

## Table of Contents

|   |     |
|---|-----|
| Table of Contents .....                                     | 1   |
| License Agreement .....                                     | 3   |
| Trademarks and Copyright .....                              | 4   |
| Creating A Form Based Application.....                      | 5   |
| Printing Hello World.....                                   | 5   |
| Writing Code File .....                                     | 5   |
| Procedural Project.....                                     | 6   |
| Printing Hello World in Console .....                       | 9   |
| Printing Hello World in Desktop Form .....                  | 11  |
| Creating a Complex Program.....                             | 14  |
| Creating Packages And Using Them In Your Applications. .... | 24  |
| Creating Databases And Tables .....                         | 28  |
| Creating Table In A Desktop Form Project .....              | 28  |
| Using Visual Query(PQuery).....                             | 36  |
| Using A Database Project .....                              | 41  |
| Using Database Project .....                                | 44  |
| Using The PGird And PDBGird .....                           | 45  |
| Creating a PGird Object.....                                | 45  |
| Creating a PDBGird object .....                             | 58  |
| Using The Report Builder.....                               | 62  |
| Using PValue, PList, PValueList And PObject Objects.....    | 65  |
| Using PList .....   | 65  |
| Using PValue .....  | 75  |
| Using PValueList .....                                      | 80  |
| Using PObject .....   | 92  |
| Creating A Game Visually.....                               | 96  |
| The Subtleties Of The Visual Programming.....               | 101 |

|   |     |
|---|-----|
| Loops and Conditions: For Loop .....  | 101 |
| Loops and Conditions: While Loop .....  | 110 |
| Loops And Conditions: ForEach Loop .....  | 118 |
| Loops and Conditions: If Condition .....  | 124 |
| Loops and Conditions: Else Condition .....  | 134 |
| Loops and Condition: Else-If Condition.....   | 143 |
| PVariable.....  | 152 |
| PMethod.....  | 157 |
| Creating Your Own Components And Distributing Them .....                            | 163 |
| PPL Source Code .....   | 163 |
| XML Definition Files .....  | 169 |
| Generate the Component Package .....  | 170 |
| More In-Depth Object Binding.....   | 171 |
| PDirectory, PFile And PFileList Example.....  | 177 |
| Displaying the number of files available in a directory with PDirectory Object..... | 177 |
| Getting the file size of a file by using PFile object.....                          | 183 |
| Using PFileList to display the constituents of a Folder .....                       | 189 |
| Presource.....  | 195 |
| Additional Support.....   | 196 |
| Appendix A: Figures.....  | 197 |



## License Agreement

This document and its contents are furnished "as is" for informational purposes only, and are subject to change without notice. ArianeSoft does not represent or warrant that any product or business plans **Expressed** or implied will be fulfilled in any way. Any actions taken by the user of this document in response to the document or its contents will be solely at the risk of the user.

ARIANESOFTE MAKES NO WARRANTIES, **EXPRESSED** OR IMPLIED, WITH RESPECT TO THIS DOCUMENT OR ITS CONTENTS, AND HEREBY **EXPRESSLY** DISCLAIMS ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR USE OR NON-INFRINGEMENT. IN NO EVENT SHALL PROOFHQ BE HELD LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING FROM THE USE OF ANY PORTION OF THE INFORMATION.

Copyright © 2009 by ArianeSoft . All rights reserved.

This document may not be reproduced, photocopied, displayed, transmitted or otherwise copied, in whole or in part, in any form or by any means now known or later developed, such as electronic, optical or mechanical means, without the written agreement of ArianeSoft . Any unauthorized use may be a violation of domestic or international law.

## Trademarks and Copyright

ArianeSoft , Pocket Programming Language, and the ArianeSoft logo are trademarks of ArianeSoft .

All other product or company names mentioned are used for identification purposes only, and may be trademarks of their respective owner.

## Creating A Form Based Application

Creating applications with PIDE is as easy as well as efficient process. PIDE implements visual programming and allows its users to create better applications with minimal programming. Visual programming not only makes it easier to create programs just by using a few simple drag and drop operations but also allows creation of faster and much robust applications.

Whether you are a novice programmer or someone who is well qualified in creating computer applications, you will be more than happy to create applications with PIDE. Visual programming is an effective way to create computer applications without much of an effort. Using the already created templates for easy creation of application programs, PIDE users can easily create applications within seconds. In the first section of this tutorial, we will see how easy it is to create easy as well as complex applications with PIDE.

## Printing Hello World

### Writing Code File

By using the code file project in PIDE2, users will gain the required experience and knowhow of coding in PIDE. This experience can be transferred to other types of projects in PIDE which will enhance the type of projects that a user can accomplish through PIDE.

- Start by creating a new **Code File** project. To do so, start PIDE and press **Ctrl+N** and select **Code File Project** from the “**New Project Type..**” window. Alternatively, you can also go to **File> New** and select **Code File** from the “**New Project Type..**” window.

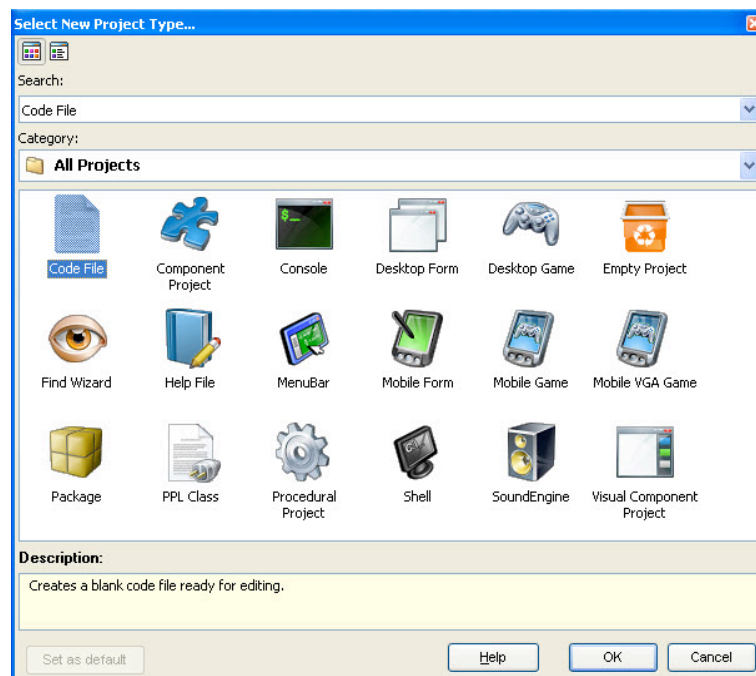


Figure 1: Creating a new project

- For creating a procedural project, users will need to use different methods and functions present in PIDE2. For printing a “**Hello World**” example in the code file project, use the **ShowMessage()** method.

