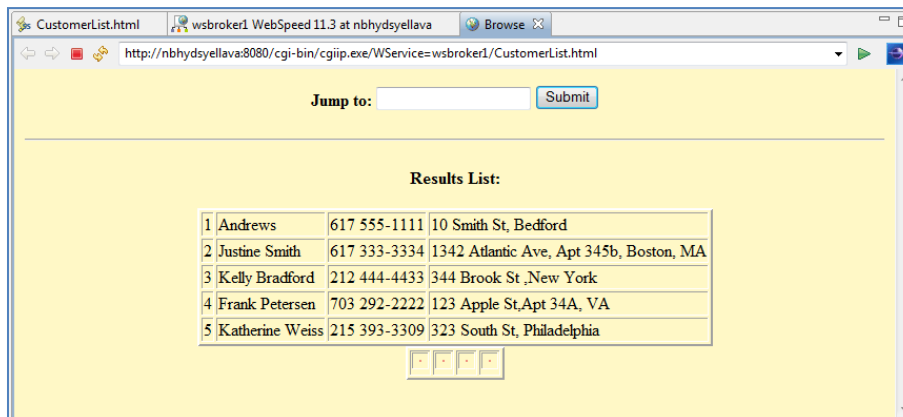
1. Select **CustomList.html** tab in the editor.
2. Right-click and select **Run As**→**Run on Server**. The **Run on Server** dialog box appears and lists all the WebSpeed servers available.





3. Select the **wsbroker1 WebSpeed** server and click **Finish**.

The SpeedScript file runs on wsbroker1 server and the result appears.



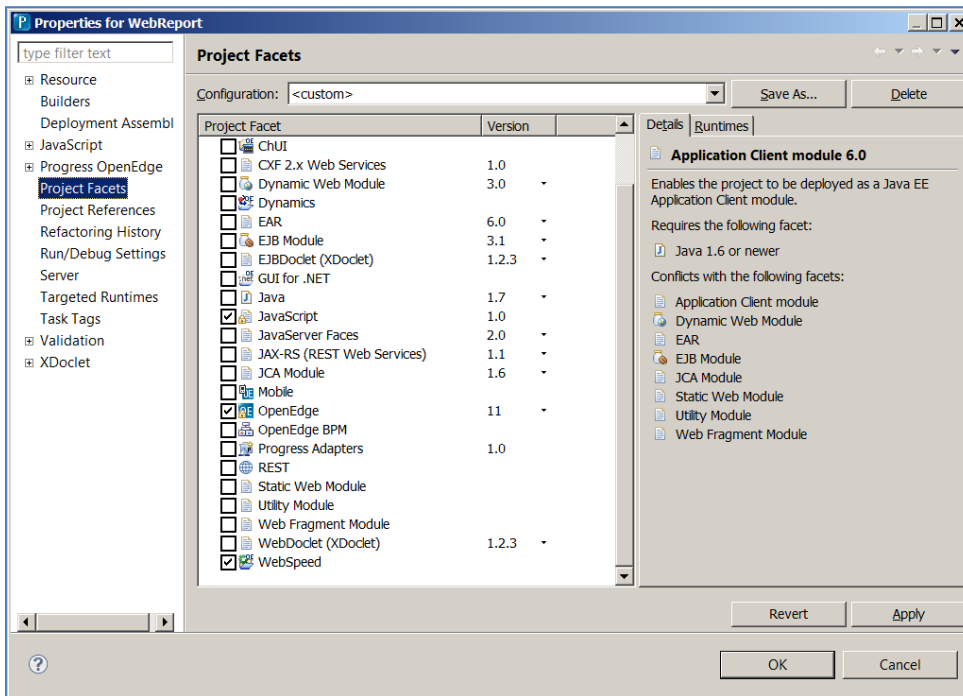
3.7 Adding an AppServer facet to the WebSpeed project

This section shows how to add an AppServer facet to the existing WebSpeed project.

1. Select the **WebReport** WebSpeed project in **Project Explorer** view.

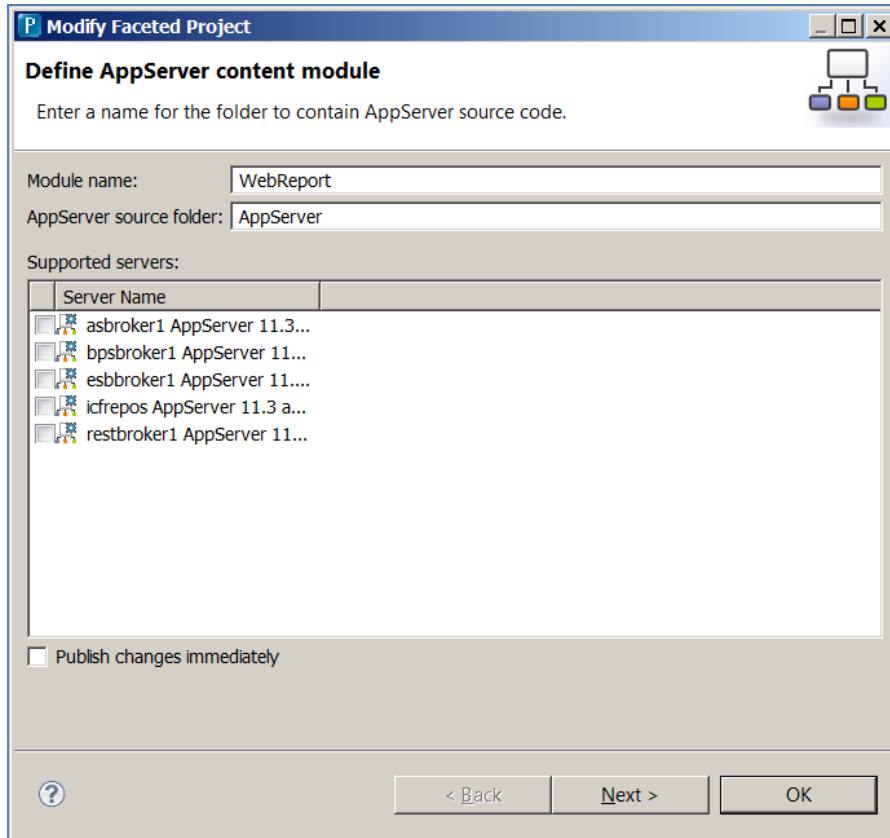


2. Right-click and select **Properties**. The **Properties for WebReport** dialog box appears.
3. Click the **Project Facets** node. The OpenEdge, JavaScript, and WebSpeed facets are selected by default.

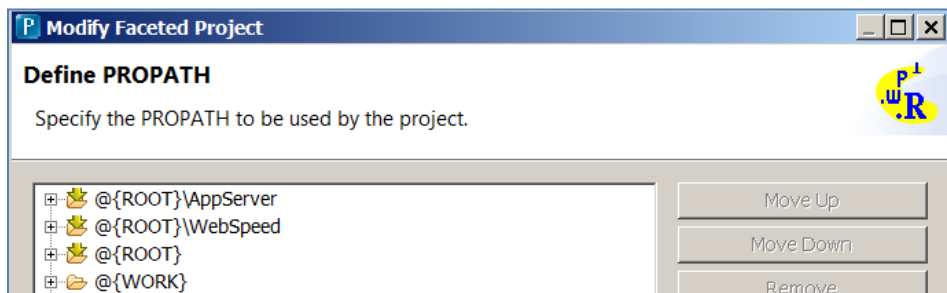


4. Select the **AppServer** check box to add the facet to project.
5. Click **Further configuration available...** link. The **Define AppServer content module** dialog box appears. By default, the **AppServer** folder appears as the **AppServer source folder**.





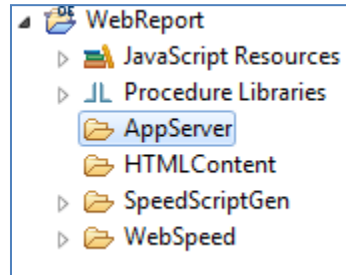
6. Select the check box next to **restbroker1 AppServer** to publish the AppServer files of this project to the restbroker1 AppServer.
7. Click **Next**. The **Define PROPATH** dialog box appears. The AppServer source folder is added to PROPATH.



8. Click **Next**. The **Select database connection** dialog box appears.
9. Click **OK** in the **Modify faceted Project** dialog box.

10. Click **OK** to close the **Properties** page.

The **AppServer** folder is created under the **WebReport** project.



3.8 Working with the SpeedScript editor

This section shows you how to work with the SpeedScript editor. We will learn how to create a SpeedScript file from a blank template and run it on the server.

In Progress Developer Studio for OpenEdge, you edit the embedded SpeedScript files in the OpenEdge SpeedScript editor. Embedded SpeedScript files are HTML files that contain SpeedScript (a subset of ABL) code contained within HTML script elements. The OpenEdge SpeedScript editor supports editing HTML, JavaScript, CSS, and SpeedScript.

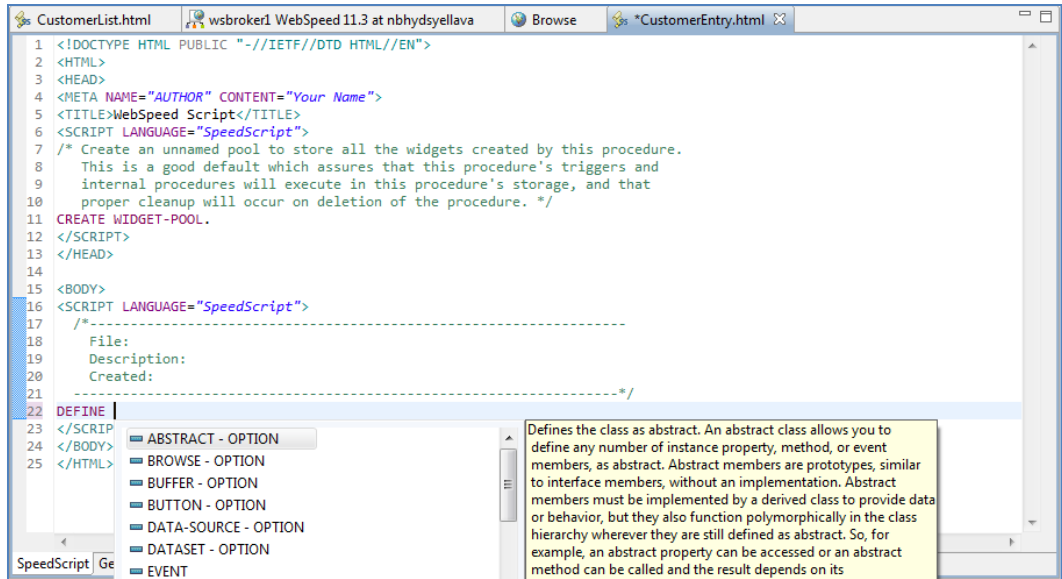
When you double-click a file that has an .htm or .html extension in Progress Developer Studio for OpenEdge, the file opens in the SpeedScript editor by default. The SpeedScript editor assists in writing code including features such as content assist, case correction, keyword expansion, hover help and so on.

Note that the SpeedScript editing features do not apply to HTML, JavaScript, or CSS elements in a SpeedScript file. For example, case correcting (Source > Correct case) only applies to the SpeedScript elements.

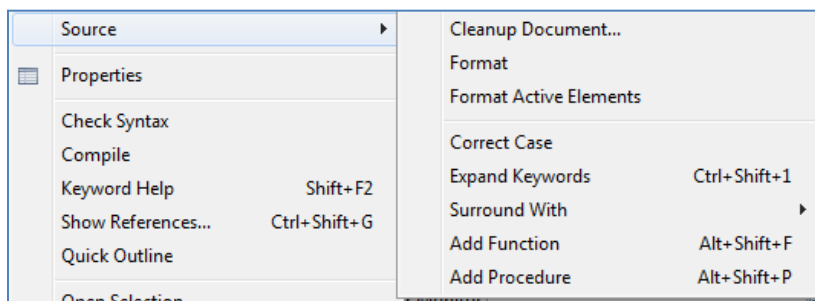
Most SpeedScript editing features are available from the main context menu, or under the source node of the context menu.

1. Create **CustomerEntry.html** SpeedScript file under the **WebSpeed** folder using **WebSpeed Blank Template**. The SpeedScript file opens in the editor.
2. Find the **SpeedScript** tag, type **Define**, add a space and press **Ctrl+Space**. Observe that content assist functions in a similar way as in ABL editor.





3. Right-click on the editor and observe that the **Source** menu displays the ABL options.



4. Delete the whole code from the editor and add the following code:

```

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<META NAME="AUTHOR" CONTENT="Your Name">
<TITLE>WebSpeed Script</TITLE>
<SCRIPT LANGUAGE="SpeedScript">
/* Create an unnamed pool to store all the widgets created by this
procedure.
    This is a good default which assures that this procedure's triggers and
    internal procedures will execute in this procedure's storage, and
    that
    proper cleanup will occur on deletion of the procedure. */
CREATE WIDGET-POOL.

</SCRIPT>
</HEAD>

<BODY STYLE="background-color: #FFF8C6">
<form method="POST">

```



```

<DIV STYLE="text-align: center;">
<image src="`RootURL`/images/exchange.png">
<SCRIPT LANGUAGE="SpeedScript">
    DEFINE VARIABLE CustNum AS INTEGER NO-UNDO.
    DEFINE VARIABLE CustName AS CHARACTER NO-UNDO.
    DEFINE VARIABLE Phone AS CHARACTER NO-UNDO.
    DEFINE VARIABLE Address AS CHARACTER NO-UNDO.
    DEFINE TEMP-TABLE ttCustomer LIKE Customer.
    DEFINE VARIABLE prochandle AS HANDLE.
    DEFINE VARIABLE serverHandle AS HANDLE.
</SCRIPT>
<TABLE border=2>
<TR><TH colspan=2>Customer Details</TH></TR>
<TR><TD>Name: </TD><TD><INPUT TYPE="Text" NAME="CustName"
VALUE="`CustName`"></TD></TR>
<TR><TD>Phone: </TD><TD><INPUT TYPE="Text" NAME="Phone"
VALUE="`Phone`"></TD></TR>
<TR><TD>Address: </TD><TD><TextArea Name="Address"
rows="4">`Address`</TextArea></TD></TR>
<TR></TR>
<TR></TR>
</TABLE>

<INPUT TYPE="submit" NAME="Submit" VALUE="Add Customer" >
</DIV>
<script language="speedscript">
/* ***** Internal Procedures ***** */
*/
IF get-value("Submit") EQ "Add Customer" THEN
    DO:
    IF GET-VALUE ("CustName") NE "" THEN DO:
    CREATE ttCustomer.
    ASSIGN ttCustomer.Name = GET-VALUE ("CustName")
        ttCustomer.CustNum = NEXT-VALUE (NextCustNum)
            ttCustomer.Phone = get-Value("Phone")
                ttCustomer.Address = get-value("Address").

        CREATE SERVER serverHandle.
        DO ON ERROR UNDO, THROW:
            serverHandle:CONNECT("-AppService restbroker1 -H
localhost -sessionModel Session-free").
            RUN custom.p PERSISTENT SET prochandle ON SERVER
serverHandle.
            RUN createCustomer IN prochandle (INPUT-OUTPUT TABLE
ttCustomer).
        FINALLY:
            DELETE PROCEDURE procHandle NO-ERROR.
            serverHandle:DISCONNECT () NO-ERROR.
            DELETE OBJECT serverHandle NO-ERROR.
        END FINALLY.

    END.
END.

```

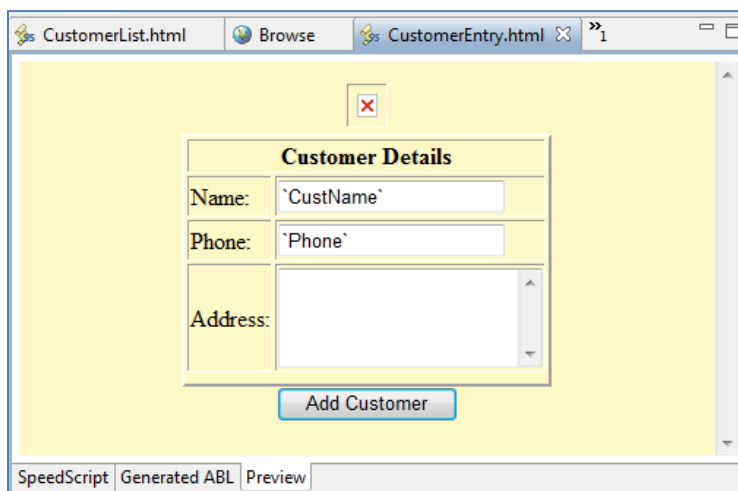


```

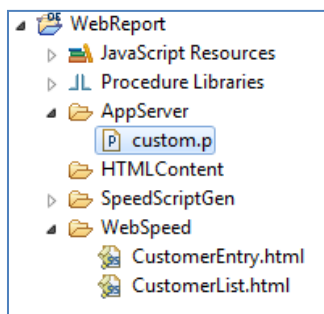
    END.
    </script>
</form>
<script language="speedscript">
/* ***** Internal Procedures ***** */
*/
</script>
</BODY>
</HTML>

```

5. Save the file and click the **Preview** tab at the bottom of the SpeedScript editor. The preview of the file is shown. The **Preview** tab will let you preview your result without running the code.



6. Copy and paste the **custom.p** business logic procedure file from the **C:\OpenEdge\WRK\PDSOEWorkshopFiles\WorkshopFiles** folder to **AppServer** folder.

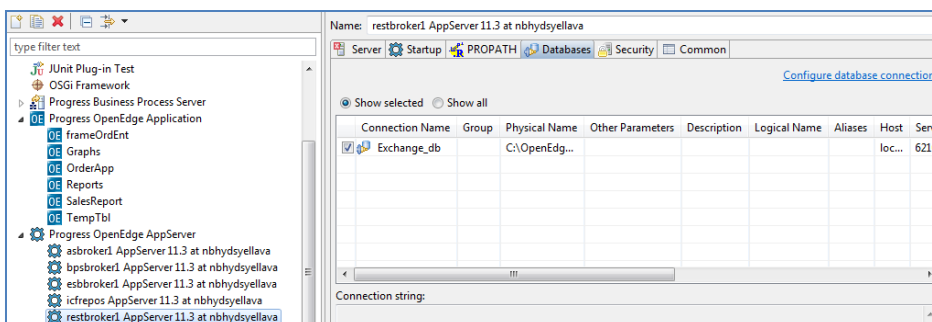


Note: Only the files that exist under the AppServer folder gets published to the server. If you want other folder files also to get published to server then those need to be included as module. You can configure it from the **Project properties Modules** dialog box.

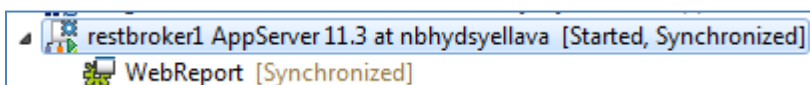
7. Open the file and observe that the procedures for Create, Delete, Read, and Update of customer records are available.



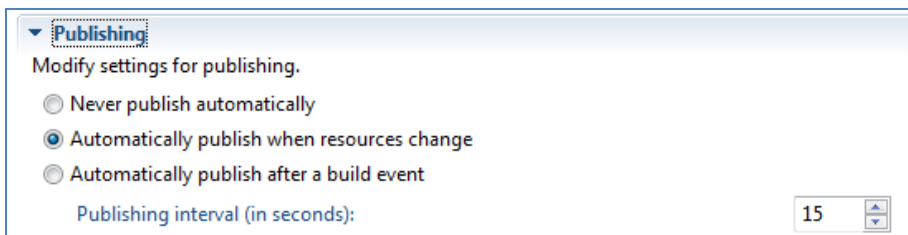
8. From the workbench menu, select **Run**→**Run Configurations**. The **Run Configurations** dialog box appears.
9. Expand **Progress OpenEdge AppServer** and select **restbroker1 AppServer**.



10. Select the **Databases** tab. Select the **Show All** option button to show all the database connections available in workspace. The **Exchange_db** connection is listed.
11. Select the check box next to **Exchange_db**. Click **Apply** and **Run** to start the server.
12. The server is started. The state of the server appears as **Started [Synchronized]**.



Note: The Publishing option automatically publishes when resources change is selected by default for server. So changes will be automatically published to the server.



13. Select the **CustomerEntry.html** tab in the editor.
14. Right-click and select **Run As**→**Run on Server**. The **Run on Server** dialog box appears. All the WebSpeed servers available are listed.
15. Select **wsbroker1 WebSpeed** server and click **Finish**. The SpeedScript file is run on wsbroker1 server and the result appears.
16. Enter the details as shown below and click **Add Customer**.



PROGRESS EXCHANGE 2013
DISCOVER. DEVELOP. DELIVER.

Customer Details

Name: Shiva

Phone: 9844378576

Address: ILabs 3rd floor, Hi-Tech City, Hyderabad

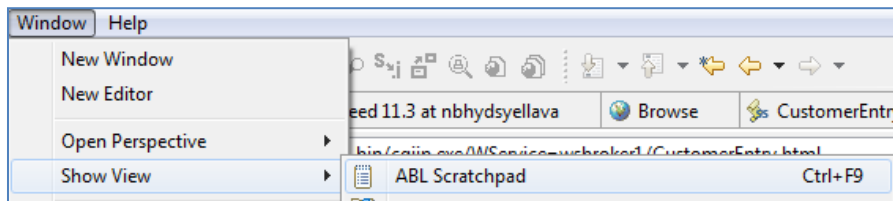
Add Customer

The CreateCustomer procedure in the custom.p file is executed in AppServer restbroker1.

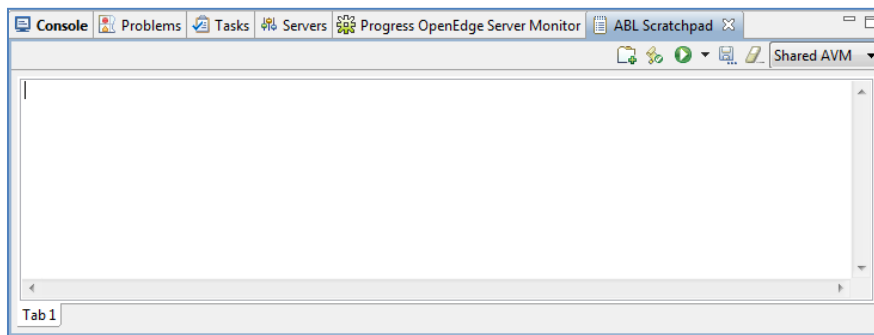
- Open the restbroker1 server log file from **C:\OpenEdge\WRK\restbroker1.server.log**. The following message appears in the log file.

```
P-014192 T-008324 1 AS -- (Procedure: 'CreateCustomer custom.p' Line:51) Customer Created
```

- To confirm that the customer record is created in database, from workbench menu, select **Window**→**Show View**→**ABL Scratchpad**.



The **ABL Scratchpad** view opens.





Tip: ABL Scratchpad is a multi-tabbed view provided by PDS OE. It allows you to write and test the ABL code without having to save it or creating a new file or project. By default, the shared AVM runtime is used for the execution of the code. However, it provides the option of selecting the runtime of any existing OpenEdge projects for execution. This saves time and avoids the need to create a project or document multiple times for testing the code in different runtimes.