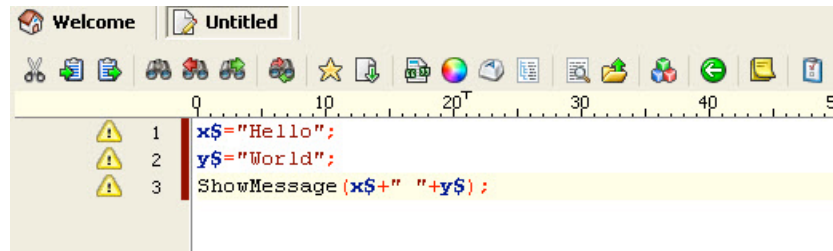


Figure 2: Code

- In the above code, we have used two variables namely variable **x** and variable **y**. Both of these variables have been assigned with string values namely, “**Hello**” and “**World**”. The **ShowMessage()** method allows a user to print a message within a standard dialog box and takes a string or a reference of a string to be printed in it. In the above example, we have included variable **x** as well as variable **y** with a string of a single space (“ ”) in between them to print **Hello World**.
- Press **F5** or Go to **Run-> Run** to execute the code. After the code is executed, you will see ‘**Hello World**’ written in a alert message box.

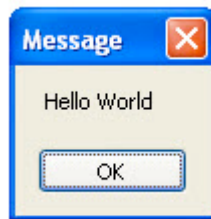


Figure 3: Output

## Procedural Project

Using procedural project creates a code based project where a programmer can write his own code (manually) in the **MainLibrary** object, he/she can add methods (which are procedures and functions only). But it is almost the same as writing a code file. Given below are the steps to create a procedural project in PIDE:

- Open PIDE2 and create a new **Procedural** project. To do this, press **Ctrl+N** from your keyboard and select **Procedural** in the **Select New Project File Type..** Window.

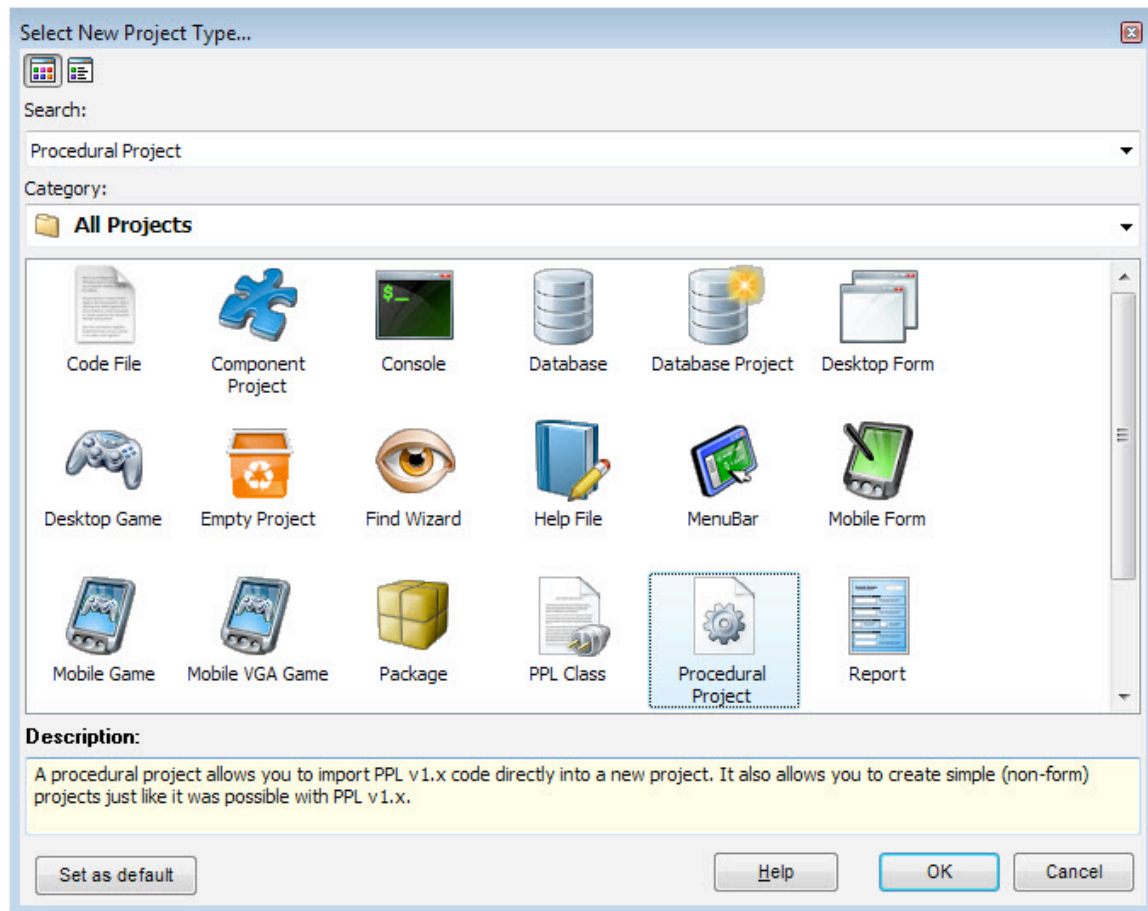


Figure 4: New Procedural project

- Just like any other programming language, we can add procedures in PIDE. Now we will work in the main library. For our *Hello World* message, we will add the **ShowMessage** command but use it within a **Proc Main** parent this time.
- To write our code, double click the **MainLibrary**. This will open a code editor that can be used to write code in it.

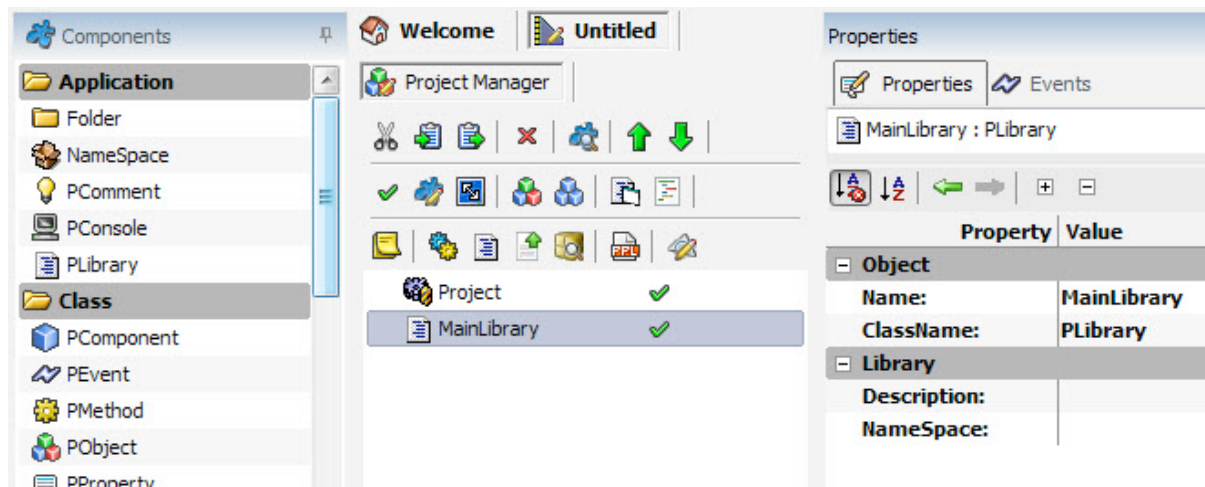


Figure 5: Main Library

- In the code editor, write the **ShowMessage** command as in the screenshot and save the file.

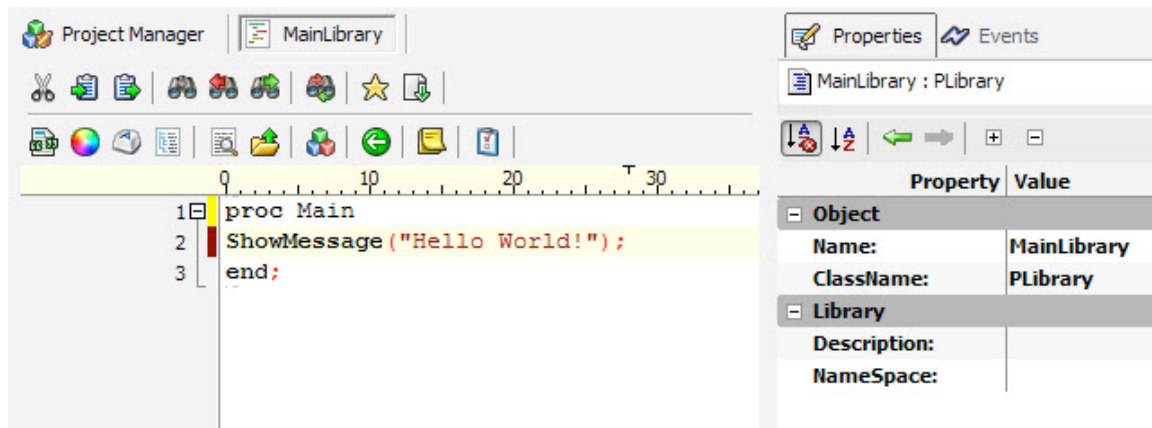


Figure 6: Hello World Code

- Now, we can run our project by pressing **F5** or by going to **Run** in the main menu and choosing the **Run** option.

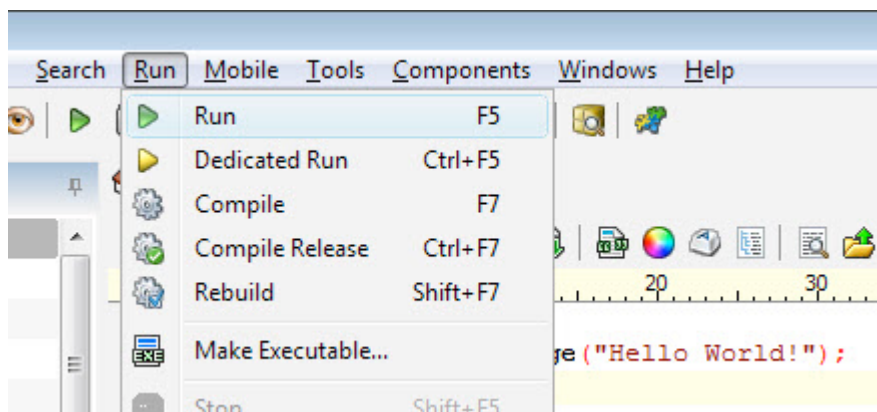


Figure 7: Run Hello World

Figure 8: Run the project

- After the project is executed you can see the results!