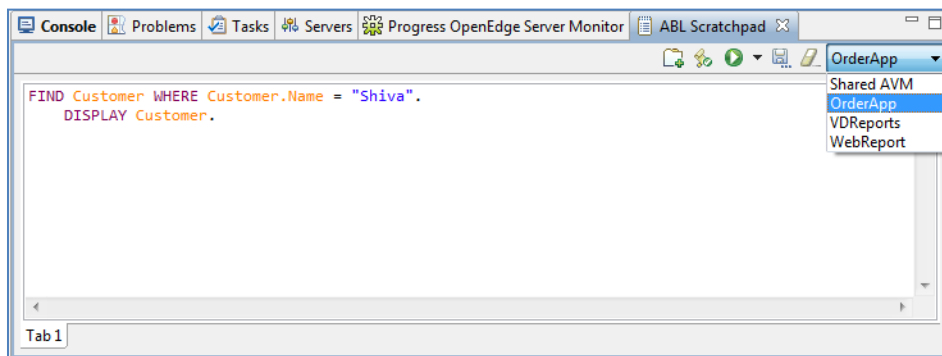
19. Copy and paste the following code in Scratchpad editor:

```
FIND Customer WHERE Customer.Name = "Shiva".
DISPLAY Customer.
```




Note: Modify the **Customer.Name** value as per the name provided in the 16th step to add a Customer.

20. Select the **OrderApp** project AVM on which this code needs to run.



Caution: If WebReport is selected, you will see a message to unhide tty window as WebSpeed project uses TTY Runtime. In the Progress OpenEdge Project Properties page, select **Hide TTY runtime console** check box to suppress the display of the project-specific runtime console by default. This setting needs to be changed to view TTY Console. This setting is changeable only if **Use shared AVM** check box is cleared and the **Use TTY for Runtime** check box is selected.

21. Click **Run** . The **Run** dialog box displays the created customer record.

Customer ID	Customer Name	Address	Phone
25	Shiva	ILabs 3rd floor,Hi-	9844378576

22. Press the **SPACEBAR** to close the **Run** dialog box.



4 LAB 04: Application development using Progress Developer Studio for OpenEdge – REST

4.1 Overview

REpresentational State Transfer (**REST**) is not a protocol; it is a set of principles or an architectural style.

RESTful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations. It is a lightweight alternative to mechanisms like RPC (Remote Procedure Calls) and Web Services (SOAP, WSDL). Every operation in REST is operated on a resource.

The sections of this lab show you how to expose the ABL resources as REST services and work with them. We will create a REST service and invoke this service using a generic REST client. We will be passing Customer id as input from the REST client and will retrieve the record of that Customer.

4.2 Prerequisites

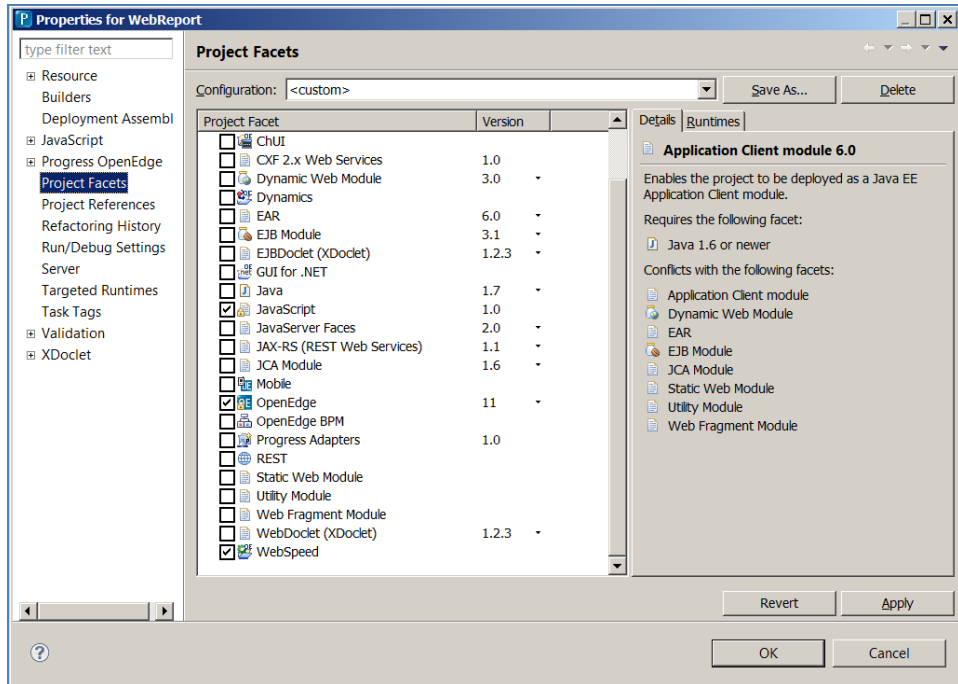
Complete **Lab 01: Configuring your workspace and customizing the project** and **Lab 03: Application development using Progress Developer Studio for OpenEdge – WebSpeed** prior to working on this lab.

4.3 Adding a REST facet to the WebSpeed project


This section shows you how to add a new REST facet to an existing WebSpeed project. We will use the same resources from the WebReport project that we created in the above section.

1. Select the **WebReport** WebSpeed project in the **Project Explorer** view.
2. Right-click and select **Properties**. The **Properties for WebReport** dialog box appears.
3. Select the **Project Facets** node.





4. Select the check box next to **REST** to add a facet to the project. The following error message appears:

 **REST requires Progress Adapters 1.0.**

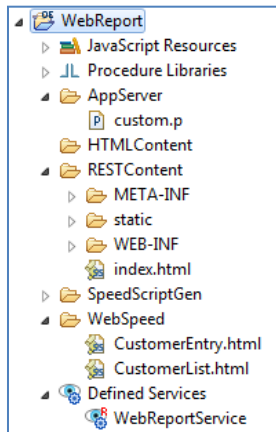


Note: REST facet is dependent on AppServer and Progress Adapters facets. As AppServer facet is already selected, the error displayed is only for Progress Adapters.

5. Select the check box next to **Progress Adapters**. It clears the error message.
6. Click **Apply** to save the changes.
7. Click **OK** to close **Properties** dialog box. The REST facet is added and appears in the **WebReport** project.



You will notice **WebReportService** REST service created under the **Defined Services** node. Whenever you create a **REST** project, a default REST service is created.

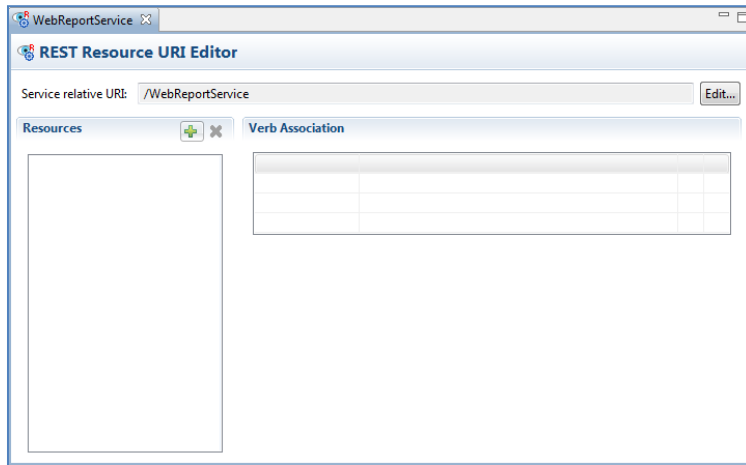



4.4 Working with the REST Editor – Mapping ABL operations with HTTP verbs

This section shows you how to map the HTTP verbs with ABL operations.

1. Select the **Defined Services** node under **WebReport** project in **Project Explorer** view.
2. Double-click the **WebReportService** REST. The **REST Resource URI Editor** dialog box opens.

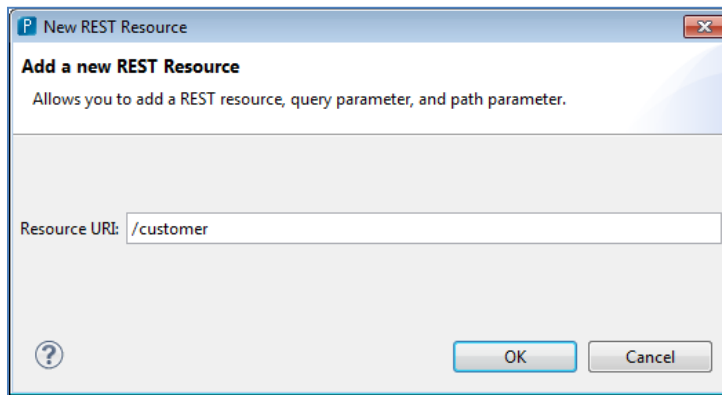




3. Click **Add Resource**  in the **Resources** category. The **New REST Resource** dialog box appears.

As mentioned before, all REST operations work on top of a resource, so we are here going to create a resource using which we can operate the service.

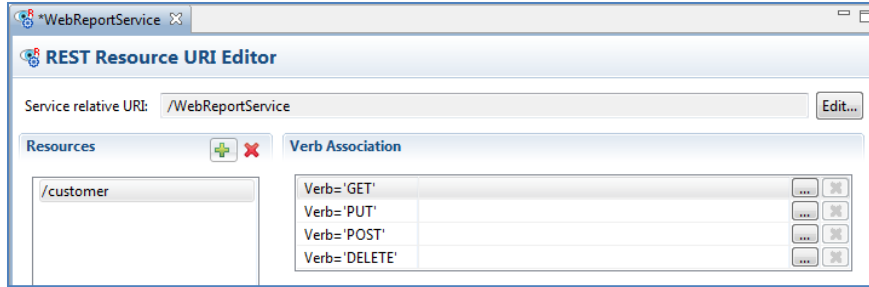
4. Enter **/customer** for the resource in **Resource URI**. The REST resources you add are identified with Uniform Resource Identifier (URI). The name must start with "/".




5. Click **OK**. The **customer** resource is added.

The **Verb Association** table shows the available HTTP verbs.

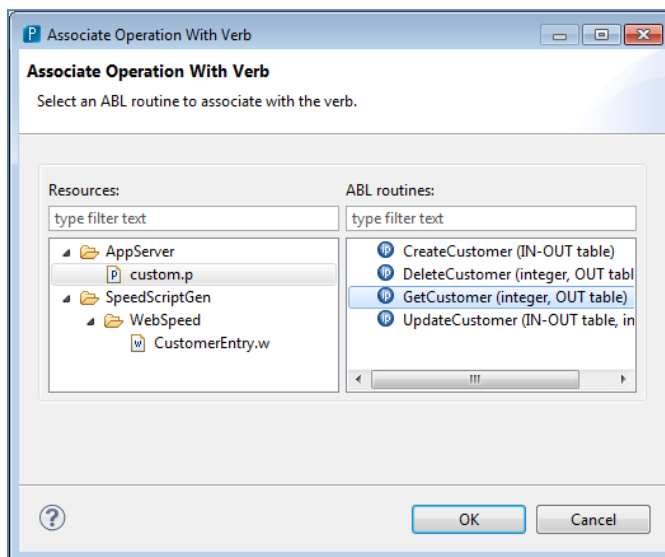




6. Select  next to **Verb='GET'** to associate GET verb to the **GetCustomer** internal procedure to execute the ABL logic when GET request is processed.

The **Associate Operation With Verb** dialog box appears and lists all the available ABL files in the current project.

7. Select **custom.p** resource and **GetCustomer** ABL routine. The **custom.p** file has all the CRUD operations/methods defined and we are selecting **GetCustomer** ABL routine which is a read operation to retrieve the records.



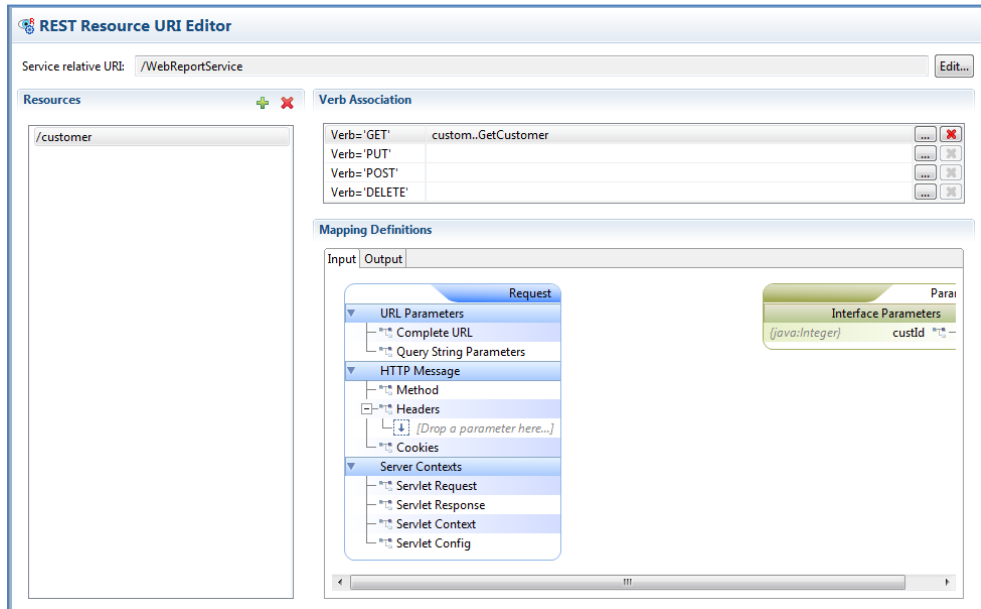
8. Click **OK** to complete the association.



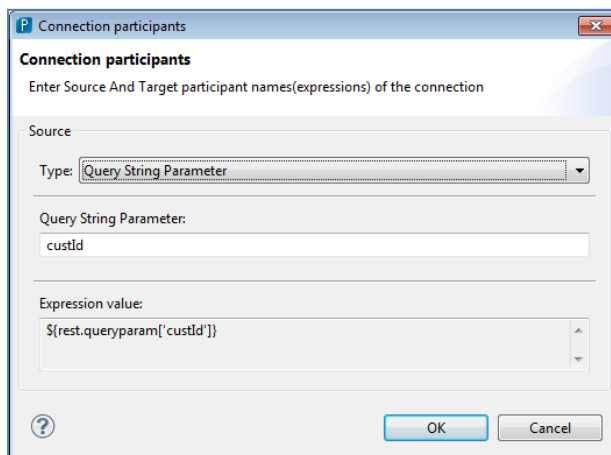
Note: Open **custom.p** file and notice that the REST annotations were automatically added at the file level and to the **GetCustomer** routine that was selected. Close to proceed to REST Resource URI Editor.

We will fetch records based on the Customer id. So we now need to map customer id as an input parameter to an URI parameter as query strings.

9. Double-click the **WebReportService** tab to maximize and view it in full screen. The **Mapping Definitions** shows **Input** and **Output** tabs with the available request mappers. The input parameter **custId** that was defined in the procedure will be listed in the **Parameter** section.

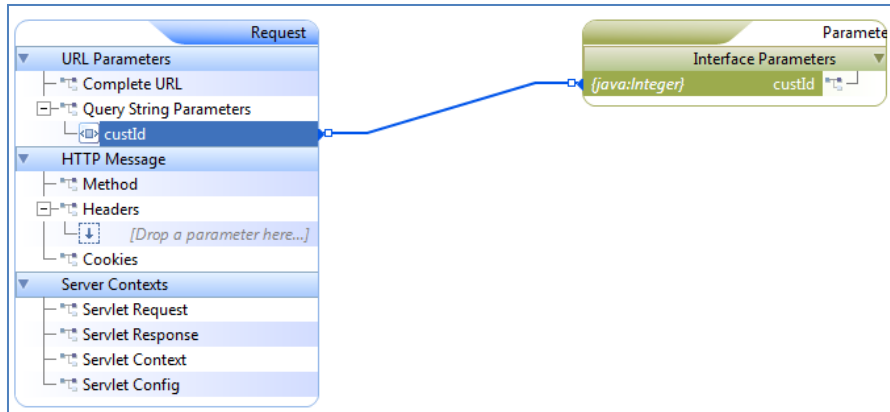


10. Place the mouse cursor on **Query String Parameters** and drag and drop the control to **custId**. The **Connection participants** dialog box appears.
11. Click **OK**.

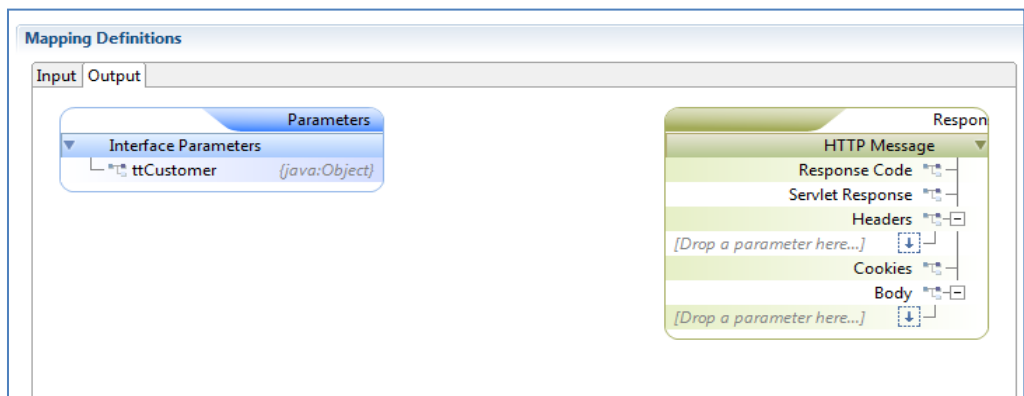


The input parameter mapping is complete and **custId** query string parameter is added and linked to the input parameter as displayed:

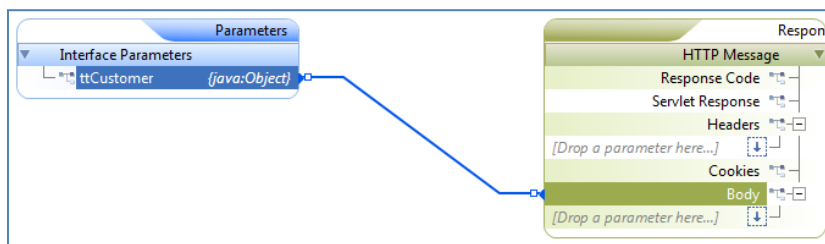




12. Select the **Output** tab.



13. Select **ttCustomer**, drag and drop to **Body** in the **Response** section.



4.5 Publish and invoke the REST service

We have completed creating the REST service and mapping the parameters with REST verbs.

This section shows you how to publish the REST service to OE Web Server and invoke the REST service from the REST client.

1. Select the **Servers** view. The **restmgr1 OE Web Server** which is the default Web Server for REST applications appears in the **Servers** view.





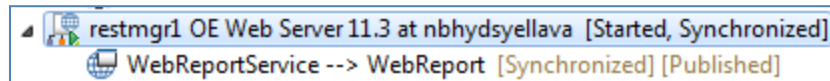
Note: Progress Developer Studio for OpenEdge now ships Tomcat which is the default web server for REST and Mobile applications.

2. We now have to associate the REST service to the OE Web Server. Right-click the **restmgr1 OE Web Server** and select **Add and Remove** from the context menu.
3. Add the **WebReportService** using **Add and Remove...** to the **restmgr1 OE Web Server**.

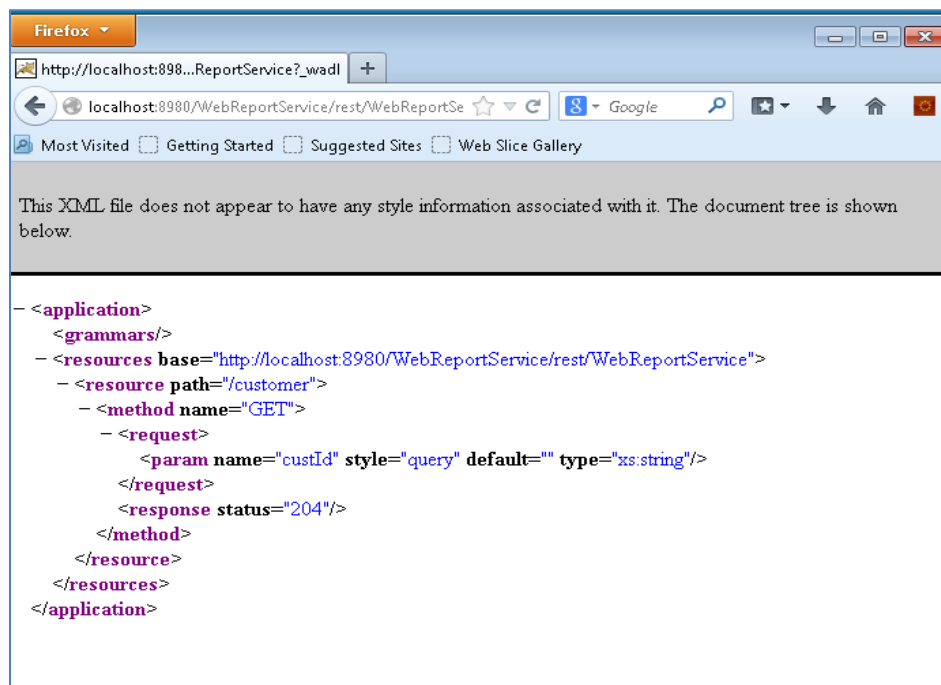



4. Select **restmgr1 OE Web Server** and click **Start the server** .

The server is started and the status changes to **Started**, **Synchronized** and **WebReportService** is published to the server:



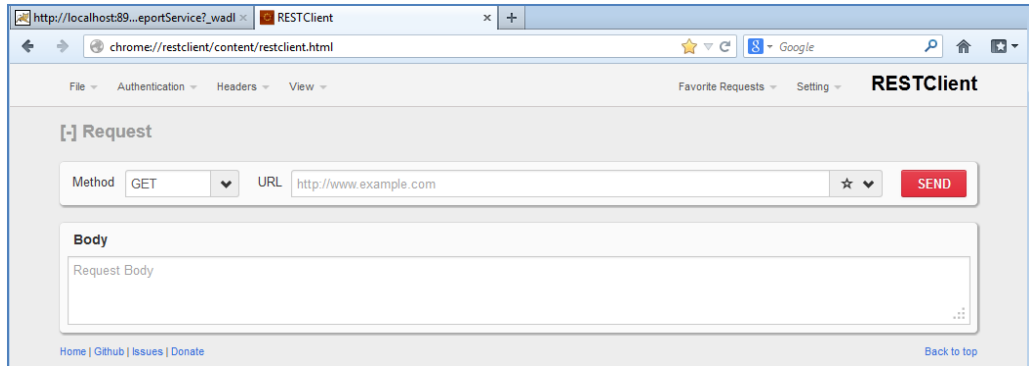
5. Open the **WADL** file to know the base URI for the REST service. In the Firefox browser, open the URL: http://localhost:8980/WebReportService/rest/WebReportService?_wadl. The WADL displays Base URI, operations, and parameter mapping information.



 **Note:** **Web Application Description Language (WADL)** provides complete information on the REST service in terms of resources and operations. This is similar to **Web Service Description Language (WSDL)** in Web Services.

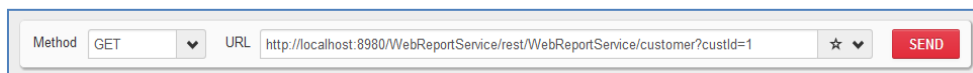
6. In a new tab in Firefox, open the **chrome://restclient/content/restclient.html** URL. The **REST Client** opens.





Note: We have already installed REST client in the system provided. In real-time, you need to install a generic REST client to test the service deployed.

- Copy the **Base URI** <http://localhost:8980/WebReportService/rest/WebReportService> from the WADL and paste it in **URL** field in the REST client. Append **/customer?custId=1** which is the resource that we are accessing and the parameter that we are passing as a query string:



- Click **SEND**. Select the **Response Body (Highlight)** tab. The **GetCustomer** routine gets executed in the AppServer and the results corresponding to the Customer with id=1 is displayed in the client.



4.6 Working with other HTTP verbs (Take-Home)

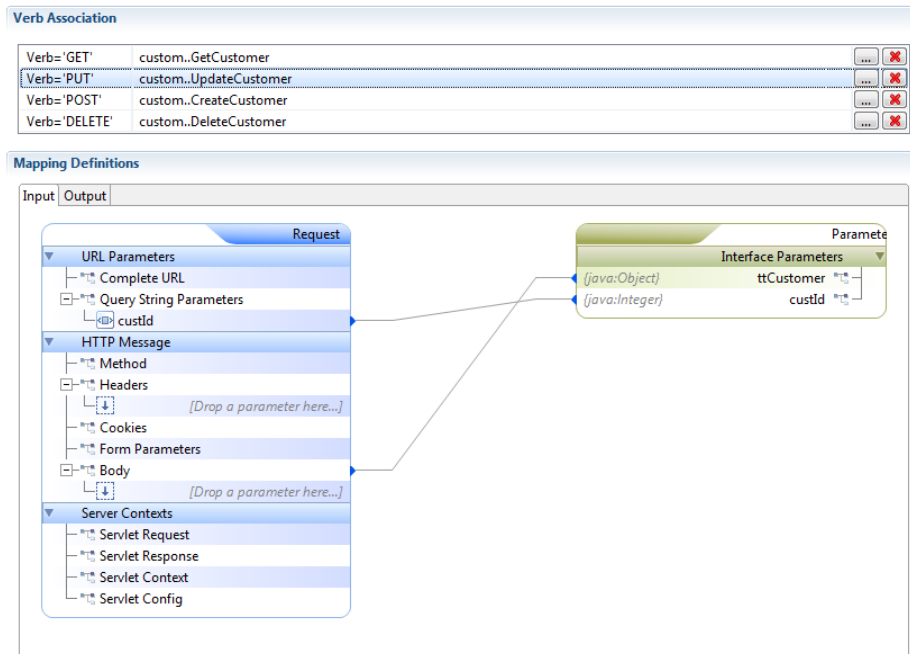
This section is a take-home section for you to work on other VERBS at your leisure. It is not mandatory to perform this section in this workshop.



1. Follow steps from above section to add the PUT, POST, DELETE verbs with ABL routines as shown in the screens below.

Verb Association		
Verb='GET'	custom..GetCustomer	...
Verb='PUT'	custom..UpdateCustomer	...
Verb='POST'	custom..CreateCustomer	...
Verb='DELETE'	custom..DeleteCustomer	...

2. For **PUT** verb **Input** mapping, map **Query String Parameter** to **custId** and **Body** to **ttCustomer**.



3. For **PUT** verb **Output** mapping, map **ttCustomer** to **Body**.
4. For **POST** verb **Input** mapping, map **Body** to **ttCustomer**.



Verb Association

Verb='GET'	custom..GetCustomer	...	✖
Verb='PUT'	custom..UpdateCustomer	...	✖
Verb='POST'	custom..CreateCustomer	...	✖
Verb='DELETE'	custom..DeleteCustomer	...	✖

Mapping Definitions

Input | Output

Request

- URL Parameters
 - Complete URL
 - Query String Parameters
- HTTP Message
 - Method
 - Headers [Drop a parameter here...]
 - Cookies
 - Form Parameters
 - Body [Drop a parameter here...]
- Server Contexts
 - Servlet Request
 - Servlet Response
 - Servlet Context
 - Servlet Config

Parameter

- Interface Parameters
 - ttCustomer (java:Object)

- For **POST** verb **Output** mapping, map **ttCustomer** to **Body**
- For **DELETE** verb **Input** mapping, map **Query Stiring Parameter** to **custId**.

Verb Association

Verb='GET'	custom..GetCustomer	...	✖
Verb='PUT'	custom..UpdateCustomer	...	✖
Verb='POST'	custom..CreateCustomer	...	✖
Verb='DELETE'	custom..DeleteCustomer	...	✖

Mapping Definitions

Input | Output

Request

- URL Parameters
 - Complete URL
 - Query String Parameters
 - custId
- HTTP Message
 - Method
 - Headers [Drop a parameter here...]
 - Cookies
 - Body [Drop a parameter here...]
- Server Contexts
 - Servlet Request
 - Servlet Response
 - Servlet Context
 - Servlet Config

Parameter

- Interface Parameters
 - custId (java:Integer)

- For **DELETE** verb **Output** mapping, map **ttCustomer** to **Body**.

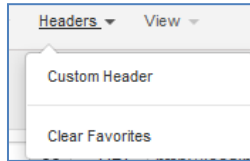
Click **File**→**Save All** to save all changes.

4.7 Invoke the REST service for remaining HTTP verbs (Take-home)



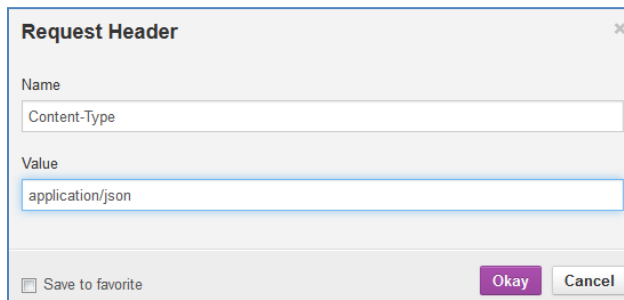
You need to complete the section 4.6 prior to working on this lab.

1. For POST verb request, the data is sent via JSON format. We need to add two custom headers to accept the JSON content. From **Headers** in the REST client, select **Custom Header**.

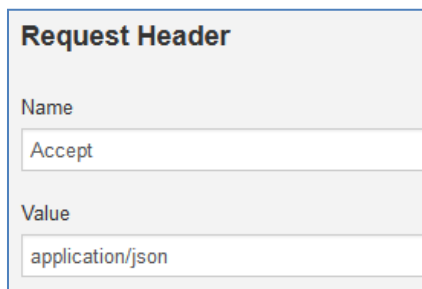


The **Request Header** dialog box appears.

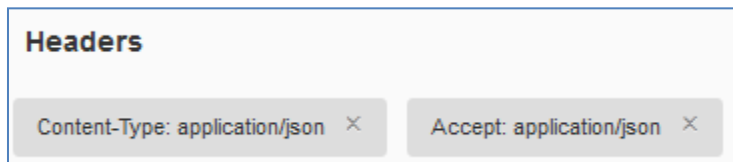
2. Enter **Content-Type** as header name and **application/json** as value and click **Okay**.



3. Similarly, repeat the above steps to add **Accept** as another **Custom header**.



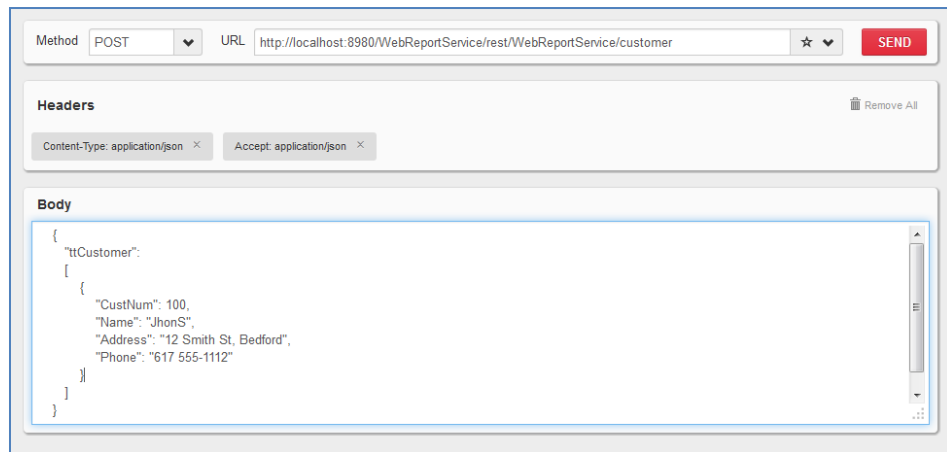
The **Headers** section displays the **Content-Type** and **Accept** headers.



4. Select the **POST METHOD** and provide <http://localhost:8980/WebReportService/rest/WebReportService/customer> URL. Copy and paste this JSON code in **Body** of the REST client:



```
{
  "ttCustomer":
  [
    {
      "CustNum": 100,
      "Name": "JhonS",
      "Address": "12 Smith St, Bedford",
      "Phone": "617 555-1112"
    }
  ]
}
```



5. We are sending a record in JSON format which will be inserted into database. Click **SEND**.
6. It executes the **CreateCustomer** routine mapped for the **POST** method. The customer record is created. As we mapped the **POST** method **OUTPUT ttCustomer** to **Response**, the same JSON record is displayed in the REST client.

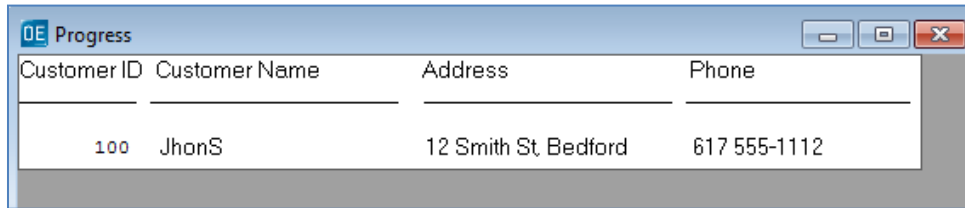


7. Use the **ABL Scratchpad** editor to check if the new record is created in the customer table. Execute the following code:

```
FIND LAST Customer WHERE Customer.Name = "JhonS".
DISPLAY Customer.
```



The result displays as:



The screenshot shows a window titled "Progress" with a table containing one row of data. The table has four columns: Customer ID, Customer Name, Address, and Phone.

Customer ID	Customer Name	Address	Phone
100	JhonS	12 Smith St, Bedford	617 555-1112

8. Press **SPACE BAR** to close the **Run** window.



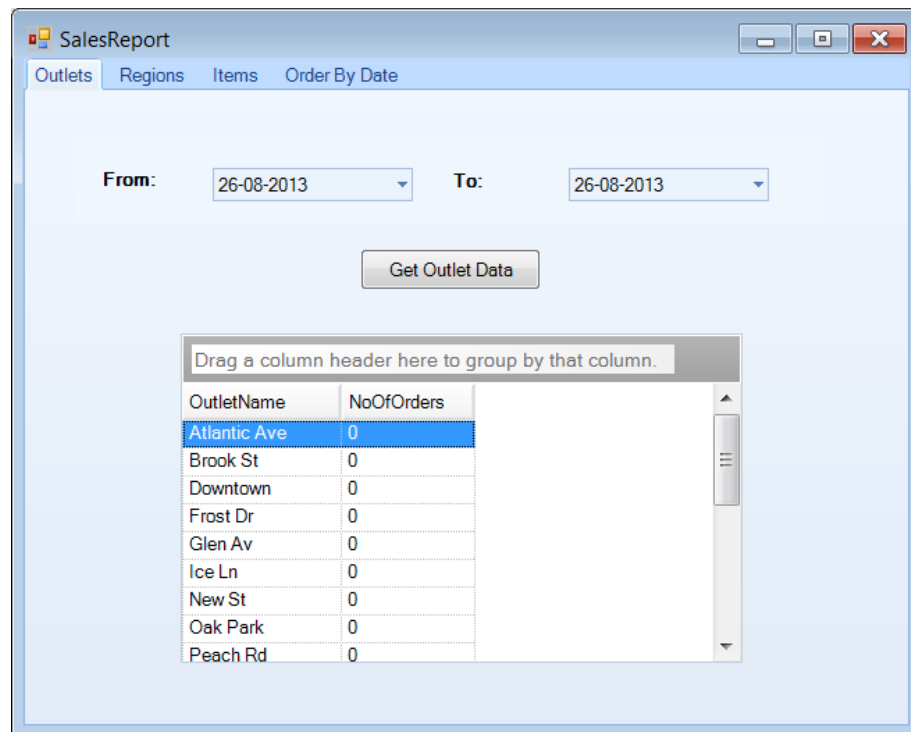
5 LAB 05: Application development using Progress Developer Studio for OpenEdge – GUI for .NET

5.1 Overview

The sections of this lab show how to design, develop, and run the ABL application using the GUI for .NET or OpenEdge Visual Designer as it is commonly called. With GUI for .NET, you can develop powerful graphical user interfaces (GUIs) for ABL applications. Visual Designer users can take advantage of the rich functionality and the look and feel of .NET controls without using any non-ABL language or leaving the Progress Developer Studio for OpenEdge environment.

In this lab, you will design a Sales report application to view Sales report in a given period. You will also design and use custom controls such as User controls and Inherited controls as part of the flow. In the process, you will understand many features that the editor provides to make development easy and productive.

Here is what you will develop in this lab.



5.2 Prerequisite

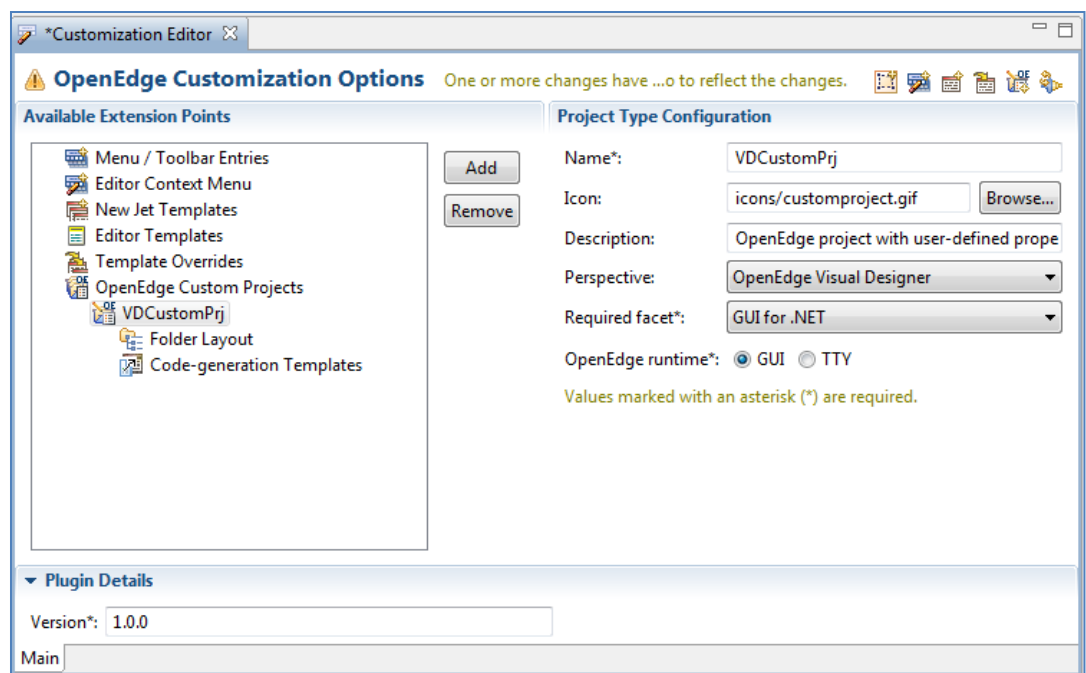
Complete **Lab 01: Configuring your workspace and customizing the project** prior to working on this lab.



5.3 Creating Custom Project

This section shows you how to create a custom project.

1. From the workbench menu, select **Window**→**Open Perspective**→**OpenEdge**. The **OpenEdge Editor** perspective opens.
2. From the workbench menu, select **OpenEdge**→**Tools**→**Customization Editor**. The **Customization Editor** opens.
3. The **OpenEdge Custom Projects** node allows defining custom projects. Select the **OpenEdge Custom Projects** node and click **Add**.
4. Enter **VDCustomPrj** in **Name**, **OpenEdge Visual Designer** in **Perspective**, **GUI for .NET** in **Required facet**.

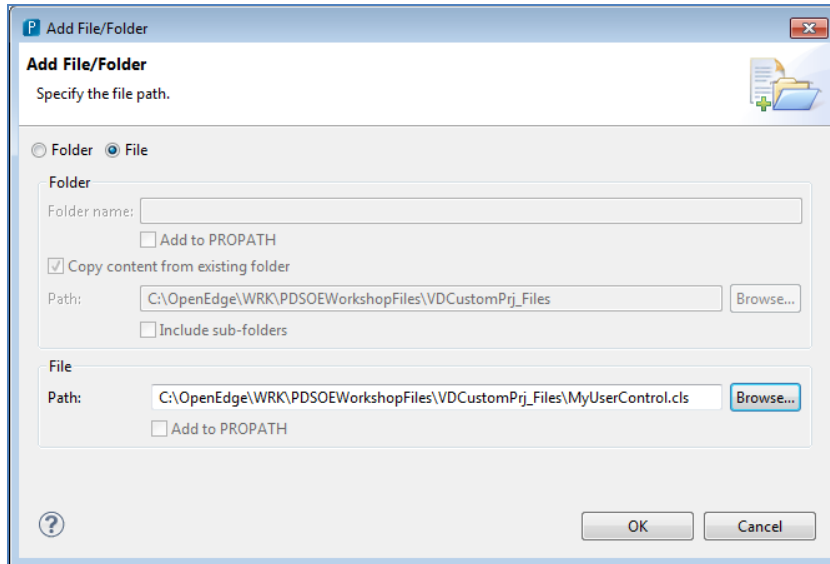


5. Select **Folder Layout** node under the custom project **VDCustomPrj**. The **Folder Layout** and **PROPATH** sections are displayed on the right pane of the Customization Editor.
6. Click **Add** in the **Folder Layout** section. This allows creating a file or folder using the **Add File/Folder** dialog box.
7. Select **File** option button. Click **Browse...** next to **Path**.

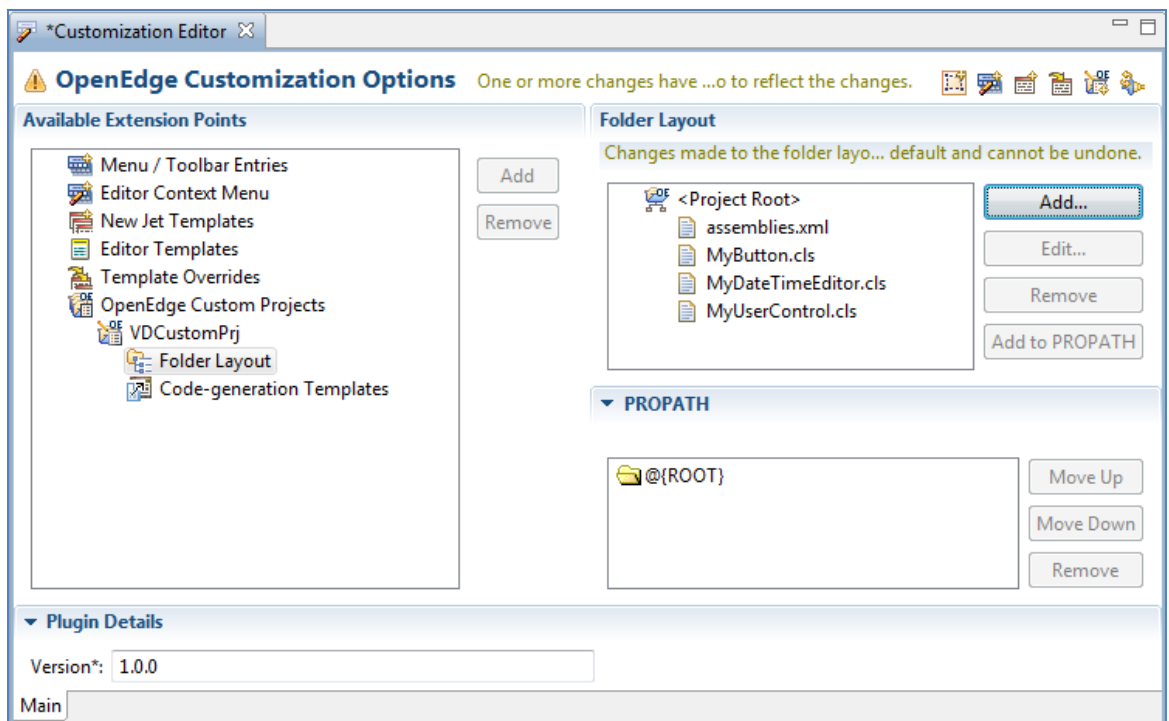


8. Browse for path **C:\OpenEdge\WRK\PDSOEWorkshopFiles\VDCustomPrj_Files** and select the **MyUserControl.cls** file.

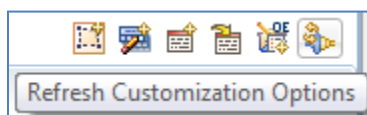




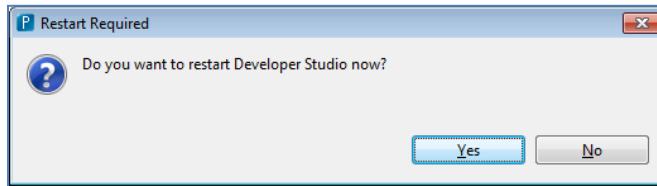
- Click **OK**. The file is added to the Folder Layout. Add all files under the **VDCustomPrj_Files** folder to **Folder Layout** section.



- Save changes and click **Refresh Customization Options** toolbar option to apply the changes.

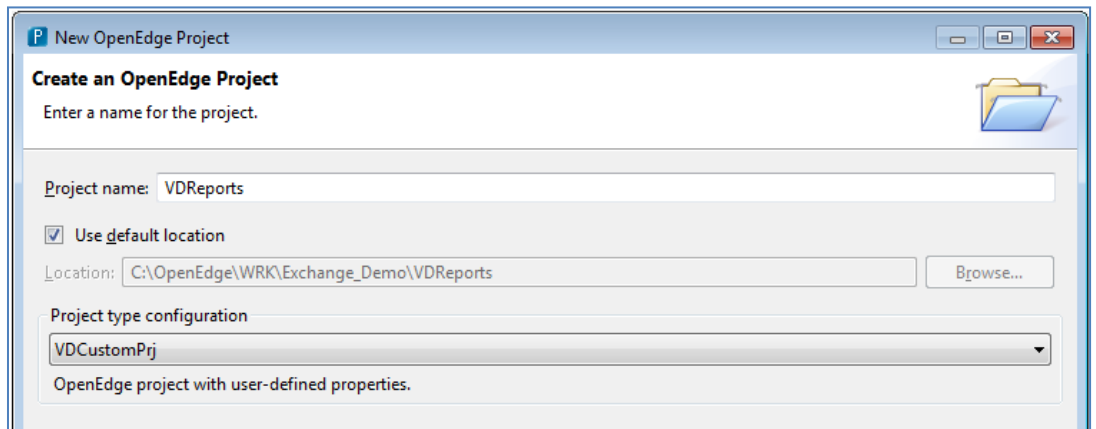


11. Once a new custom project is created, workbench needs to be restarted to list the created project type in OpenEdge Project Creation window. Click **Yes** in the **Restart Required** dialog box.

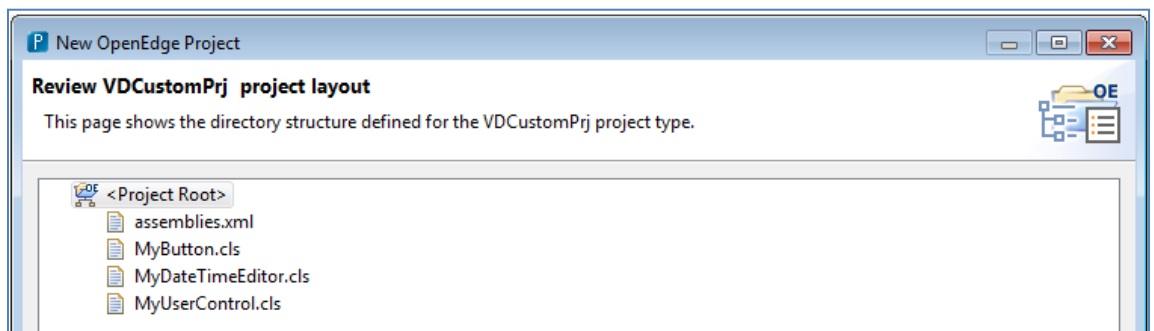


The workbench is restarted and **Workspace Launcher** dialog appears.