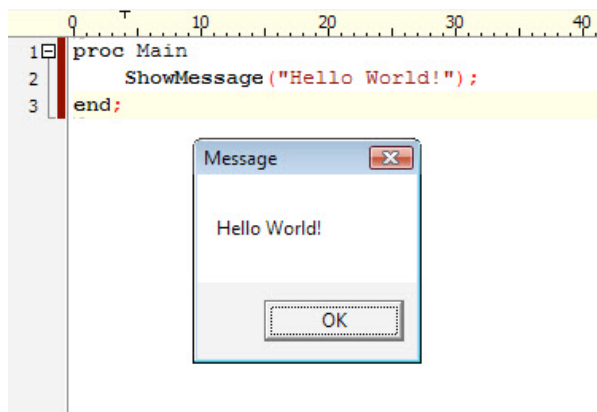


Figure 9: Output



*Enclosing strings within quotes is a common way of determining strings in PIDE and this distinguishes variables from strings.*

## Printing Hello World in Console

Printing text in console is really simple. Follow the steps given below to print “Hello World” in PIDE console window.

- Start by creating a new **Console** project. To do so, start PIDE and press Ctrl+N and select **Console** Project from the “**New Project Type..**” window. Alternatively, you can also go to **File> New** and select **Console** from the “**New Project Type..**” window.

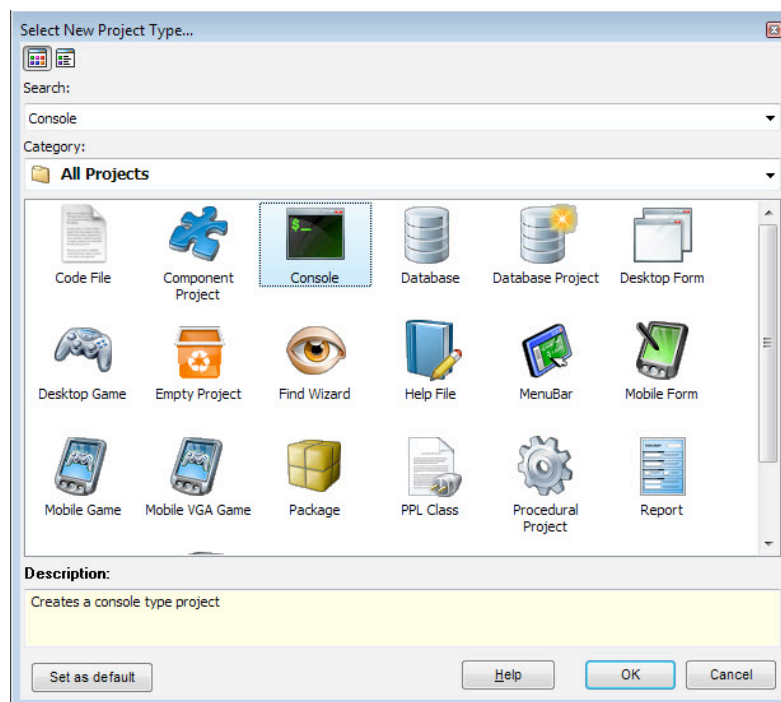


Figure 10: Create a new project

- Once the new project opens, double click the **MyApplication** object to create a new **OnStart** event. Right-click the **MyApplication** object and select **Events**, finally, choose **OnStart**.



Figure 11: OnStart event

- Click the **OnStart** procedure and press **Ctrl+Space** key to bring the code-completion window. Select **Writeln** in the **Code Completion** window.

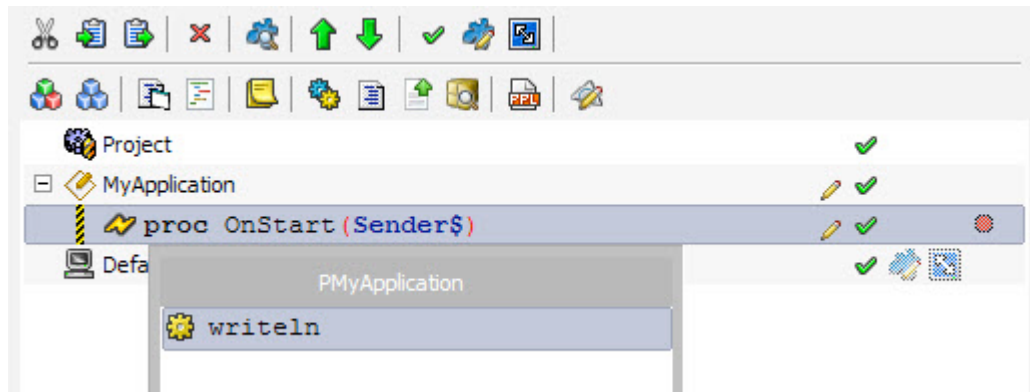


Figure 12: Code Completion window

- Once selected, click or press enter key on the **Writeln** object and change its **Expr** property to “**Hello World**” including the double quotes.

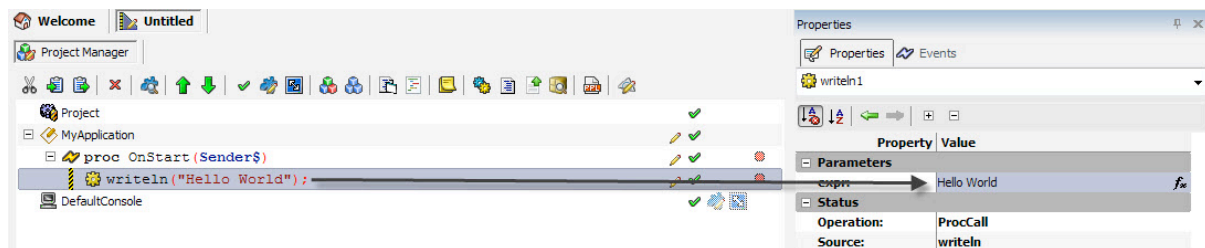


Figure 13: Expr property

- Press **F5** to run your project.