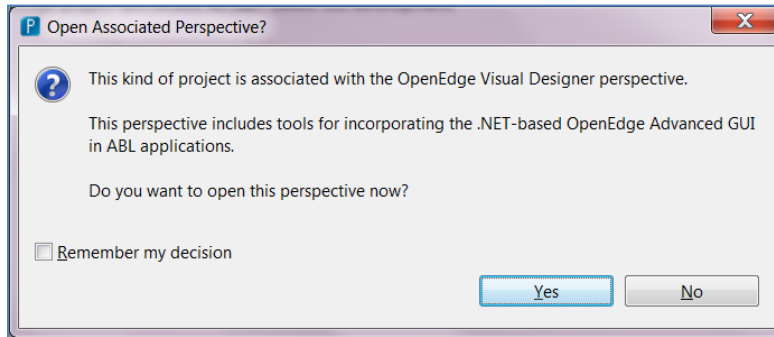
12. Click **OK** to open workbench.
13. From the workbench menu, select **File→New→OpenEdge Project**. The **New OpenEdge Project** dialog box appears.
14. Specify **VDR** in **Project name**.
15. Observe custom project is created **VDCustomPrj** and listed in **Project type configuration**. Select **VDCustomPrj** from the **Project type configuration** drop-down list.



16. Click **Next**. The **Select AVM and layout options** dialog box appears.
17. Click **Next**. The **Review VDCustomPrj project layout** dialog box appears. Expand **Project Root** and observe the folder layout added is listed.



18. Click **Finish**. Select **Yes To All** in the **Overwrite** dialog box. The GUI for .NET project by default creates **assemblies.xml** file and as part of custom project folder layout, we are adding this file as the default and therefore you need to overwrite the files.
19. As this project is associated with the **OpenEdge Visual Designer** perspective, the **Open Associated Perspective** dialog box appears.

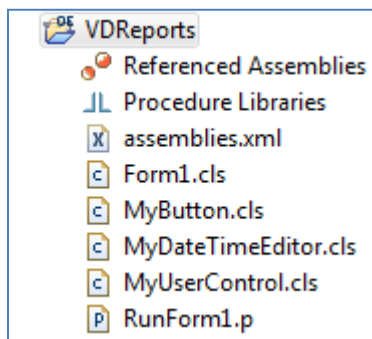


Note: The projects are associated with a specific perspective. You should open the views as suggested. Views are arranged to make it easy for development. **Project Explorer**-> **New** shows the file options that are related to this project development so that you can find options easily.

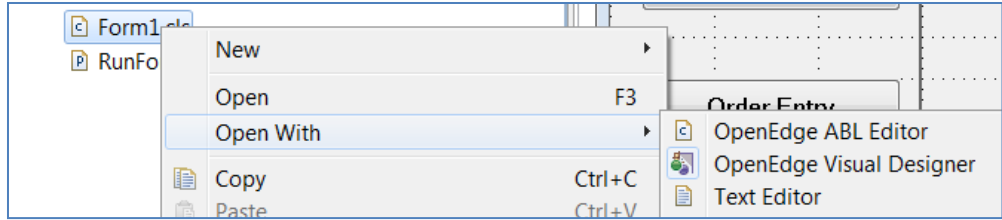


Tip: If you click the **Remember my decision** check box, the next time when you create a new project of this type, the decision is remembered and the required action is performed. For example, if you selected **Yes**, it opens a perspective or switches to this perspective whenever this configuration type project is created.

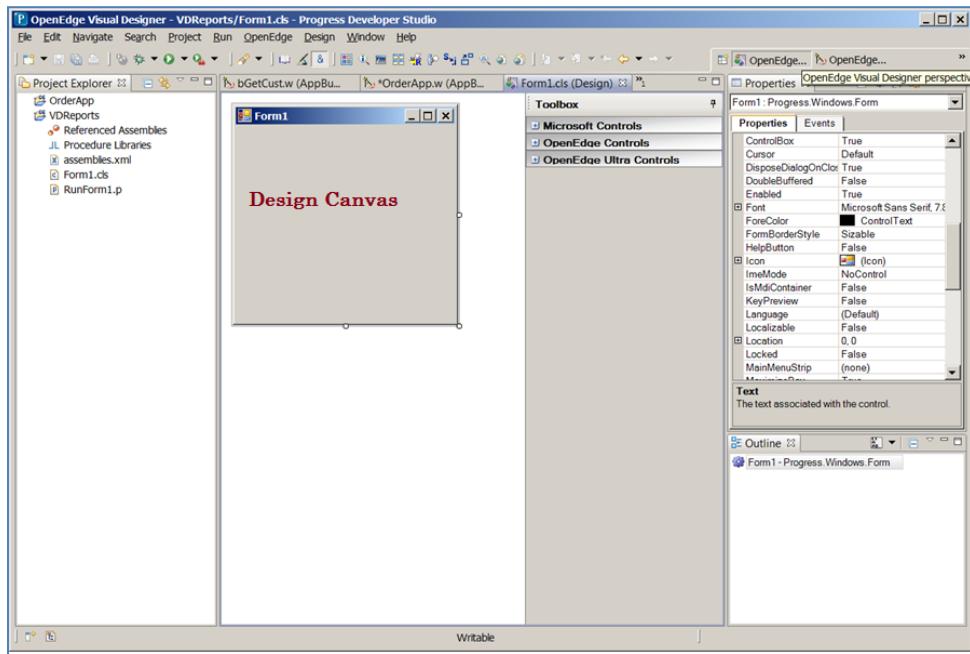
20. Select the **Remember my decision** check box and click **Yes**. The **Visual Designer** perspective opens, the **VDReports** project is created, and the folder is available in the **Project Explorer** view with all the files.




21. By default, an ABL form is created under the project. Right-click the **Form1.cls** file and select **Open With**→**OpenEdge Visual Designer**.



The form opens in the design view. The **Visual Designer** perspective has a wide area for designing form (editor area) where toolbox is on the right side of the screen. The **Project Explorer**, **Properties**, and **Outline** views are displayed. The design workbench menu option is available where options related to forms designing are available.



 **Note:** The **Design** menu option is visible only when the Visual Designer files are opened in editor. The options are specific to design view of the files. When any other file is opened in editor, this menu option is not visible. Observe this by selecting the OrderApp file.

 **Note:** The major components of the Visual Designer are the Toolbox, the Design Canvas, and the Properties views.

Toolbox

The Toolbox is a list of controls that are available for use in the UI. You can select controls that you want to place on the form from this list. You can customize the set of controls available to the current project, and arrange them in logical groups that you define.

Design Canvas

The Design Canvas is the editing area in which you model the appearance of the UI. It provides an accurate visualization of how the window appears to the user at run time, with the exception that it does not display the actual data. You use the mouse to size objects by dragging their borders, and to position controls on the form.



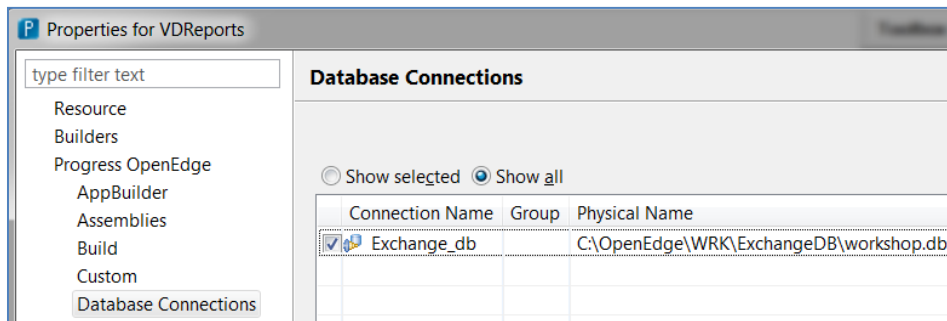
Properties view

While a Visual Designer editing window has focus, the Properties view includes both Properties tab and Events tab, where you can view and edit the visual and behavioral characteristics of the object or objects that are selected on the Design Canvas.

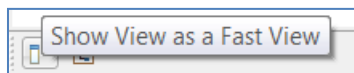
5.4 Assigning a database connection to a project

This section shows you how to assign database connection to the VDRreports project.

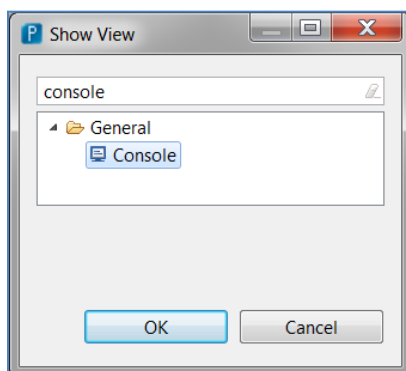
1. Select the **VDRreports** project in the **Project Explorer** view.
2. Right-click **VDRreports** and select **Properties**. The **Properties** dialog box appears.
3. Select **Progress OpenEdge**→**Database Connections**. The defined database connection for the workspace is displayed.
4. Select the check box next to **Exchange_db**.



5. Click **OK**. The **Properties** dialog box closes and the connection is made by the AVM associated with project **VDRreports** to the **workshop.db** database using the database connection profile. The splash screen displays briefly as the AVM is started.
6. Select **Show View as a Fast View** at bottom left corner of PDS OE to open the Console view.

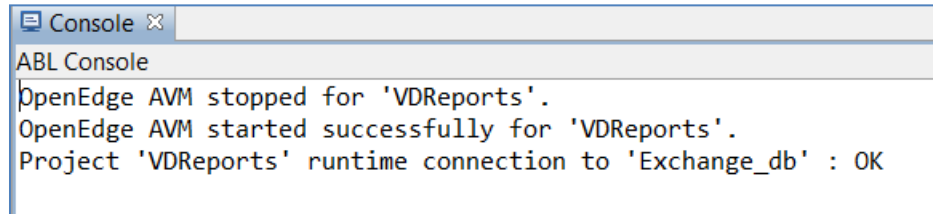


7. Select **Other**. The **Show view** dialog box appears. The textbox displays **type filter text**. Enter the filter text as **Console**.



8. Select **Console** and click **OK**. This opens the **Console** view in the **Visual designer** perspective.

The **Console** view shows messages that the runtime was started and that the connection to the `Exchange_db` database succeeded.



```
Console x
ABL Console
OpenEdge AVM stopped for 'VDReports'.
OpenEdge AVM started successfully for 'VDReports'.
Project 'VDReports' runtime connection to 'Exchange_db' : OK
```

5.5 Creating and designing a form

This section shows you how to create a form, add controls, and design a form.

1. Select project **VDReports** in the **Project Explorer** view. Right-click and select **New**→**ABL Form**. The **New ABL Form** wizard for form creation appears.
2. Enter **SalesReport** in **Form name**.



 **Note:** The **New Form** wizard provides many options.

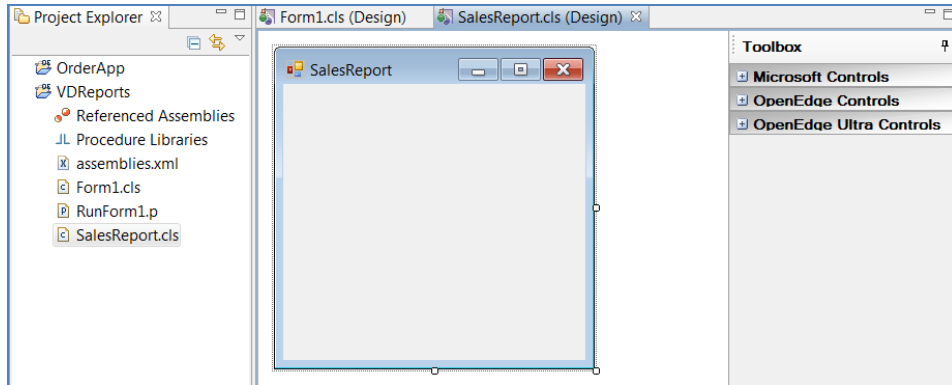
Package root: Specifies a currently open project to contain the class code and other project code. Click **Browse** if you want to select a project other than the current one (the default value).

Inherits: Optionally specifies another class in the current project as a super class from which the new class inherits state and behavior.

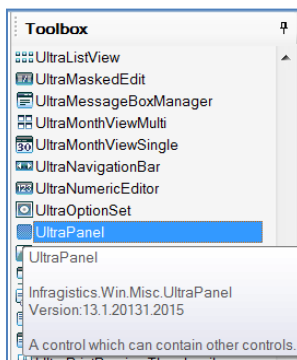
Implements: Optionally specifies one or more interfaces in the current project that the class implements. Click **Add** and select the desired interfaces in the **Interface Selection** dialog box. Use the **Remove** button to remove an interface from the list after adding it.

3. Click **Finish**. The **New ABL Form** wizard creates a blank form. The form opens in the Design View in editor and the respective **SalesReport.cls** file is created under the **VDRports** project.



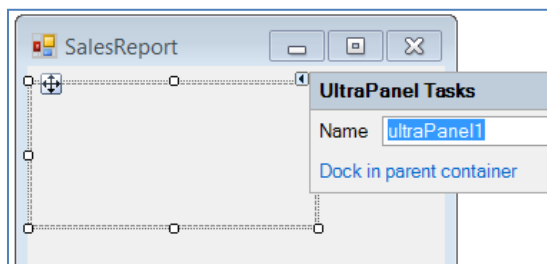


- Expand the **OpenEdge Ultra Controls** category in **Toolbox** and select **UltraPanel** control.



Tip: If you place cursor over the control in ToolBox, a brief help on the control class name, version, and usage of control is displayed. OpenEdge Ultra Controls are not shipped with Progress Developer Studio for OpenEdge and are sold as a separate product. You can buy it directly from Infragistics.

- Drag and drop the **UltraPanel** control on the **SalesReport** form. The **UltraPanel** control is added to form. Select **SmartTag**, the small right-arrow as shown in screen:



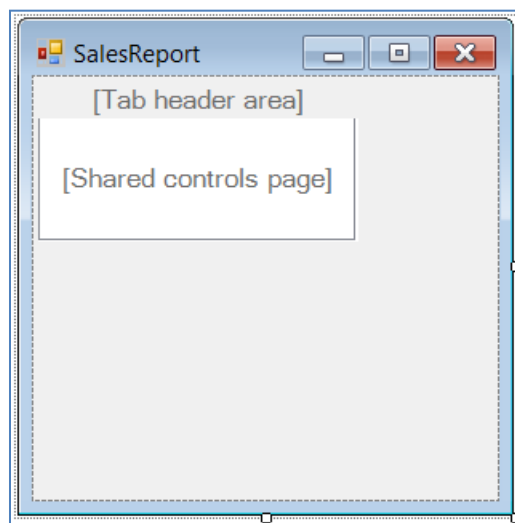
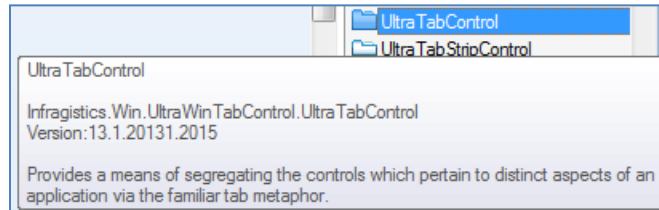
Note: UltraPanel is a control which can contain other controls and provide automatic scrolling or sizing. This control provides almost the same functionality as the .NET Panel control in the Microsoft controls. In addition, it also allows advanced appearance customization of the scroll bars and UltraPanel itself, including the ability to use application styling.




Tip: SmartTags offer a subset of the most frequently used properties for the control. If SmartTags are available, the control has a small right-arrow button at the top right corner when the control is selected.



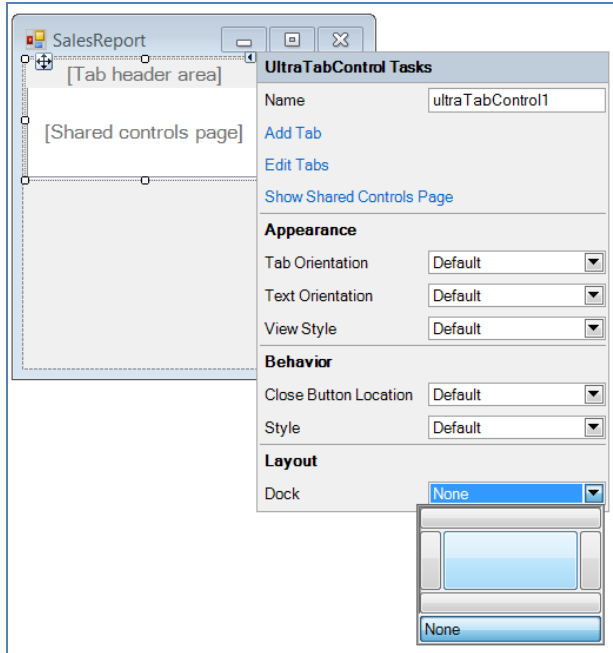
6. Select the **Dock in Parent container** link. Observe that the panel expands and adjusts in the space of the form.
7. Double click **UltraTabControl** in **ToolBox**. The **UltraTabControl** is added to **SalesReport** form in default area. **UltraTabControl** gives you the ability to add tabs to your window.



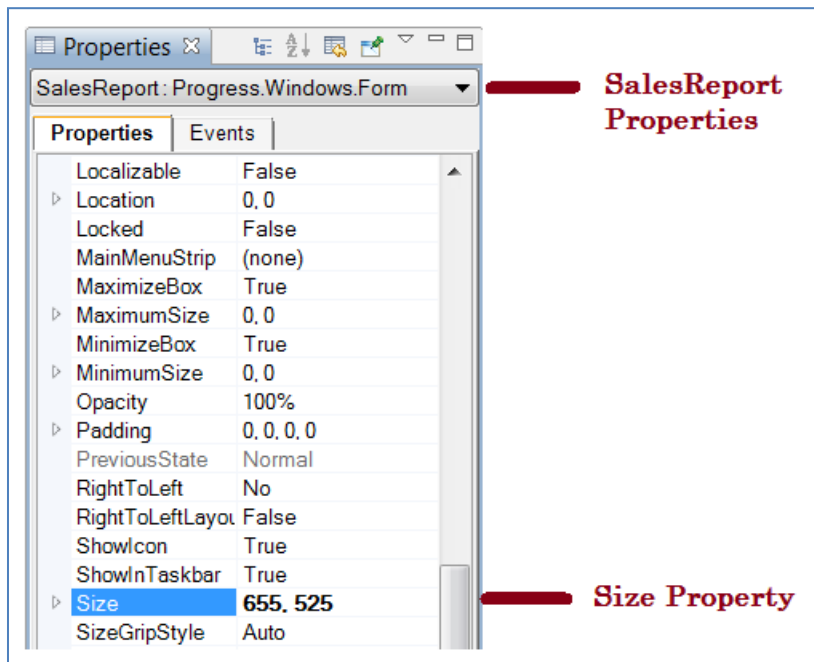
 **Tip:** If you double-click a control in ToolBox, it adds the control to the default area in the default size. If control needs to be added to a specific area with a specific size, hold down the primary mouse button, and drag the control to the desired position on the form.

8. Open **SmartTag** of **UltraTab** in the form and select **Fill** in the **Dock** property. Observe that Tab control is filled in form.





9. Select the title area of the **SalesReport** window in the Design View. The **Properties** view shows the properties of the selected form.
10. Enter **655, 525** in the **Size** property and press **Enter** to apply changes.

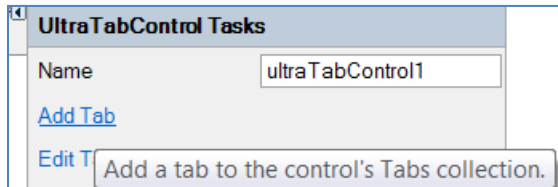


Caution: Once property value is modified in the **Properties** tab, the changes are viewed only when the cursor is placed outside the property value field. Press enter or click anywhere in the **Properties** view or **Design Canvas** to view the changes.

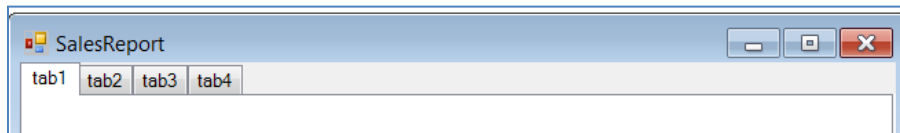


Observe that the form size has also increased. Also, sizes of UltraTab and UltraPanel are increased as they are docked to the form.

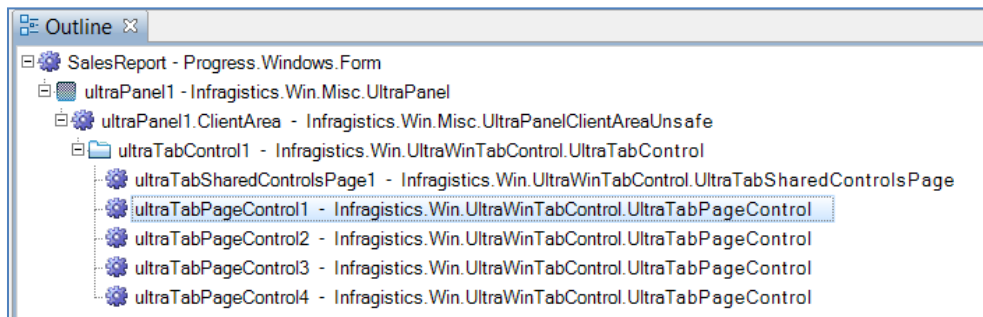
11. Click **File**→**Save** to save the changes.
12. Select **Tab header area**. The **SmartTag** of UltraTab appears.
13. Click **SmartTag** of UltraTab. Observe that the **Add Tab** link is available.



14. Click the **Add Tab** link. **Tab1** is added. Similarly, add 3 more tabs.



15. Select **ultraTabPageControl1** in the **Outline** view. The **Property** view shows the properties of **ultraTabPageControl1**.



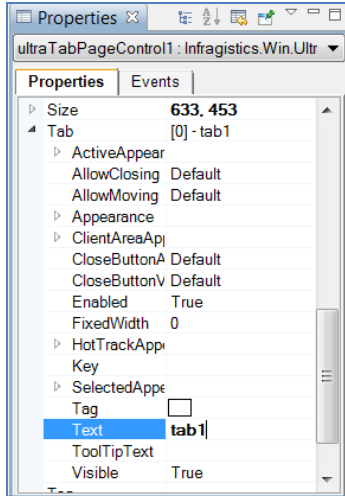
Tip: The **Outline** view of a file that is open in the Visual Designer is called as **Document Outline** view, it shows the contents of the Design Canvas and provides an alternative means of selecting, copying, deleting, renaming, and re-parenting controls.

The **Document Outline** view shows a collapsible-expandable hierarchy of the top-level container (form or user control) and all of its controls in a tree structure. Controls appear in the **Document Outline** view in the order in which you place them on the Design Canvas.

When there are many controls added and its hard to select a control in Design View, control selection can be easily done from the **Document Outline** view.

16. From the **Properties** view, expand the **Tab** property and notice that the **Text** Property is available. This property represents the caption text for the tab.