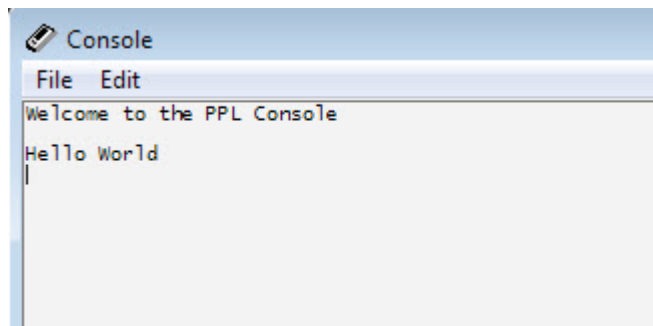


Figure 14: Output



*Almost all objects in PIDE can be double clicked to generate default event. For example, PButton creates an OnClick event when double clicked while MyApplication created OnStart.*



## Printing Hello World in Desktop Form

**Desktop Form** project can be used by users of PIDE to create a form like interface for their applications. **Desktop Form** makes it utmost easy to create applications that can run in the windows based environment. Making the full use of visual programming, users can create applications with ease. In our **Desktop Form**, we will be creating a button that would print “**Hello World**” in a label, when clicked.

- Create a new **Desktop Form** by selecting **Desktop Form** option in the “**Select new project type...**” window. For doing this, you can press the **Ctrl+N** key or Go to **File>New**. In the **Select New project Type...** window that appears, select **Desktop Form** project.

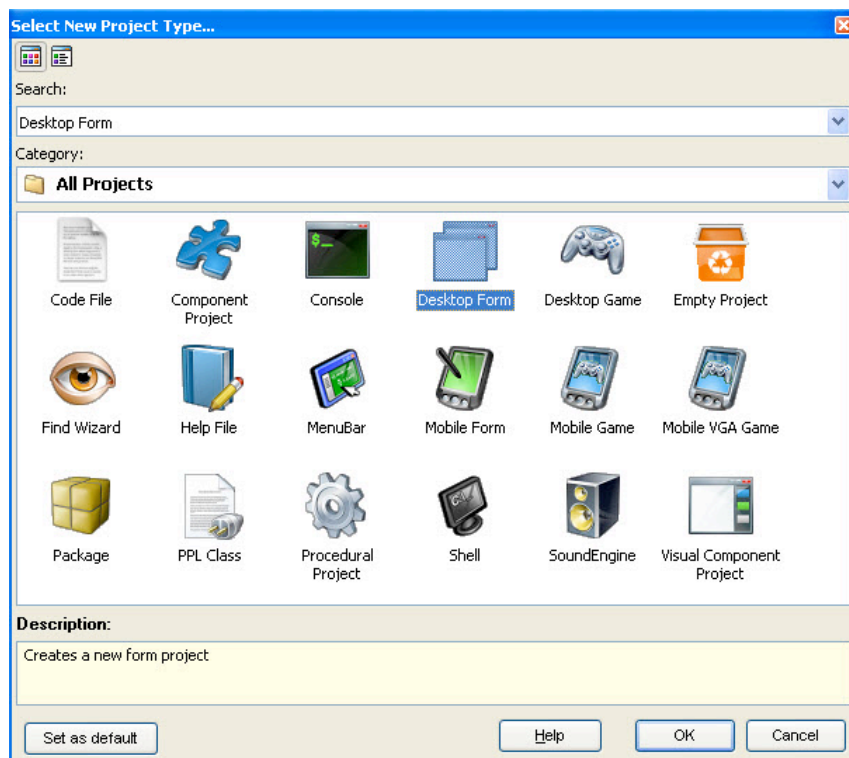


Figure 15: New Project

- Now, in the **Desktop Form**, right click the **Default Form** object to select **Edit Form** or **Press F12** while it is selected or double-click the **DefaultForm** object to bring the **Form Edit** screen.

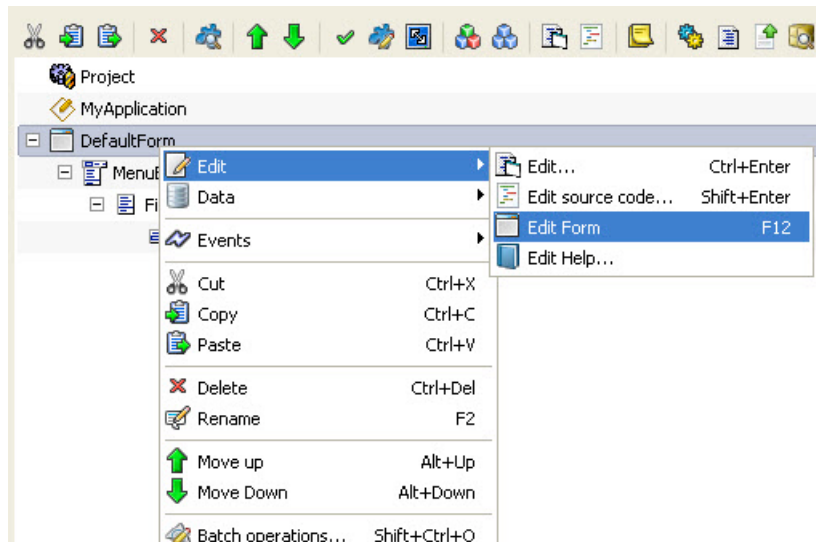


Figure 16: Right click to Edit Form

- In the edit form view, create a **PButton** object by double clicking it from the **Components Panel** and rename the **PButton** to “Click” by changing its **Caption** property.

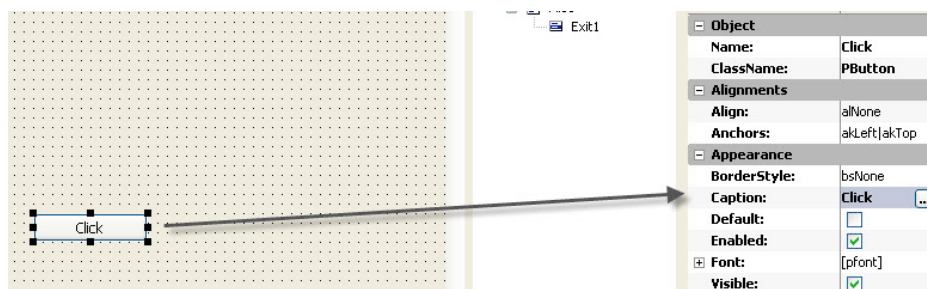


Figure 17: Caption Property

- Right click the button and use the **Edit** option to open **DefaultForm** in the code view.