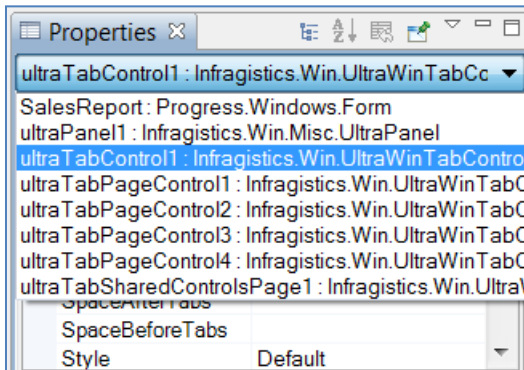




17. Enter value for the **Text** property as specified in table below and press **enter** after each value is entered in the **Properties** view. The tabs will now show the specified text.

Tab Page Control	Text Property Value
ultraTabPageControl1	Outlets
ultraTabPageControl2	Regions
ultraTabPageControl3	Items
ultraTabPageControl4	Order By Date

18. Select the combo-box in the **Properties** view and select **ultraTabControl1**.



19. Select the following values for the properties of **ultraTabControl1**:

Property	Value	
View Style	Office 2007	ViewStyle Office2007
Style	Office 2007 Ribbon	Style Office2007Ribbon

20. Observe that the look of the form has changed. Save the changes.



5.6 Creating Inherited controls (Take-home)


We have already provided the Inherited controls that you can use in this workshop. This section shows you how to create the same Inherited controls, you can try it at your leisure. When you are doing this section, delete already created Inherited buttons or add them with a different name.

In addition to the extensive set of commercially published UI controls that the Visual Designer provides, you can also create and reuse custom controls that extend or combine the existing controls. You can create two types of custom controls:

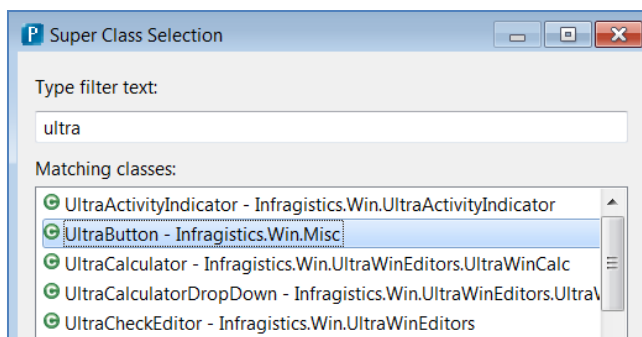
User controls: Composite controls made up of multiple individual controls grouped in a container.

Inherited controls: Controls that derive properties and events from a super class (that is, a parent control).

1. Select the **VDReports** project in the **Project Explorer** view. Right-click **VDReports** and select **New**→**ABL Inherited Control**. The **New ABL Inherited Control** wizard opens.
2. Specify **MyButton** in **Inherited Control name**. The following error message is displayed because Inherited Control inherits all properties from the super class control.

 You must specify the parent control (super class) in the Inherits field.

3. To create Inherited Control for **UltraButton** control, click **Browse** next to **Inherits**. Enter **ultra** in **Type filter text**. All the classes starting with ultra are displayed.



Note: If you cannot find the class you are looking for, add the same control that you are looking for; in this case it is UltraButton to a dummy form and then try this again.

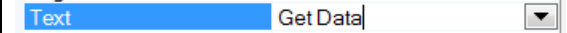
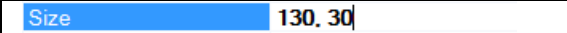
4. Select **UltraButton** from the list and click **OK**. **Inherits** now displays the **Infragistics.Win.Misc.UltraButton** class.

Inherits:

5. Click **Finish**. The **MyButton(Design).cls** file opens in the editor and it also appears under the **VDReports** project.

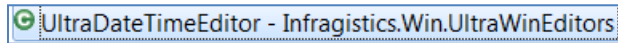


6. From the **Properties** view, modify the following **MyButton** properties and set values as shown:

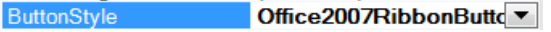
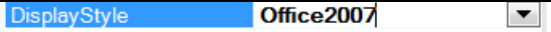
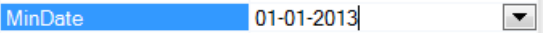
Property	Value	Displays as
Text	Get Data	
Size	130, 30	

7. Save the changes.
8. Similarly, create another ABL Inherited Control with the following details:

Inherited Control name: MyDateTimeEditor
Inherits: Infragistics.Win.UltraWinEditors.UltraDateTimeEditor



9. From the **Properties** view, modify the following **MyDateTimeEditor** properties and set the values as shown:

Property	Value	Displays as
ButtonStyle	Office2007RibbonButton	
DisplayStyle	Office2007	
MinDate	01-01-2013	

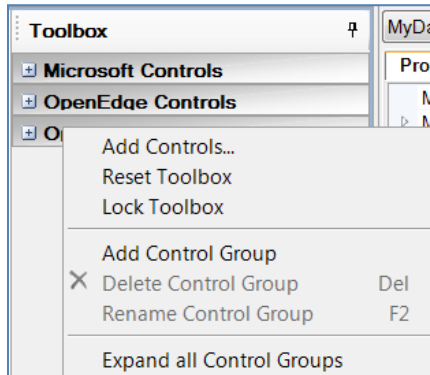
10. Save the changes. We have successfully created two Inherited controls which can be added to the Toolbox and used as regular controls.



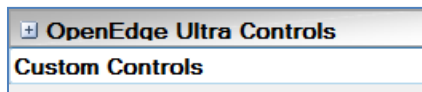
5.7 Adding controls to Toolbox

This section shows you how to add control to Toolbox.

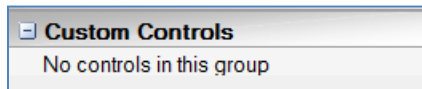
1. Right-click **Toolbox** and select **Add Control Group**.



2. Enter **Custom Controls** in the text box and press **Enter**.

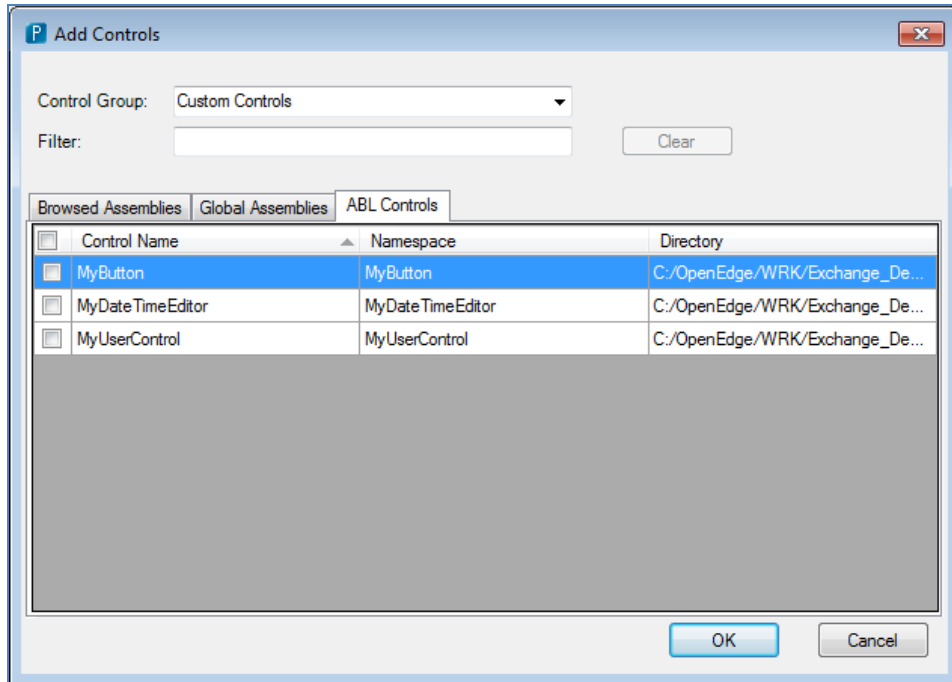


Control group **Custom Controls** is added to **Toolbox**.

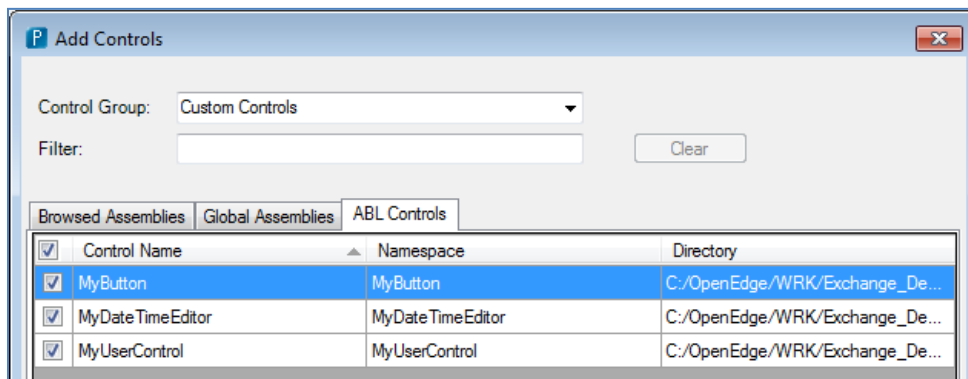


3. Right-click **Custom Controls** and select **Add Controls...**. The **Add Controls** dialog box appears.
4. Click the **ABL Controls** tab. We have already provided you with the **Inherited controls** and the **User control**. They are listed as follows:

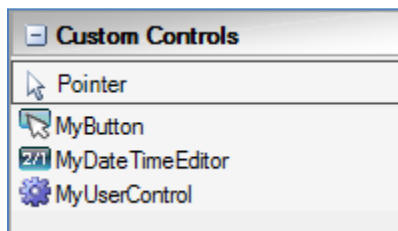




5. Select the check boxes corresponding to all the controls. Make sure that **Control Group** selected is **Custom Controls**.



6. Click **OK**. The selected controls are listed in Toolbox under the **Custom Controls** category.



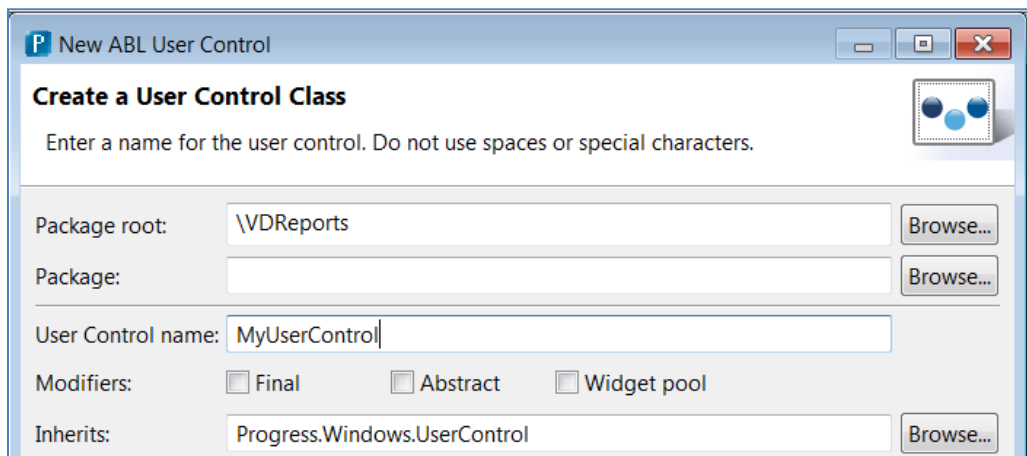
7. Click **File**→**Close All** to close all the files. We have now successfully added custom controls to the Toolbox. They are now ready for use.



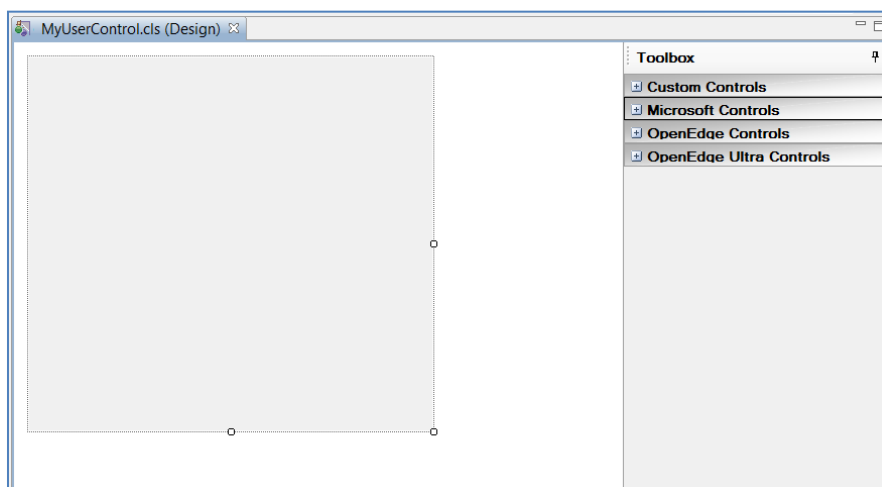
5.8 Creating User control (Take-home)

This section shows you how to create a User control so that you can combine the functionality of several controls into a single reusable unit. You will design a User control with the Inherited controls that you have created in the above section and add the User control to **Toolbox**.

1. Right-click the **VDReports** project in the **Project Explorer** view and select **New**→**ABL User Control**. The **New ABL User Control** wizard opens.
2. Specify **MyUserControl** in **User Control name**. By default, user control inherits **Progress.Windows.UserControl**.

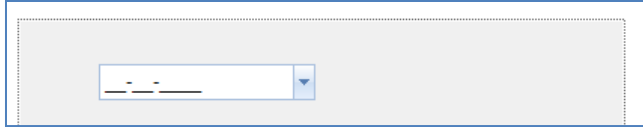


3. Click **Finish**. The **MyUserControl(Design).cls** file opens in the Design View.



4. Expand the **Custom Controls** category in **Toolbox**. Drag and drop **MyDateTimeEditor** control to the **User control**.





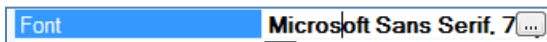
5. Similarly, add the **MyDateTimeEditor** control.
6. Expand the **Microsoft Control** category in **Toolbox** and select the **Label** control.



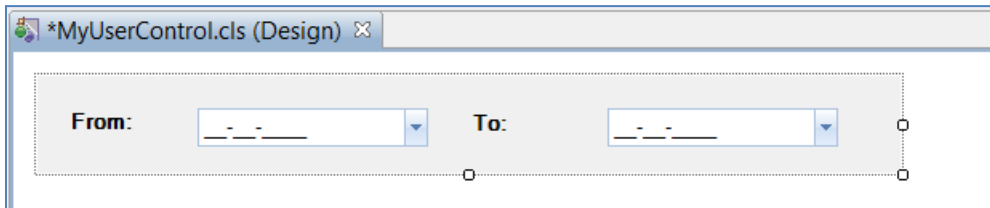
7. Drag and drop the **Label** control twice.
8. Arrange the controls as shown in the image Below:



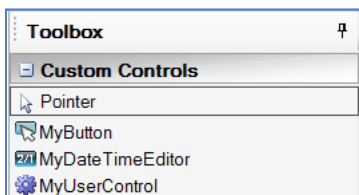
9. For both the **Label** controls, modify the **Font** property. Click  in the **Font** property value. A list of other **Font** properties appears. Select **ellipsis** button  and select **Font style** as **Bold**.



10. Specify the **Text** property of the two labels as shown in the image and resize the user control:



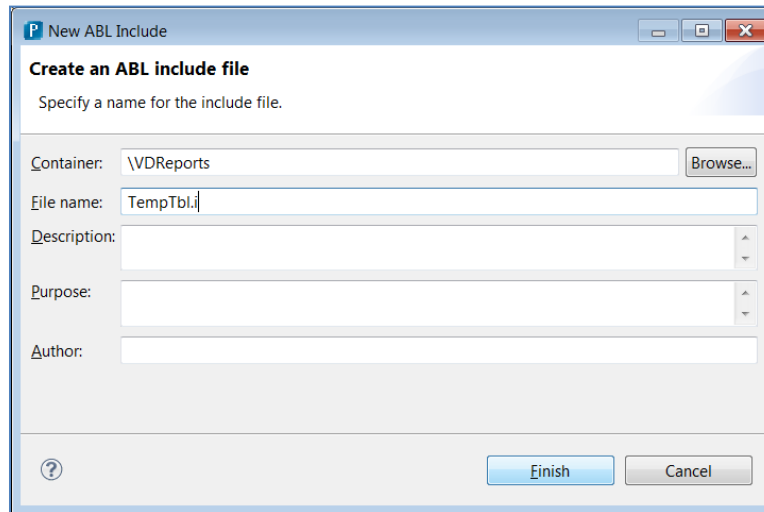
11. Save all the files.
12. Add **MyUserControl** User control to **ToolBox** under the **Custom Controls** group.



5.9 Designing the SalesReport form to display outlets data

This section shows you how to design the SalesReport form with controls created in the previous sections. You will design the **Outlets** tab to display the outlets data in a list for the selected dates.

1. To define the schema, create an ABL Include file. Right-click the **VDReports** project and select **New**→**ABL Include**. The **New ABL Include** dialog box appears.
2. Enter **TempTbl.i** in **File name** and click **Finish**.



The Include file opens in the editor and the TempTbl.i file is created under the VDReports project.

3. Copy and paste the following temp-table definitions in the include file.

```
DEFINE TEMP-TABLE ttOrderInfo
    FIELD ttDateOfOrder LIKE ORDER.OrderDate
    FIELD ttNoOfOrders AS INTEGER.

DEFINE TEMP-TABLE ttTotalTickets
    FIELD ttttDateOfOrder LIKE ORDER.OrderDate
    FIELD ttTotalOrders AS INTEGER
    INDEX idxOrder IS UNIQUE ttttDateOfOrder.

DEFINE TEMP-TABLE ttItemOrders
    FIELD fItemNum LIKE ITEM.ITEMNUM
    FIELD fItemName LIKE ITEM.ITEMNAME
    FIELD fNoOfOrders AS INTEGER
    INDEX idxItem IS UNIQUE fItemName.

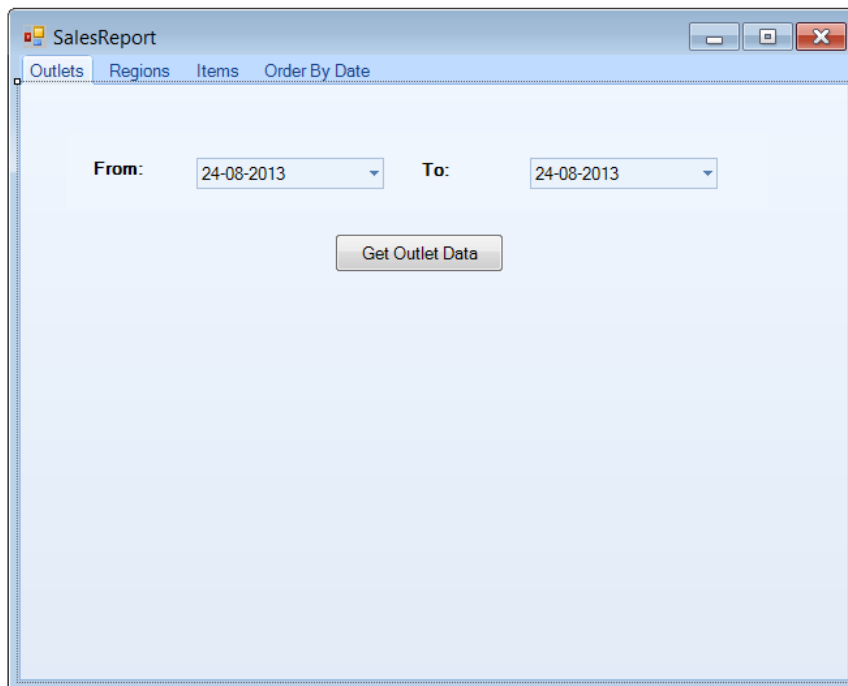
DEFINE TEMP-TABLE ttAreaOrders
    FIELD ttArea LIKE Outlet.Region
    FIELD ttNoOfOrders AS INTEGER.

DEFINE TEMP-TABLE ttOutletOrders
```



```
FIELD Outlet LIKE Outlet.OutletName
FIELD NoOfOrders AS INTEGER.
```

4. Save the file.
5. Double-click the SalesReport.cls file in **Project Explorer** view. The file opens in the **Design View** in the editor.
6. Add **MyUserController** User control and the **MyButton** Inherited control to the **Outlets** tab. The button has default properties as defined.
7. Specify **Get Outlet Data** in the **Text** property of **MyButton1**. Place controls as shown in screen below:




8. Expand the **OpenEdge Controls** category in **ToolBox** and select the **ProBindingSource** control.

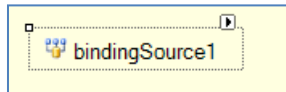



9. Drag and drop the **ProBindingSource** control onto the form. The **ProBindingSource Designer** dialog appears.



 **Note:** ProBindingSource is a non-visual control whose properties define the schema for the data to be displayed by the accompanying visual control. The ProBindingSource control includes a designer tool that helps you define this schema. The tool also gives you the option of importing the schema from an XML schema (XSD) file, or from an ABL source file (like p, cls, w, i, and html).

10. Click **OK**. The controls appear in a separate area at the bottom of the Design View.

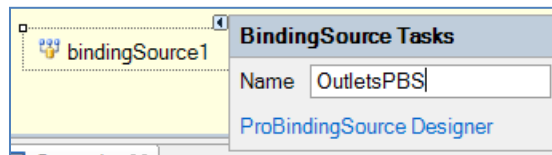


 **Note:** UI controls fall in one of two basic categories:

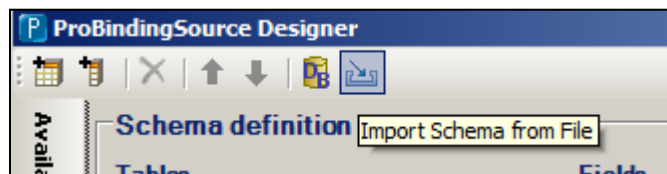
Visual controls appear on the application's user interface and allow user interaction, display data, or both. Examples include buttons, combo boxes, and data grids.

Non-visual controls do not appear on the user interface at run time; instead, they support visual controls or provide other services such as logging. Non-visual controls typically hold data, formatting, or other information needed by one or more interactive controls. An example is the ProBindingSource, which serves as an intermediary between an actual ABL data source and the control, such as a grid, that displays the data.

11. Open the **SmartTag** of **bindingSource1** control and enter **OutletsPBS** in **Name** and click the **ProBindingSource Designer** link.