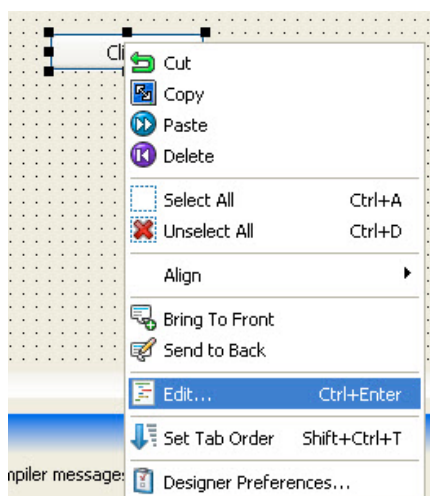
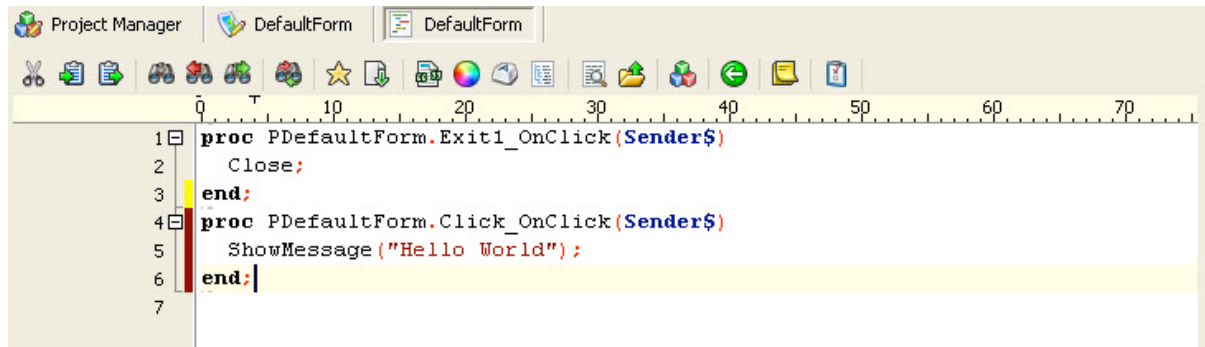


Figure 18: Right click to open DefaultForm

- Create a procedure pertaining to the **OnClick** event of **Click** button and include **ShowMessage("Hello World")** within its body. Right-click the **DefaultForm** object and select **Events**, finally, choose **OnClick**.



```

1 proc PDefaultForm.Exit1_OnClick(Sender$)
2   Close;
3 end;
4 proc PDefaultForm.Click_OnClick(Sender$)
5   ShowMessage("Hello World");
6 end;
7

```

Figure 19: Code

- Press **F5** to run the project and view the results. Use the **“Click”** button to view **“Hello World”** message.



*It is beneficial to learn shortcuts in PIDE as they save time and effort. Quick one's like Ctrl+N for a new project and Ctrl+I for marking an IF statement is a nifty addition to PIDE's environment that goes a long way improving programming experience.*

## Creating a Complex Program

For creating a complex program, we will create a windows form which will allow a user to enter his details and display the details in a pre-formatted manner after the form is submitted. In this example, we will use form elements like **PButtons**, **PLabel**, **PEdit**.

- Create a new **Desktop Form** project by going to **File> New> Desktop Project**. Alternatively, you can also press **Ctrl+N** to create a new **Desktop Form** project.

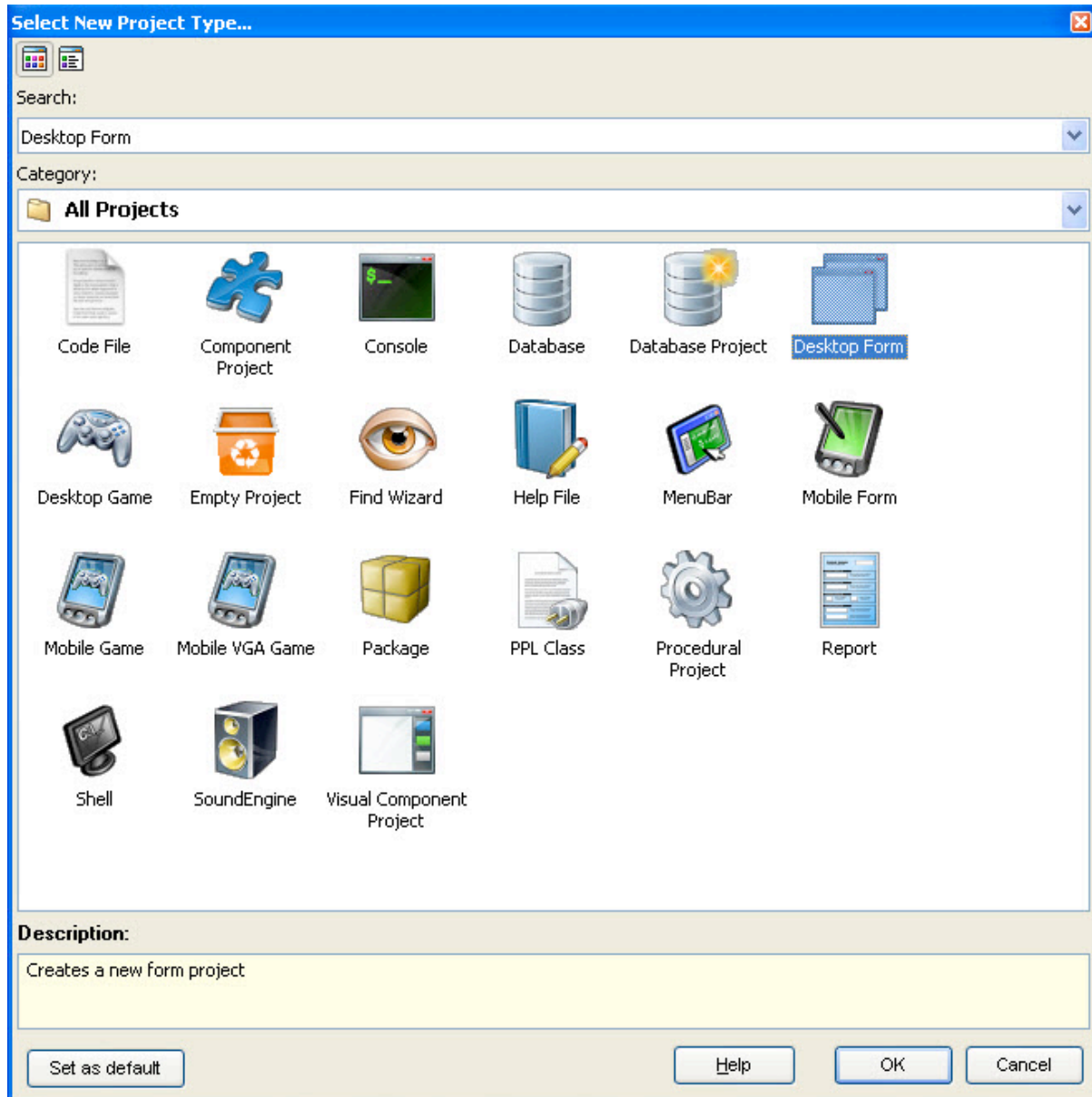


Figure 20: New Project

- In **Desktop Form** project, Right click the “**Default Form**” and select “**Edit**” Option. Alternatively, you can also double-click the **Default Form** option to open **DefaultForm** in edit mode.

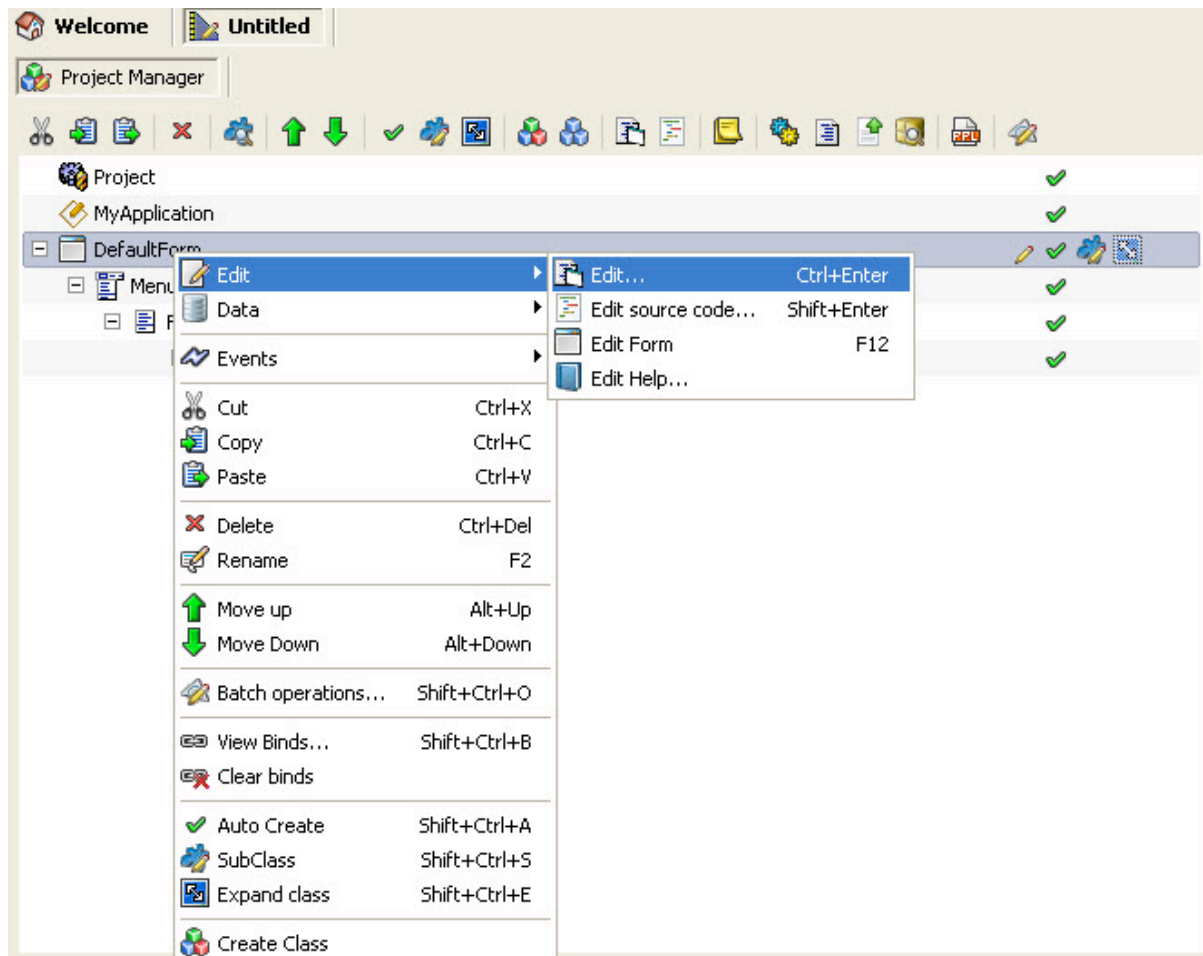


Figure 21: Right click to Edit Form

- Once in the **DefaultForm** Edit view, you can easily create a form by taking form elements from the **Components Panel**.
- Start by clicking the **PLabel** component from the Component Panel by using the area in the form where you want to display it. Along with the **PLabel** component, you can also include **PEdit** components by clicking the individual elements and then clicking the area in the form where you want to display them. We will also have a **PListBox** component in our form, so click on the **PListBox** component in the component pane and click on the area beside a label.