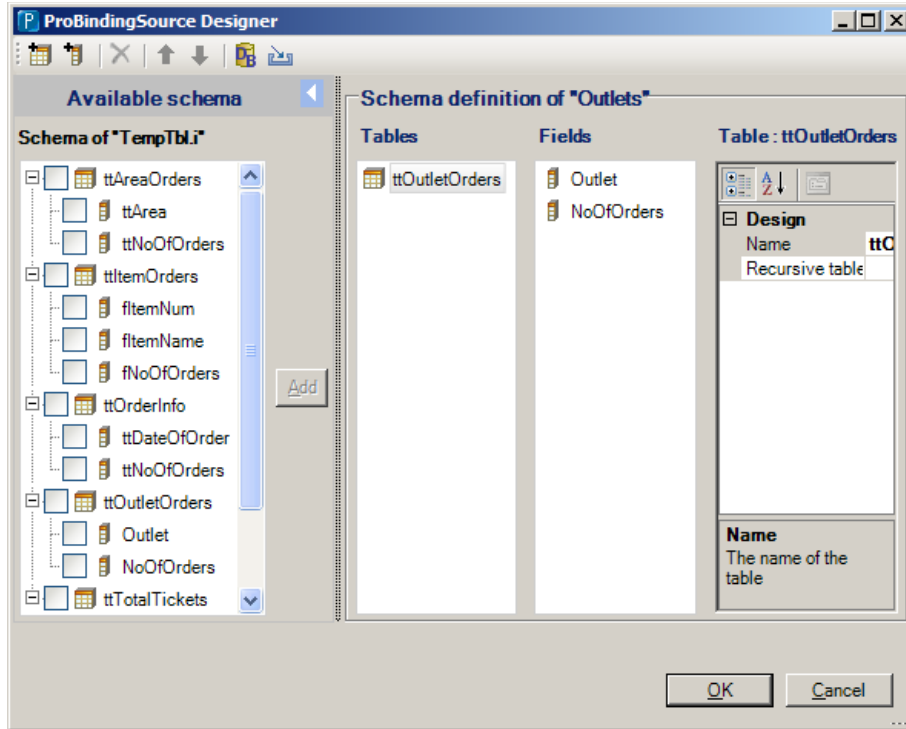
12. In the **ProBindingSource Designer** dialog box, select **Import From File** on toolbar as shown below:



13. Select **TempTbl.i** from the location
C:\OpenEdge\WRK\Exchange_PDSOE\VDReports.

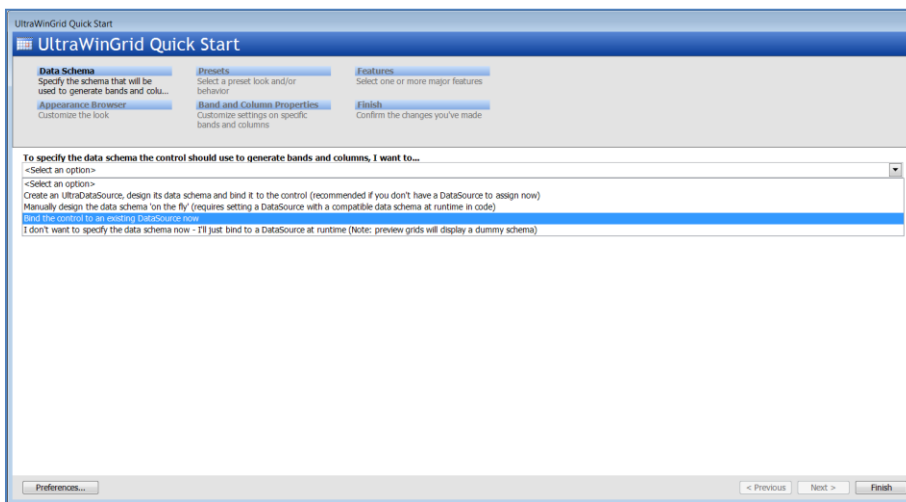
14. Select **ttOutletOrders** from the schema in left section of designer and click **Add**.





15. Click **OK**.

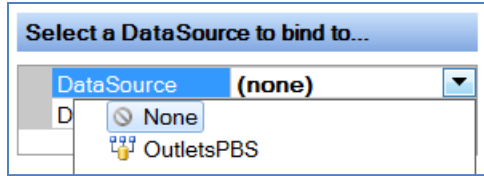
16. Expand the **OpenEdge Ultra Controls** category in **ToolBox**. Drag and drop the **UltraGrid** control onto the form. The **UltraWinGrid Quick Start** wizard appears.



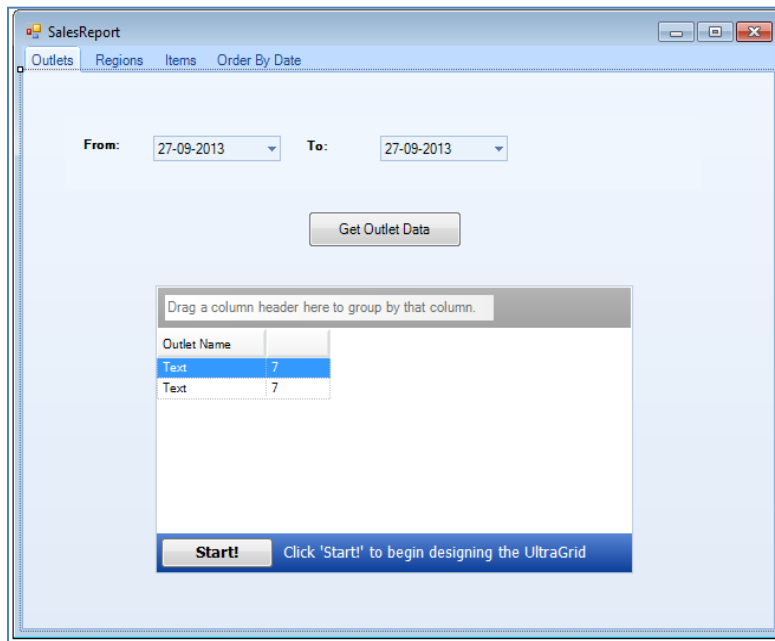
17. Select the **Bind the control to an existing DataSource now** option from the combo-box.

18. From the **DataSource** combo-box, select **OutletsPBS**.





19. Click **Finish**. The **UltraGrid** control is added to the **Outlets** tab. Modify the **UltraGrid** control size and place the control as shown in screen below:



20. Save all the files.

We have now completed designing our Sales data screen. We have added a User control which contains two **DateTime** fields to pick the dates, added a **ProBindingSource** control to fetch the data from database, and configured **ProBindingSource** with **Outlet** and **No of Orders** fields. We have also added **UltraGrid** control which can show flat or hierarchical data and associated this control with the **ProBindingSource** to show the data.

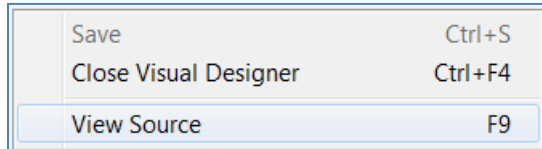
We will now add an event to the **Button** control and then add logic to the event to retrieve the data.



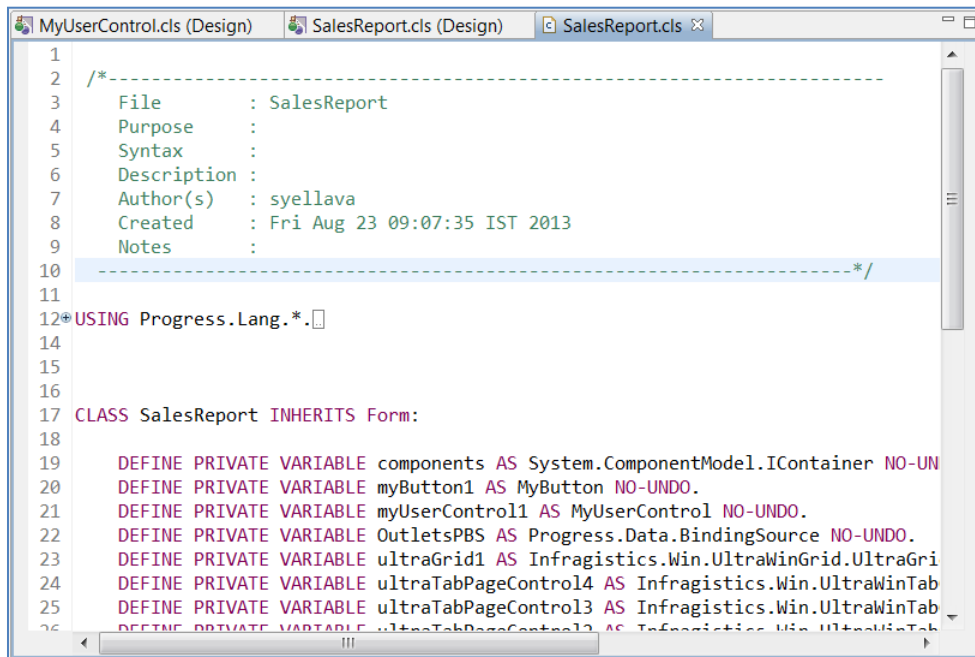
5.10 Working with the source editor - Adding methods and events

This section shows you how to work with the source editor, add methods and events to the form, and use Editor Actions.


1. Open the source view of the form. Right-click the form and select **View Source**.




The source view of the **SalesReport.cls** file opens in the editor. The source and design view will always be in sync.

A screenshot of the ABL Editor showing the source code for SalesReport.cls. The code is color-coded and includes a header block with metadata, a USING statement, and a CLASS definition for SalesReport inheriting from Form. The class contains several DEFINE PRIVATE VARIABLE statements for components like components, myButton1, myUserControl1, OutletsPBS, ultraGrid1, ultraTabPageControl14, and ultraTabPageControl13.

```
1
2 /*-----
3 File      : SalesReport
4 Purpose   :
5 Syntax    :
6 Description:
7 Author(s) : syellava
8 Created   : Fri Aug 23 09:07:35 IST 2013
9 Notes     :
10 -----*/
11
12*USING Progress.Lang.*.
13
14
15
16
17 CLASS SalesReport INHERITS Form:
18
19     DEFINE PRIVATE VARIABLE components AS System.ComponentModel.IContainer NO-UN
20     DEFINE PRIVATE VARIABLE myButton1 AS MyButton NO-UNDO.
21     DEFINE PRIVATE VARIABLE myUserControl1 AS MyUserControl NO-UNDO.
22     DEFINE PRIVATE VARIABLE OutletsPBS AS Progress.Data.BindingSource NO-UNDO.
23     DEFINE PRIVATE VARIABLE ultraGrid1 AS Infragistics.Win.UltraWinGrid.UltraGri
24     DEFINE PRIVATE VARIABLE ultraTabPageControl14 AS Infragistics.Win.UltraWinTab
25     DEFINE PRIVATE VARIABLE ultraTabPageControl13 AS Infragistics.Win.UltraWinTab
26     DEFINE PRIVATE VARIABLE ultraTabPageControl12 AS Infragistics.Win.UltraWinTab
```

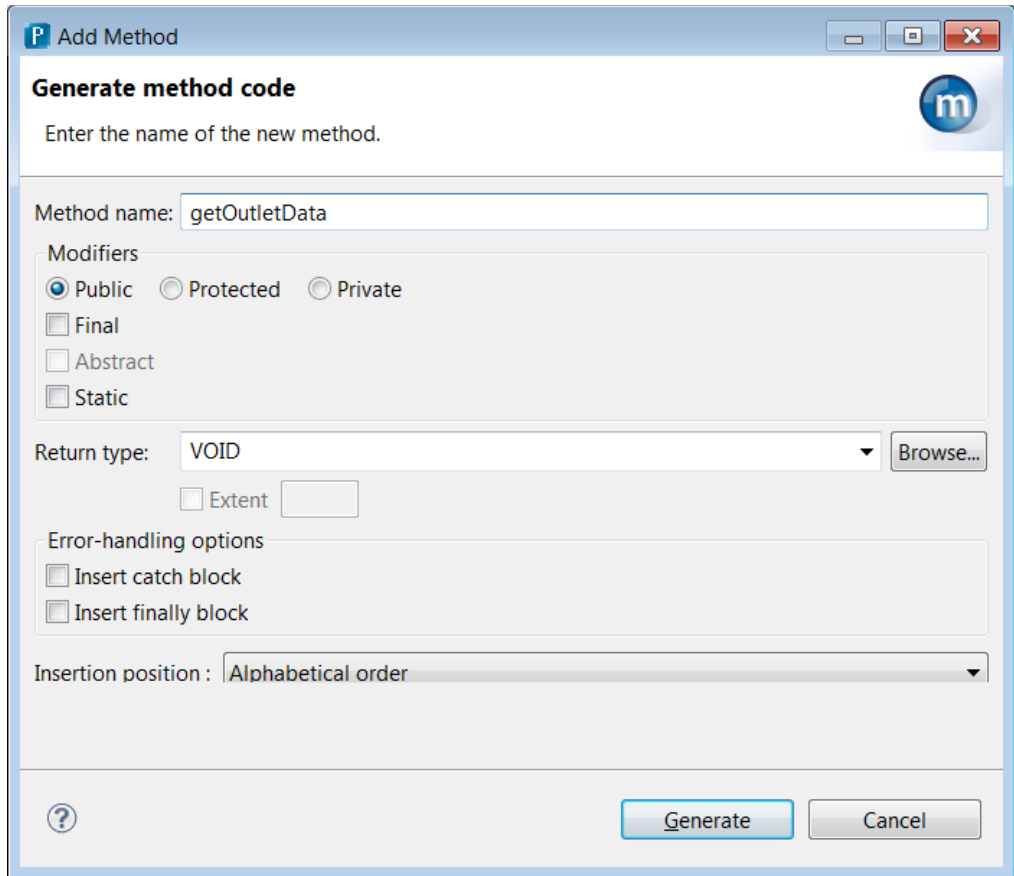
 **Note:** The ABL Editor displays different syntax elements in different colors to make them easily recognizable at a glance. You can use the default color scheme or go to the ABL Editor Preferences page to create your own.

2. Right-click the editor and select **Source** → **Add Method**. The **Add Method** dialog box appears.

 **Note:** The Source context menu has options to add code using wizards. For example, Add method, properties, procedure, and function options open a dialog. When you provide name or related information, it generates the code in editor automatically.

3. Enter **getOutletData** in **Method name**.





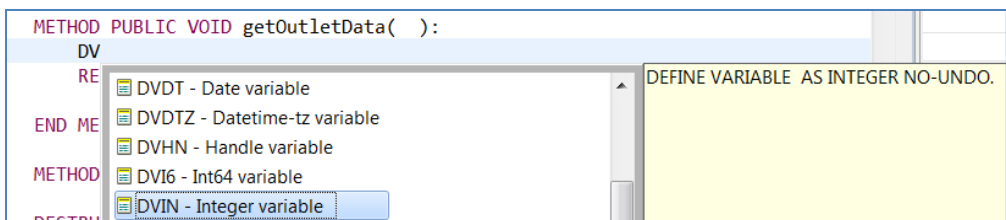
4. Click **Generate**. The **getOutletData** method is added to source.

```

45  /*-----
46  Purpose:
47  Notes:
48  -----
49
50  METHOD PUBLIC VOID getOutletData( ):
51
52      RETURN.
53
54  END METHOD.
55

```

5. After the method statement, type **DV** and press **Ctrl+SPACEBAR**. The content assist window opens and displays the macros available for Define variables. When you select an option, its expanded code is displayed in the next box.



- Double-click **DVIN** from list. The define integer variable statement is added and cursor is placed at position where variable name needs to be added. Enter **cntr** in variable name.

```
DEFINE VARIABLE cntr AS INTEGER NO-UNDO.
```



Tip: Syntax-completion assistance proposes syntax to complete the code that you are typing in the ABL Editor. When you press **CTRL+SPACEBAR**, proposals appear in the left pane of a pop-up window, with reference information about the selected item in the right pane. To insert an element at the current cursor position, select it, and double-click or press **ENTER**.

By default, syntax-completion assistance filters proposals based on the context (for example, showing only relevant keywords). Press **CTRL+SPACEBAR** while the pop-up window is open to toggle between context-filtered proposals and all proposals. The assistant is also data aware, when you select a keyword; it lists all the database tables.

- After the define statement, press **Enter**. Press **CTRL+SPACEBAR**. The content assist dialog-box opens.
- Type **FOR**. The filtered list appears and displays the options starting with **for**. Similarly add the following code with the help of content assistant. Press **SPACEBAR** after each selected option. Press **CTRL+SPACEBAR** after providing each keyword.

```
FOR EACH Outlet BREAK BY Outlet.OutletName
```

The content assistant displays the following after each keyword:

Content Assist List	Observation
<pre>DEFINE VARIABLE cntr AS INTEGER NO-UNDO. for </pre>	Options applicable to FOR are only displayed.
<pre>DEFINE VARIABLE cntr AS INTEGER NO-UNDO. for EACH</pre>	All relevant tables from the connected database are listed. Select workshop.Outlet .
<pre>DEFINE VARIABLE cntr AS INTEGER NO-UNDO. for EACH Outlet BREAK BY</pre>	As the Outlet table is already selected, after BREAK BY , field proposals of the Outlet table are listed. Select workshop,Outlet.City .

- Add the following code to the method statement to get the outlet data:




```

METHOD PUBLIC VOID getOutletData( ):
    DEFINE VARIABLE cntr AS INTEGER.
    FOR EACH Outlet BREAK BY Outlet.OutletName:
        cntr = 0.
        FOR EACH Order WHERE DATE(Order.OrderDate) > frmDate AND
date(Order.OrderDate) < tDate:
            IF Order.Outlet = Outlet.OutletName THEN
                DO:
                    cntr = cntr + 1.
                END.
            END.
        END.

    /*          Create temp table with data*/
    CREATE ttOutletOrders.
    ASSIGN
        ttOutletOrders.Outlet      = Outlet.OutletName
        ttOutletOrders.NoOfOrders = cntr.

    END.
END METHOD.

```

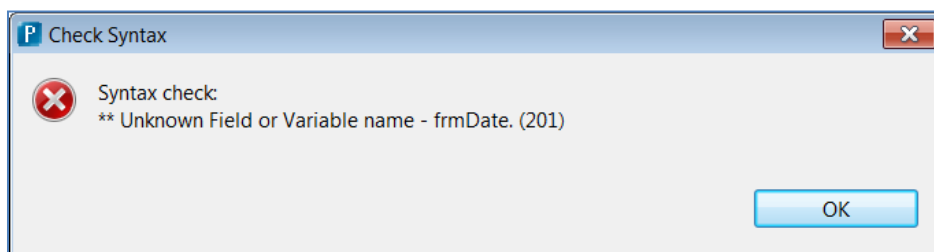
10. Save the file. Error markers  are displayed for **FOR EACH Order** statement, for the **SalesReport.cls** file in the **Project Explorer** view and the editor. Error marker represents compilation errors in file.
11. Right-click the editor and select **Check Syntax** or press **Ctrl+Shift+C**.

Check Syntax
Ctrl+Shift+C



Tip: The options also provide the short cut keys.

The **Check Syntax** dialog box appears with the following error message:



12. Select **OK** to close the **Check Syntax** message.

The **Outlets** tab design takes **From** and **To** dates as input and if you click **Get Outlet Data**, it displays the results of Outlets data between given dates. Add the following code for **From** and **To dates** to the method.

```

METHOD PUBLIC VOID getOutletData(frmDate AS DATE, tDate AS DATE
):

```

13. Save the file.



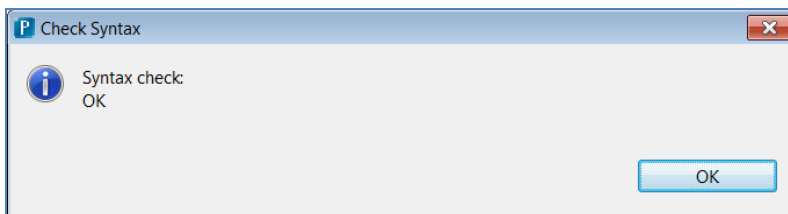
14. Open the source view of the SalesReport.cls file in the editor.
15. Above the Constructor statement, add the following code to include {TempTbl.i} file in source.

```

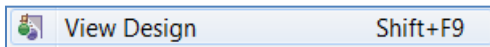
DEFINE PRIVATE VARIABLE ultraPanel1 AS Infragistics.Win.Misc.UltraPanel
    {TempTbl.i}
CONSTRUCTOR PUBLIC SalesReport ( ):

```

16. Save the SalesReport.cls file and observe that the error markers are cleared. Click **Check Syntax** again and click **OK** at the message.



17. Right-click the editor and select **View Design** to open the Design View.




18. Double-click **Get Outlet Data** on the Design Canvas, the source view opens. The **myButton1_Click** event method is automatically generated and the cursor is placed at the method.

```

321 CAST(THIS-OBJECT:ultraTabControl1, System.ComponentModel.ISupportInitialize):EndInit().
322 THIS-OBJECT:ultraTabControl1:ResumeLayout(FALSE).
323 THIS-OBJECT:ResumeLayout(FALSE).
324 CATCH e AS Progress.Lang.Error:
325     UNDO, THROW e.
326     END CATCH.
327 END METHOD.
328
329 /*-----
330 Purpose:
331 Notes:
332 -----*/
333 @VisualDesigner.
334 METHOD PRIVATE VOID myButton1_Click( INPUT sender AS System.Object, INPUT e AS System.EventArgs ):
335
336     RETURN.
337
338 END METHOD.
339
340 DESTRUCTOR PUBLIC SalesReport ( ):
341
342     END DESTRUCTOR.
343
344 END CLASS.

```

 **Tip:** The quickest way to define an event method is to subscribe to the event by double-clicking in the **Design Canvas** or the **Events** tab. When you do this, you are taken to the automatically generated event-handling method in the ABL code.



The double-click technique minimizes typing errors, but you can also use the **Properties** view to associate any method with an event. On the **Events** tab, click the value cell to the right of the Event name. The down-arrow button that appears lets you select from a drop-down list of all methods defined in the source code whose signature matches that of the selected event. Alternatively, you can type a method name in the cell. You can change the name of the event handler method by editing the value on the Events tab. The change is reflected in the source code.

- Copy and paste code shown in the **myButton1_Click** method to get From and To dates from the design view, query ttOutletOrders table with those dates to get data, and assign that query to ProBindingSource control handle to hold query results in the ProBindingSource handle.

```
METHOD PRIVATE VOID myButton1_Click( INPUT sender AS
System.Object, INPUT e AS System.EventArgs ):

DEFINE VARIABLE qh      AS HANDLE      NO-UNDO.
DEFINE VARIABLE frmDate AS DATE        NO-UNDO.
DEFINE VARIABLE tDate   AS DATE        NO-UNDO.
DEFINE VARIABLE qryStr  AS CHARACTER  NO-UNDO.

    frmDate = myUserControl1:getFromDate().
    tDate = myUserControl1:getToDate().
    getOutletData(frmDate,tDate).
    qryStr = "FOR EACH ttOutletOrders".
    CREATE QUERY qh.
    qh:SET-BUFFERS(BUFFER ttOutletOrders:HANDLE).
    qh:QUERY-PREPARE(qryStr).
    qh:QUERY-OPEN.
    OutletsPBS:Handle= qh.

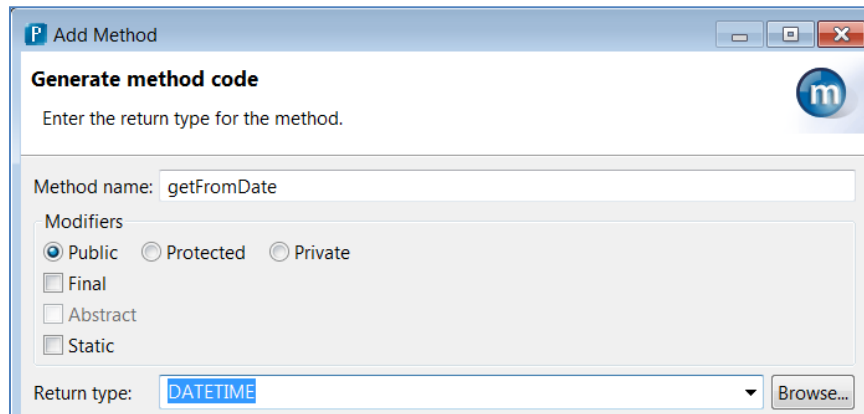
END METHOD.
```

- Press **Ctrl+I** to indent file code. Save the file.
- Click **Check Syntax** or alternately hover over the error marker. The following error message is displayed.

- Press **CTRL+SPACEBAR** after **getFromDate**, there are no proposals listed representing the methods are not added to User Control.

- Double-click **MyUserControl.cls** in the **Project Explorer** view to open the file in Design View.
- Right-click and select **View Source** or alternatively press **F9**.
- Press **Alt+Shift+M** and add two public methods **getFromDate** and **getToDate** with return type as **DATETIME**.





26. Add the following statements to get the value of the DateTime field from UI and return the same result variable:

```

METHOD PUBLIC DATETIME getFromDate( ):
    DEFINE VARIABLE result AS DATETIME NO-UNDO.
    result = myDateTimeEditor1.Value.
    RETURN result.

END METHOD.

METHOD PUBLIC DATETIME getToDate( ):
    DEFINE VARIABLE result AS DATETIME NO-UNDO.
    result = myDateTimeEditor2.Value.
    RETURN result.
END METHOD.

```

27. Save all the changes.
28. Open the **SalesReport.cls** file.
29. Right-click and select **Progress OpenEdge**→**Compile**. The file is compiled and you see no error markers now.



Tip: By default, Eclipse compiles source files when they are added to the workspace or saved. You can disable the Build automatically option from the Eclipse Workspace preferences (Window > Preferences > General > Workspace).

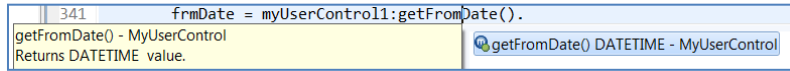
You may choose to disable automatic builds if the contents of your workspace are frequently updated significantly (for example, by copying files from an external source), triggering build processes that block creating or saving resources until they finish. We recommend that you should enable the automatic builds to ensure that up-to-date r-code is always available for tools and features that require it.

If you do disable automatic builds, it is recommended that you have Progress Developer Studio for OpenEdge compile ABL source files when you save them. To do so, select the **Compile on save if required** option in the **Editor Build** preferences.

At any time, you can explicitly compile your current file by selecting **Compile** from either the Source menu or the ABL Editor context (right-click) menu.

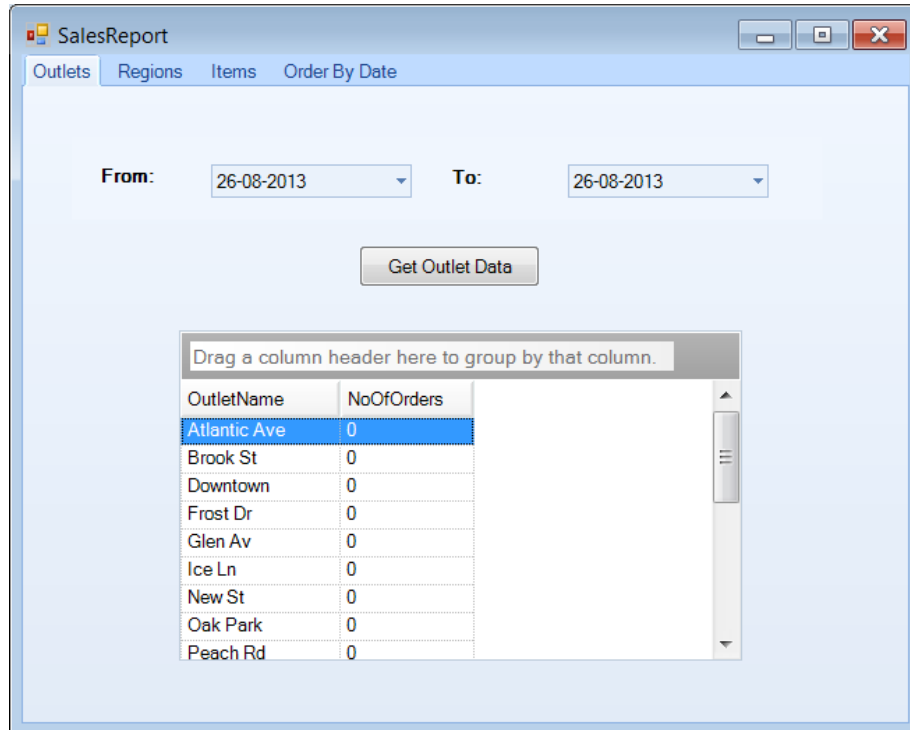


30. Press **CTRL+SPACEBAR** after **getFromDate** again and the proposals are listed now.



31. Run the **SalesReport.cls** file.

32. Select **From** and **To** dates and then click on **Get Outlet Data** button. The sale per outlet data appears.



33. Close the **Run** dialog box. We have now completed creating an application to view Sales data by date.



5.11 Working with OpenEdge Debugger (Optional)

This section shows you how to work with Debugger and use the Debugger to find logic errors in ABL applications by adding a breakpoint for a specific executable line in the ABL code.

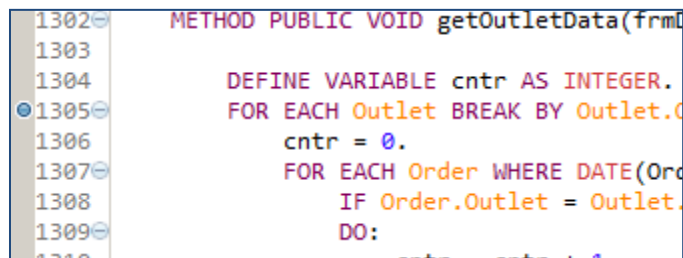
The key to debugging is the ability to run a program and suspend execution at strategic points so that you can monitor and evaluate the results. To allow you to control the program flow in this manner, Debugger includes the following features:

Breakpoints: You can insert breakpoints on executable statements anywhere in your source code and in the include files that might appear in many different procedures. The Debugger suspends execution at each breakpoint.

Code-stepping: You can discretely execute the next statement, the next statement plus any sub-procedure or trigger called by that statement, or the remainder of the current procedure.

Suspend-Resume-Terminate commands: You can explicitly interrupt or resume execution. You can also suspend and resume an attached external AVM at any time.

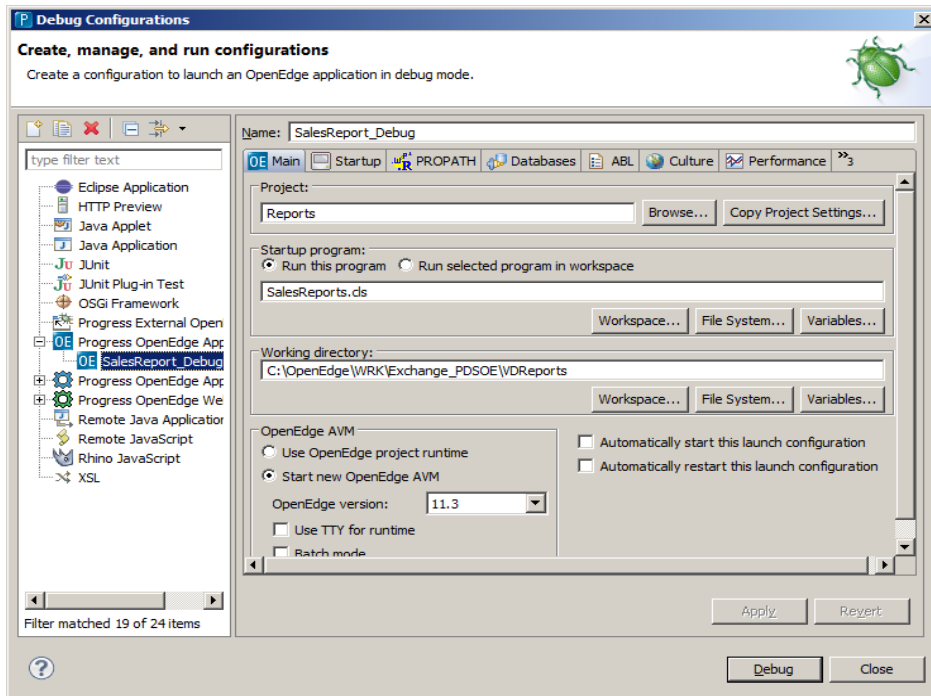
1. Open the source view of the **SalesReport.cls** form that you created in the previous section in editor view.
2. From the **Outline** view, select the **getOutletData** method from the Methods section.
3. On the left margin to the first executable line in **getOutletData**, double-click to set the break point.



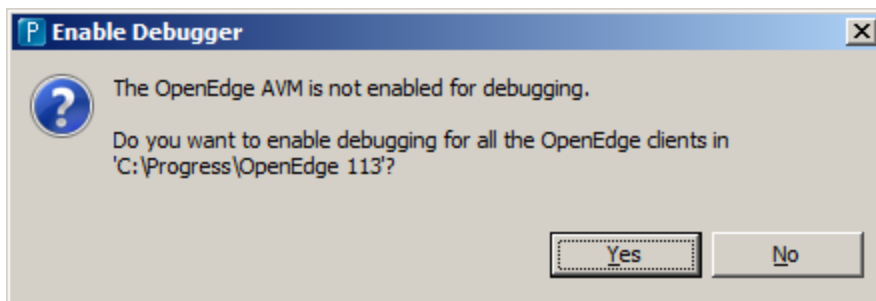
```
1302 METHOD PUBLIC VOID getOutletData(frml
1303
1304     DEFINE VARIABLE cntn AS INTEGER.
1305     FOR EACH Outlet BREAK BY Outlet.C
1306         cntn = 0.
1307     FOR EACH Order WHERE DATE(Ord
1308         IF Order.Outlet = Outlet.
1309         DO:
```

4. Click **Run ->Debug Configurations**. The **Debug Configurations** dialog box opens.
5. Double-click **Progress OpenEdge Application**. Name the configuration as **SalesReport_Debug**.

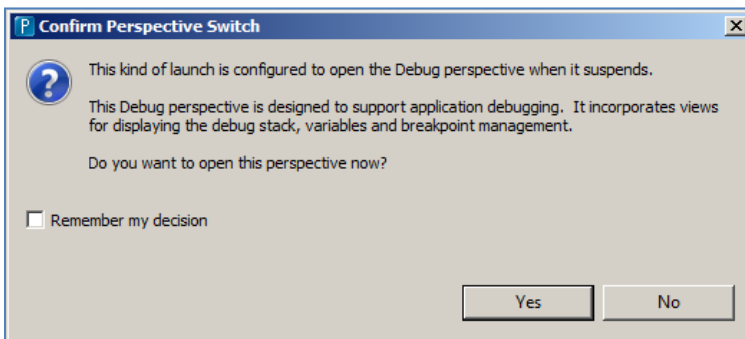




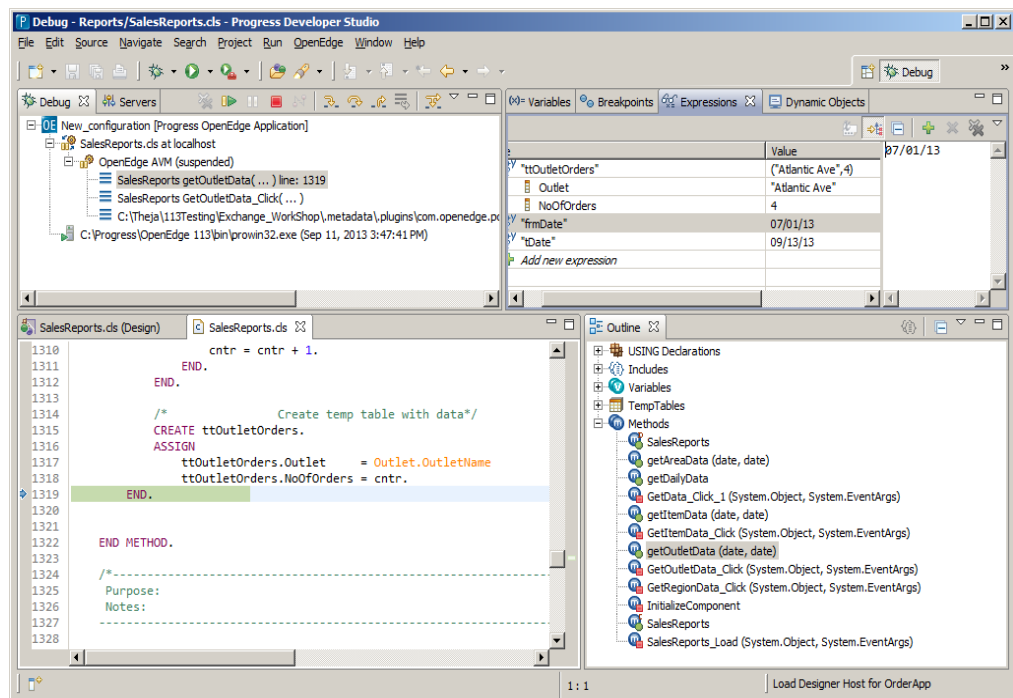
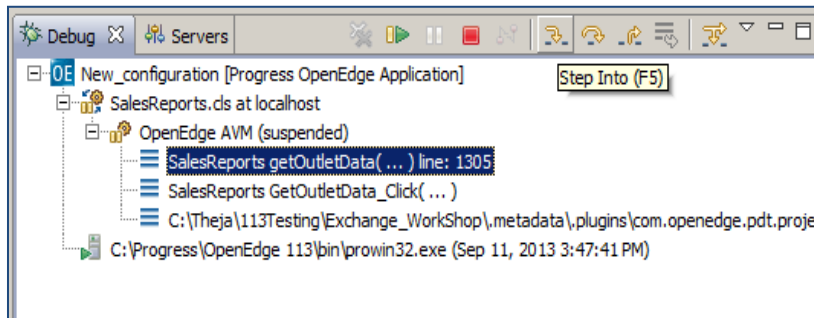
6. Click **Debug** to run the **SalesReport.cls** form in Debug mode.
7. Select **Yes** to enable debugging in the **Enable Debugger** dialog box.



8. Provide **From** and **To Dates** in the **Outlet** tab from UI and click **GetOutletData**.
9. Select **Yes** to switch the perspective.



10. In the **Debug** perspective, from the **Variables** section, select **frmData**, **tDate**, and **ttOutletOrders**, right-click and select **Watch**. The **Expressions** tab opens.
11. Select **Step Into** toolbar in the **Debug** view. **Step Into** causes the Debugger to execute the current line and continue until it reaches the next executable statement, which may be in the current procedure, a subprocedure, or a trigger. That statement becomes the current line, and is not executed until you continue. You can do **Step Into** to inspect each line and the value of variables at each line.



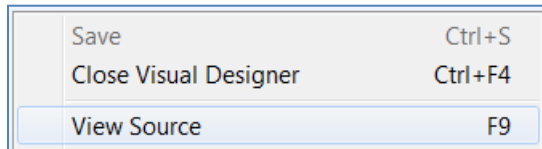
12. Select **Resume** for the variables to continue running.
13. Select **Terminate** to stop debugging the form.
14. Close the **SalesReport** form.



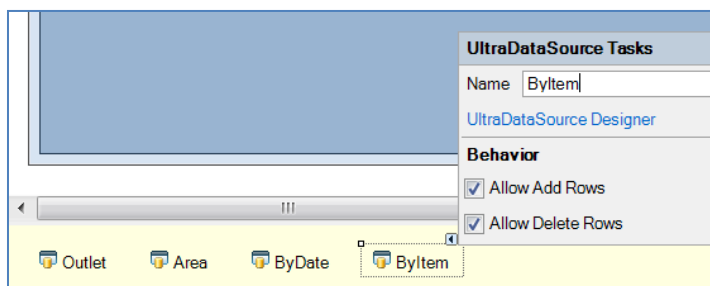
5.12 Working with UltraChart (Take-home)

This section shows you how to design graphs to provide data in a graphical format using the UltraChart control.

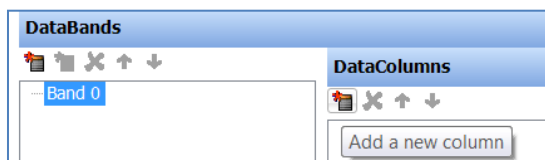
1. Open the **Visual Designer** perspective.
2. Copy and paste the **Graphs.cls** file from **C:\OpenEdge\WRK\PDSOEWorkshopFiles\WorkshopFiles** folder to the **VDReports** project in the **Project Explorer** view.



3. Open the **Graphs.cls** in the Design View.
4. Add four **UltraDataSource** controls to the Graphs form and change the **Name** property value to **Outlet**, **Area**, **ByDate**, and **ByItem** in the **Property** tab or using the **SmartTag**.

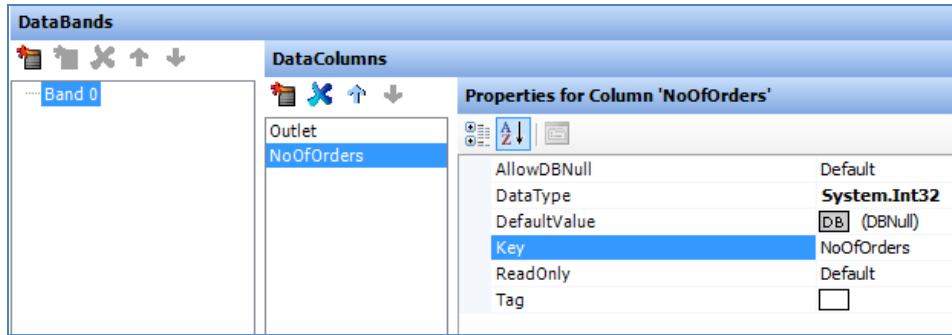


5. Click **File**→**Save All** to save all files.
6. Open **SmartTag** for the **Outlet** UltraDataSource and select the **UltraDataSource Designer** link. The **UltraDataSource Designer** dialog box appears.
7. Click **Add a new column** toolbar option in **DataColumns** section.



8. In the Properties section, enter **Outlet** in **Key** and **System.String** in **Data Type**. Similarly add another column for **NoOfOrders** and **System.Int32** in **Data Type**. Your screen appears as follows:

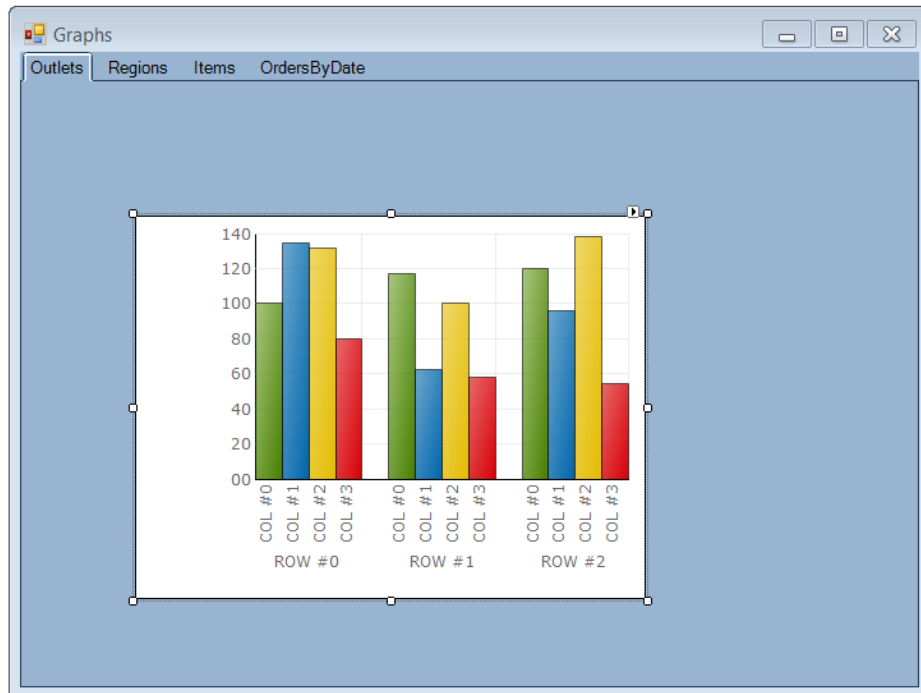




9. Select **OK** to close the **UltraDataSource Designer** dialog box.
10. Similarly, add **DataColumns** for other **UltraDataSources**.

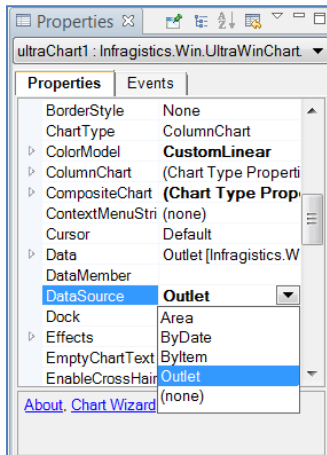
UltraDataSources	DataColumns Key	DataType
Area	Area	System.String
	NoOfOrders	System.Int32
ByDate	OrderDate	System.DateTime
	NoOfOrders	System.Int32
ByItem	Item	System.String
	Orders	System.Int32

11. Drag and drop the **UltraChart** control on to the **Outlets** tab. The **Select a Chart Type to Begin** dialog box appears.
12. Click **Finish**. The **UltraChart** control is added to Graphs form.



13. Select the chart on the form. In the **Properties** tab, select **Outlet** in **DataSource**.



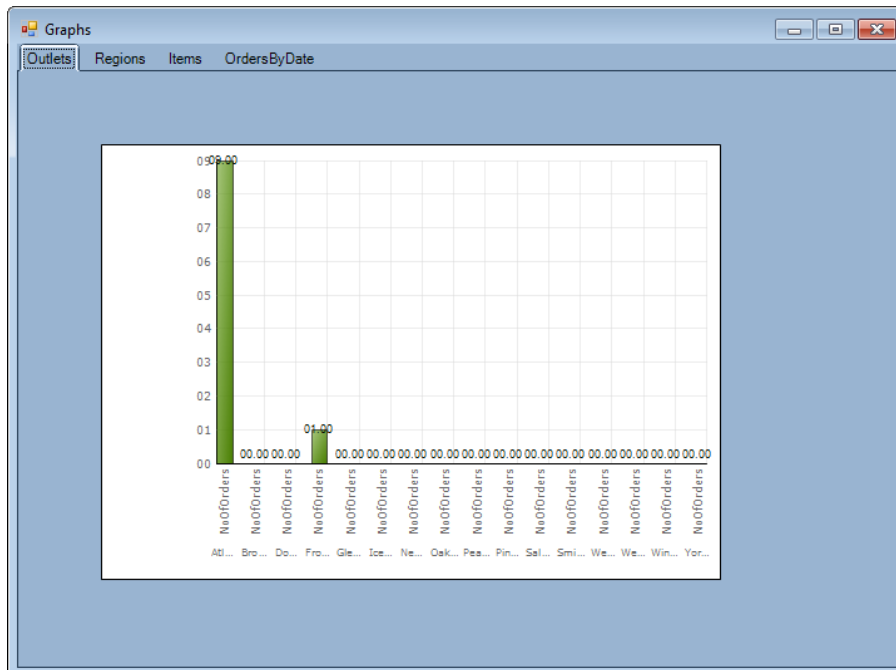


14. Similarly, add **UltraChart** in other tabs and select the **DataSource** as follows:

Tab	DataSource
Regions	Area
Items	ByItem
OrderByDate	ByDate

15. Save all the files.

16. Click **Run As**  and select **Run As Progress OpenEdge Application**.



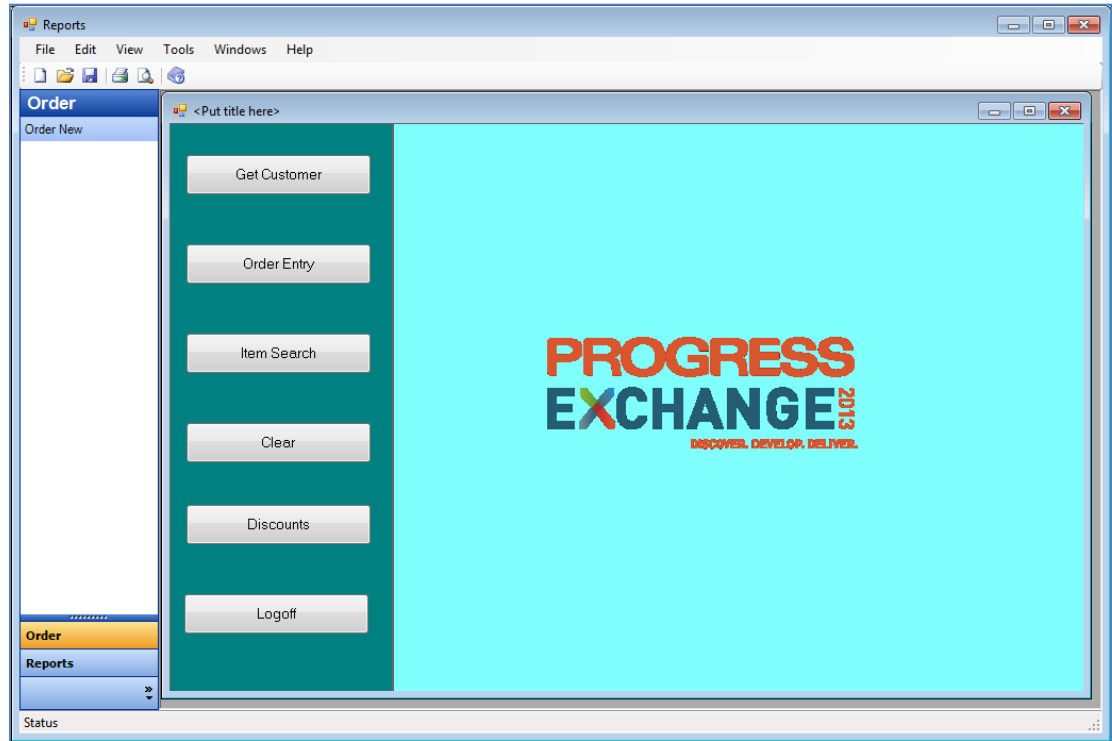
17. Close the **Run** dialog box. Click **File**→**Close All** to close all files.



6 LAB 06: Embedding an ABL window into the MDI form (Take-home)

6.1 Overview

The sections of this lab show you how to create an MDI form and embed AppBuilder and Visual designer forms into this MDI form.



6.2 Prerequisites

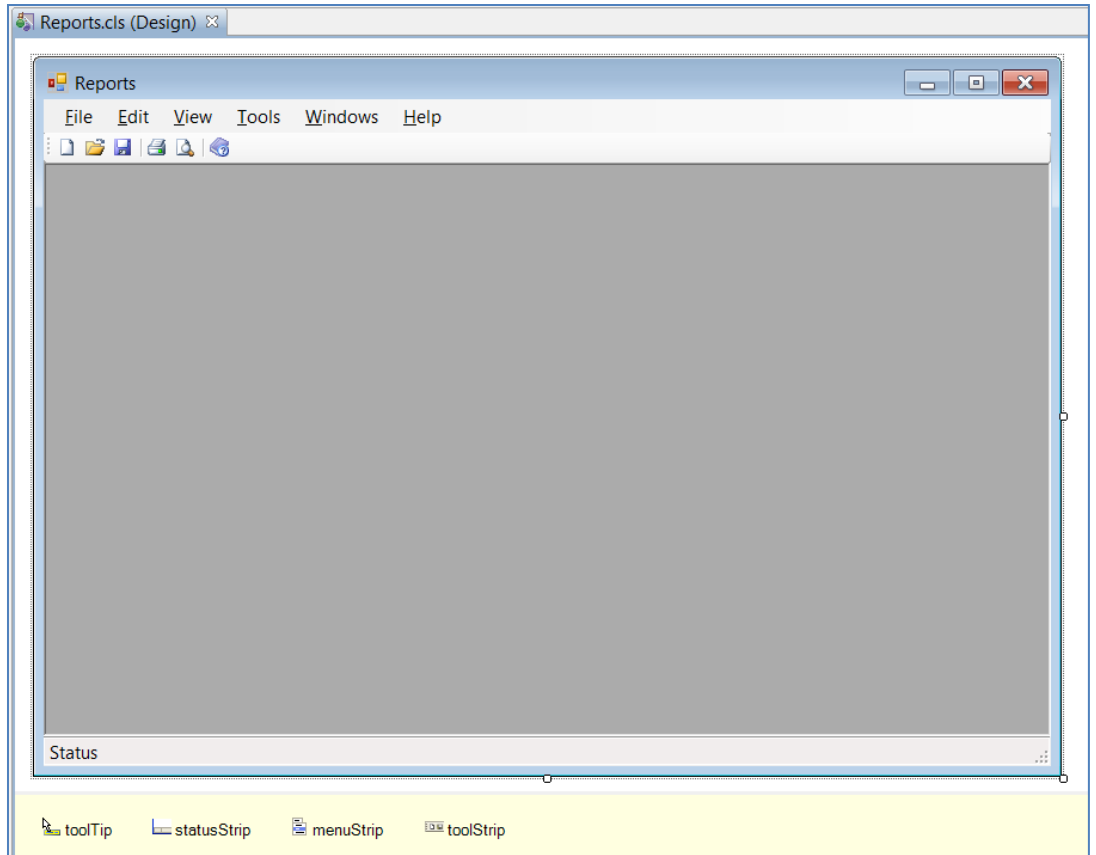
Complete all the above labs prior to working on this lab.


6.3 Creating an MDI form


This section shows you how to create an ABL MDI form.


1. Right-click **VDReports** and select **New→ABL MDI Form**. The **New ABL MDI Form** dialog box appears.
2. Enter **Reports** in **MDI Form name**.
3. Click **Finish**. The MDI form that is created opens in editor.





 Note: The **New ABL MDI form** wizard creates a form enabled for multiple-document interface functionality; you can use it as a parent form for other forms. The form includes a toolbar containing common menus (File, Edit, View, Tools, Windows, and Help) and command buttons (New, Open, Save, Print, Print Preview, and Help) with pre-coded event subscriptions and logic. It also includes a status bar at the bottom.

 **Tip:** To view complete MDI form double-click a tab in editor, it will expand tab size and occupies complete workbench area. Similarly, clicking any view expands the view area.

4. Click  icon in the **Console** view to close view.
5. Drag and drop the **UltraExplorerBar** control on to the **Reports** MDI form. Modify the following properties in the **Properties** view and save the changes.

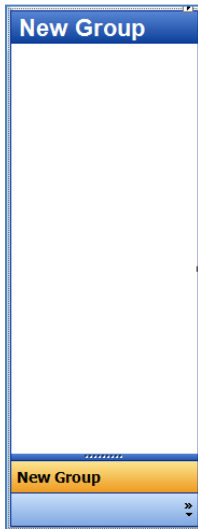
Property	Value	Displays as:
Dock	Left	Dock Left
Style	OutlookNavigationP ane	Style OutlookNavigationPane

In the **Properties** view, the following links are provided for the UltraExplorerBar control:

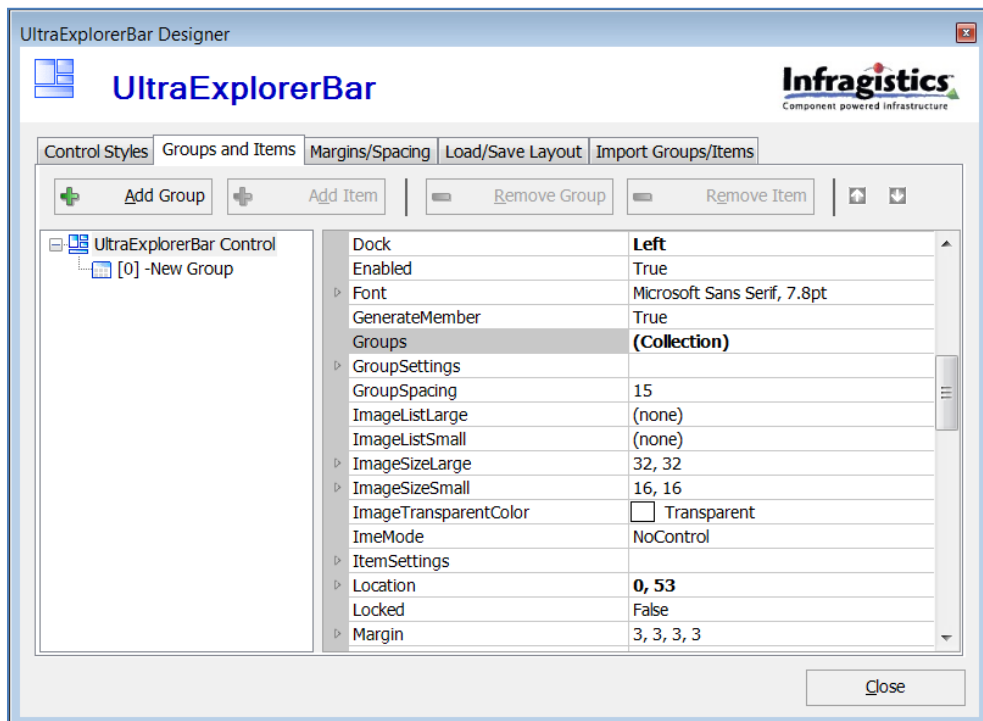
[About Custom Property Pages...](#) [UltraExplorerBar Designer...](#) [Add Group](#) [Add Item](#) [Remove Group](#) [Remove Item](#) [Load Layout...](#) [Save Layout...](#)



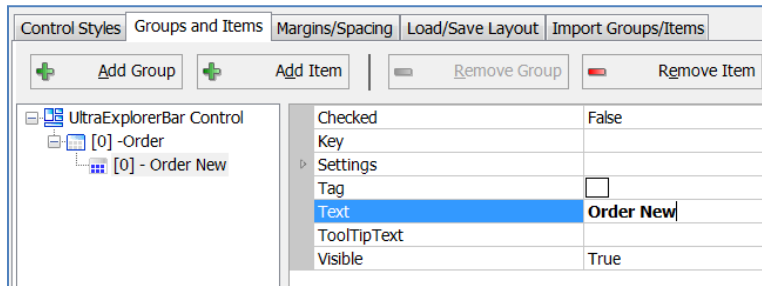
- Click the **Add Group** link. A **New Group** section is added to **UltraExplorerBar**.



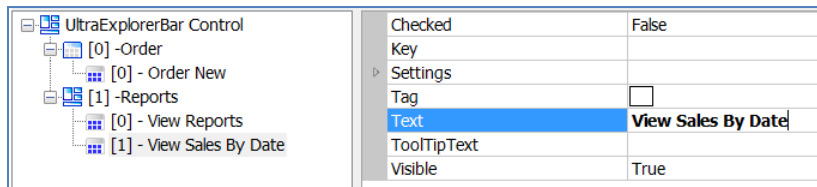
- Click the **UltraExplorerBar Designer** link. The **UltraExplorerBar Designer** dialog box appears.
- Click the **Groups and Items** tab.



9. Select the **[0] – New Group** node. Enter **Order** in the **Text** property.
10. Click **Add Item**. It adds a child node to Order.
11. Enter **Order New** in the **Text** property.

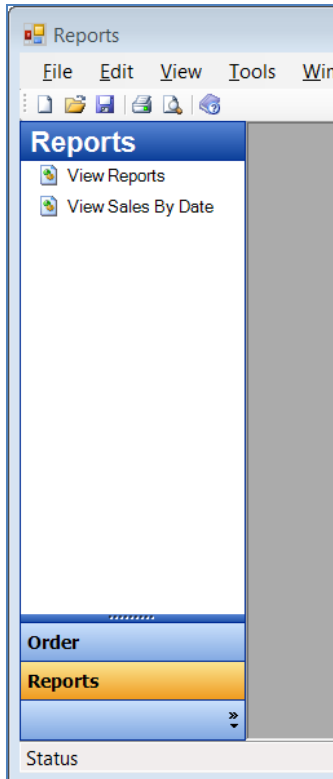


12. Similarly, add another **Reports** group and **View Reports** and **View Sales By Date** child nodes.

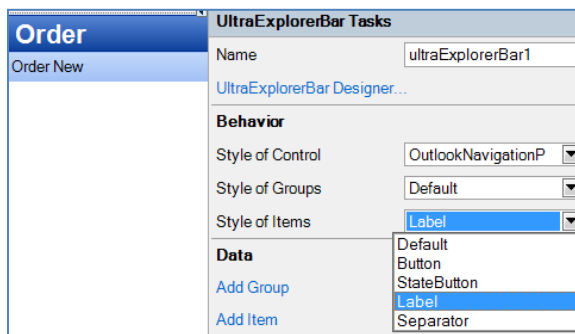


13. Click **Close**. The groups you added are listed and when you click the groups, the corresponding items are listed.





- Open SmartTag of **UltraExplorerBar** and select **Label** in **Style of Items** as shown in the screen below:



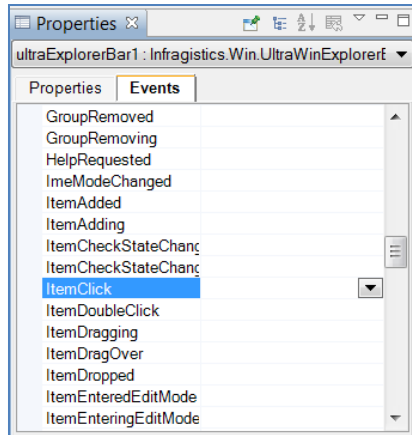
- Save the changes.

6.4 Embedding the ABL windows into an MDI form

This section shows you how to add a code to embed an ABL window and GUI windows designed as part of the above labs into this MDI form.

- Select **UltraExplorerBar** bar in the editor.
- Click the **Events** tab in the **Properties** view. The events available for this control are listed.





3. Double-click the field next to the **ItemClick** event method. The source view opens and the **ultraExplorerBar1_ItemClick** event method is added to the code.
4. Copy and paste the following code to display **SalesReports** and **Graphs** as child forms when specific items are clicked and when **OrderNew** is selected, ABL window **Order.w** is opened as the embedded window. You can alternatively write a code inside the ItemClick method.

```

METHOD PRIVATE VOID ultraExplorerBar1_ItemClick( INPUT sender AS
System.Object, INPUT e AS
Infragistics.Win.UltraWinExplorerBar.ItemEventArgs ):

    DEFINE VARIABLE cKey AS CHARACTER NO-UNDO.
    DEFINE VARIABLE cls1 AS form .
    DEFINE VARIABLE cls2 AS form .
    cKey = e:ITEM:Text.
    cls1 = NEW Graphs().
    cls2 = NEW SalesReport().
    CASE cKey:
        WHEN 'Order New' THEN embedABLWin(e).
        WHEN 'View Reports' THEN ShowChildForm(cls1,e).
        WHEN 'View Sales By Date' THEN ShowChildForm(cls2,e).
    END.

END METHOD.
/*----- ShowChildForm method -----*/

METHOD PRIVATE VOID ShowChildForm(childForm AS
Progress.Windows.Form, e AS System.EventArgs):

    /* Make it a child of this MDI form before showing it. */
    childForm:MdiParent = THIS-OBJECT.
    childForm:FormClosed:SUBSCRIBE(Form_Closed).
    childForm:Show( ).

END.
/*----- embedABLWin method -----*/

METHOD PUBLIC VOID embedABLWin( e AS System.EventArgs ):

```



```

NO-UNDO.      DEF    VAR      childForm AS Progress.Windows.MDIChildForm
NO-UNDO.      DEFINE VARIABLE h          AS HANDLE
NO-UNDO.      DEF    VAR      hwin       AS HANDLE
NO-UNDO.      RUN OrderApp.w PERSISTEN SET h.
              hwin = h:CURRENT-WINDOW.
              /* Create the WindowContainer, embedding the window into it.
              */
              childForm = NEW Progress.Windows.MDIChildForm( ).
              childForm:Size = New System.Drawing.Size( hwin:WIDTH-PIXELS,
hwin:HEIGHT-PIXELS ).
              childForm:Dock = System.Windows.Forms.DockStyle:Fill.
              childForm:EmbeddedWindow = hwin.
              childForm:MdiParent = THIS-OBJECT.
              childForm:FormClosed:SUBSCRIBE(Form_Closed).
              childForm:Show( ).
              RUN initializeObject IN h.

END METHOD.

```

5. Save the file. Warning  marker appear as follows:

```

896 Keywords are in lower case. e = New System.Drawing.Size( hwin:WIDTH-PIXELS, hwin:HEIGHT-PJ

```

6. Press **Ctrl+Shift+F** and observe that the casing of **New** keyword is changed to All Caps.

```

896 childForm:Size = NEW System.Drawing.Size( hwin:WIDTH-PIXELS, hwin:HEIGHT-PJ

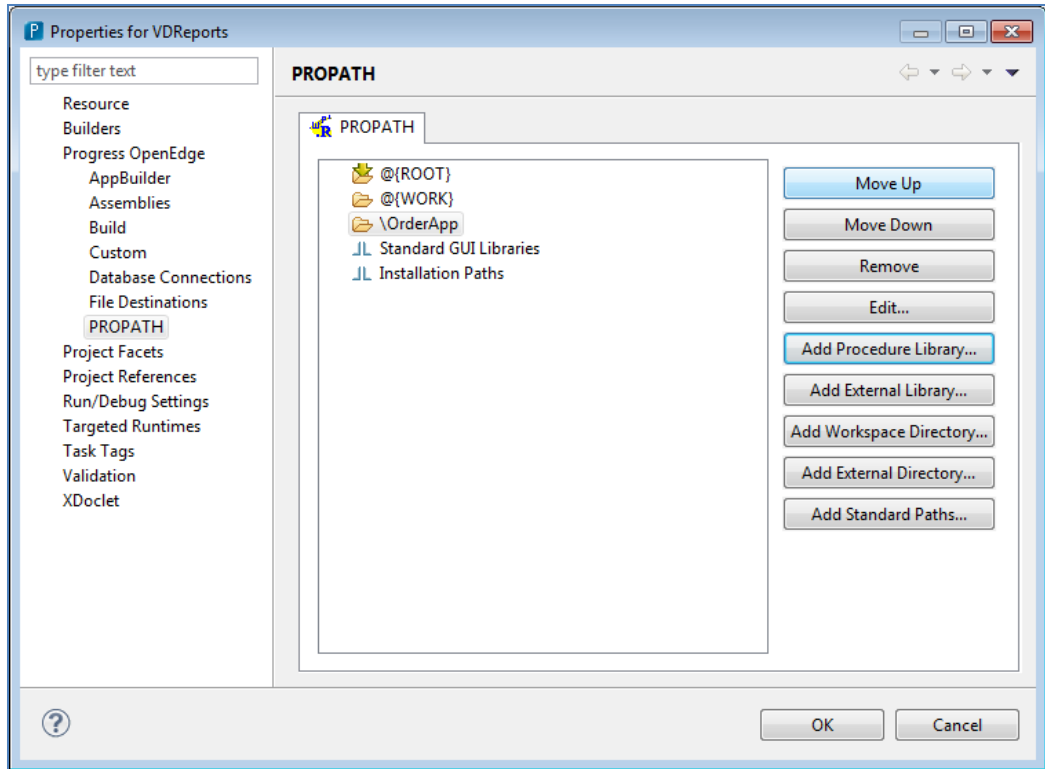
```



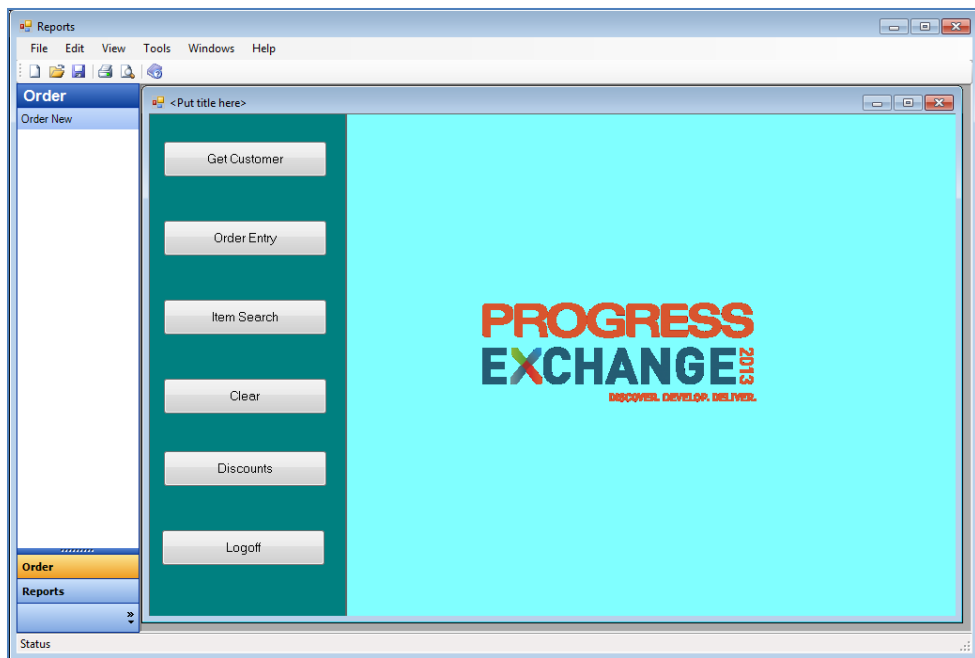
Tip: To set your keyword casing preference, update the preferences in the Editor Preferences dialog box. You can enable ABL Editor to automatically apply keyword casing as you type and when you save the file. Optionally, select a block of code to which you want to apply keyword casing, and press CTRL+SHIFT+F, or select Source > Correct Case. If no text is selected, the entire file is formatted.

7. To access the **OrderApp.w** AppBuilder file from the **VDReports** project, you need to change the PROPATH for the project.
8. Right-click the **VDReports** project and click **Properties**. The **Properties** dialog box opens.
9. Click PROPATH from the left section, and click **Add Workspace Directory**. Add the **OrderApp** directory and click **OK**. OrderApp appears in the PROPATH list.





10. Save the file and run the file as Progress OpenEdge Application.
11. Click **Order New** Item, the embedded form appears:



12. Close the **Run** dialog box.



7 Configuring Apache WebServer (not required for the Workshop)

For the workshop, Apache is pre-configured on the machines that are provided to you. The steps below are provided to help you in configuring Apache WebServer beyond workshop.

1. Install Apache HTTP Server from following location:
<http://httpd.apache.org/download.cgi#apache22>
Win32 Binary without crypto (no mod_ssl) (MSI Installer): httpd-2.2.25-win32-x86-no_ssl.msi
2. Change the listening port from 80 to 8080 in the Listen field; open httpd.conf file from C:\Program Files (x86)\Apache Software Foundation\Apache2.2\conf. Specify 8080. Save the file.
3. Copy the cgiip.exe file from DLC\bin (DLC refers to the install location C:\Progress\OpenEdge) to C:\Program Files (x86)\Apache Software Foundation\Apache2.2\cgi-bin.
4. Copy the WebSpeed directory from \$DLC to C:\Program Files (x86)\Apache Software Foundation\Apache2.2\htdocs
5. Restart the Apache server.



Note: Apache installation directory provided is for sample. In real-time, use the paths where you have installed Apache.



Progress Developer Studio for OpenEdge gives you a robust development environment for various types of application that you build. We continue to add and enhance features in every release with continued focus on the productivity of a developer.

We hope you will give Progress Developer Studio for OpenEdge a try.