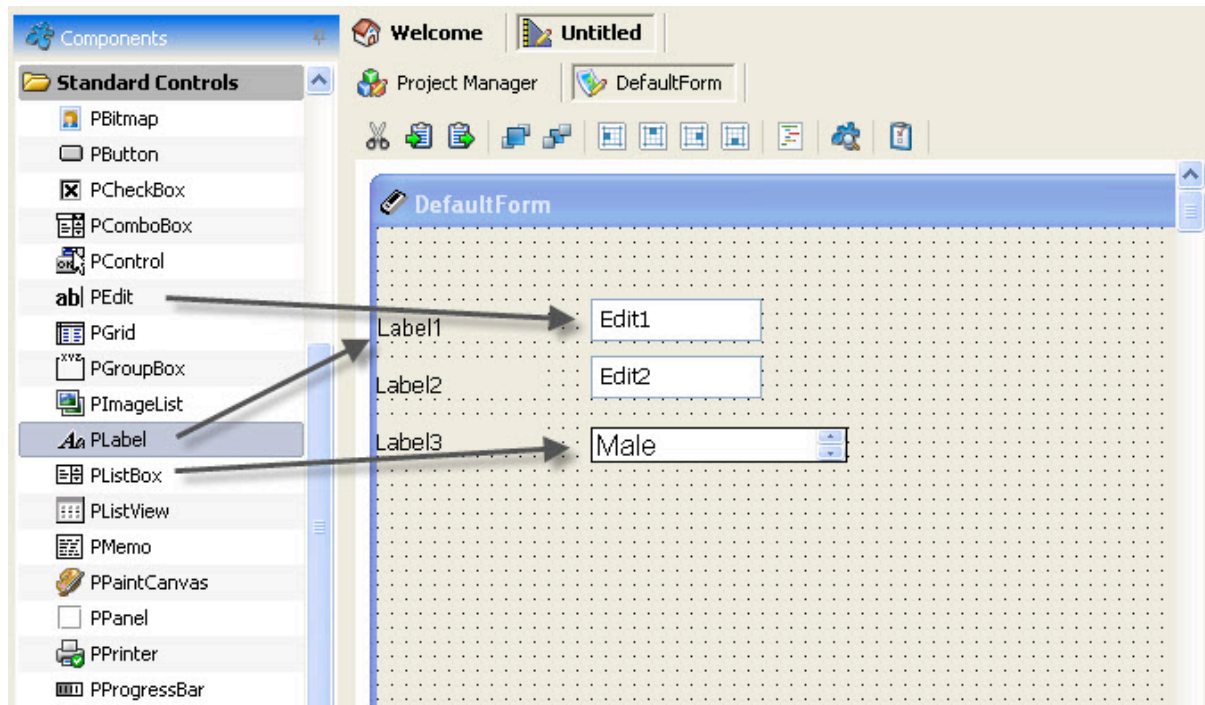


Figure 22: Place components

- To make our application more user friendly, we will change the Caption property of our **PLabel** components so that a user can recognize them. While we are at it, we can also create a caption or heading for form by changing the caption property of our form.

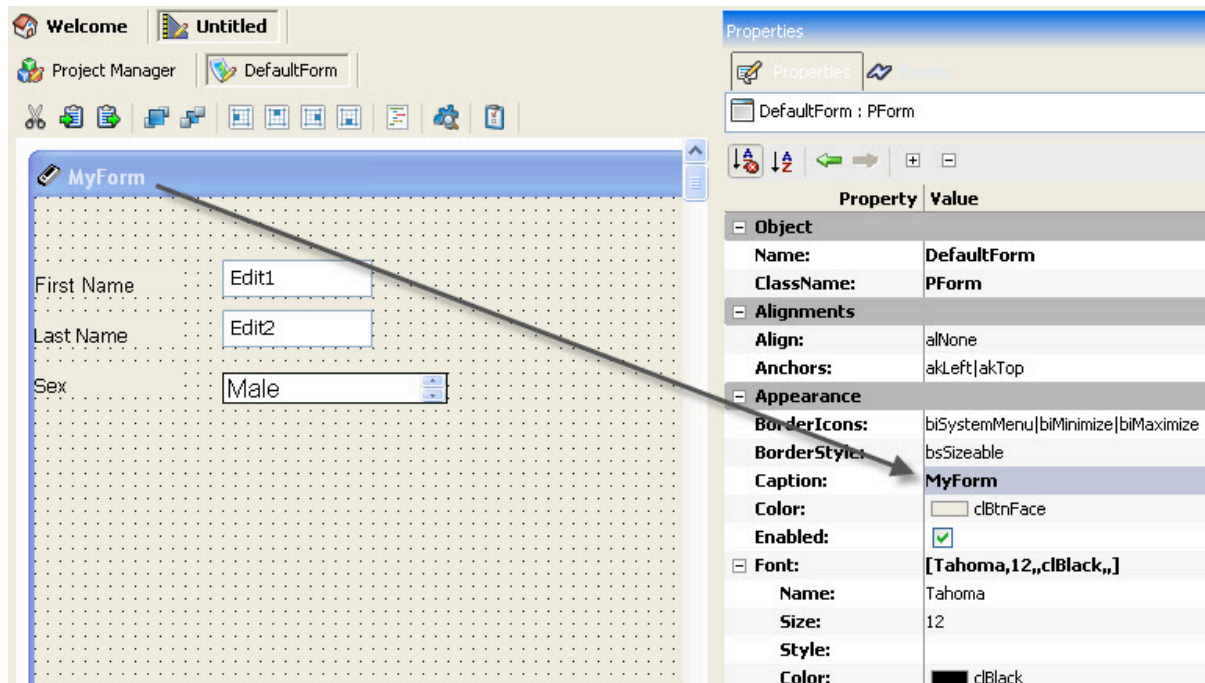


Figure 23: Change Caption property

- We will also create a **PLabel**, mark it with **Age** caption and correspond it with a **PEdit** object so that we can input the age of a person.



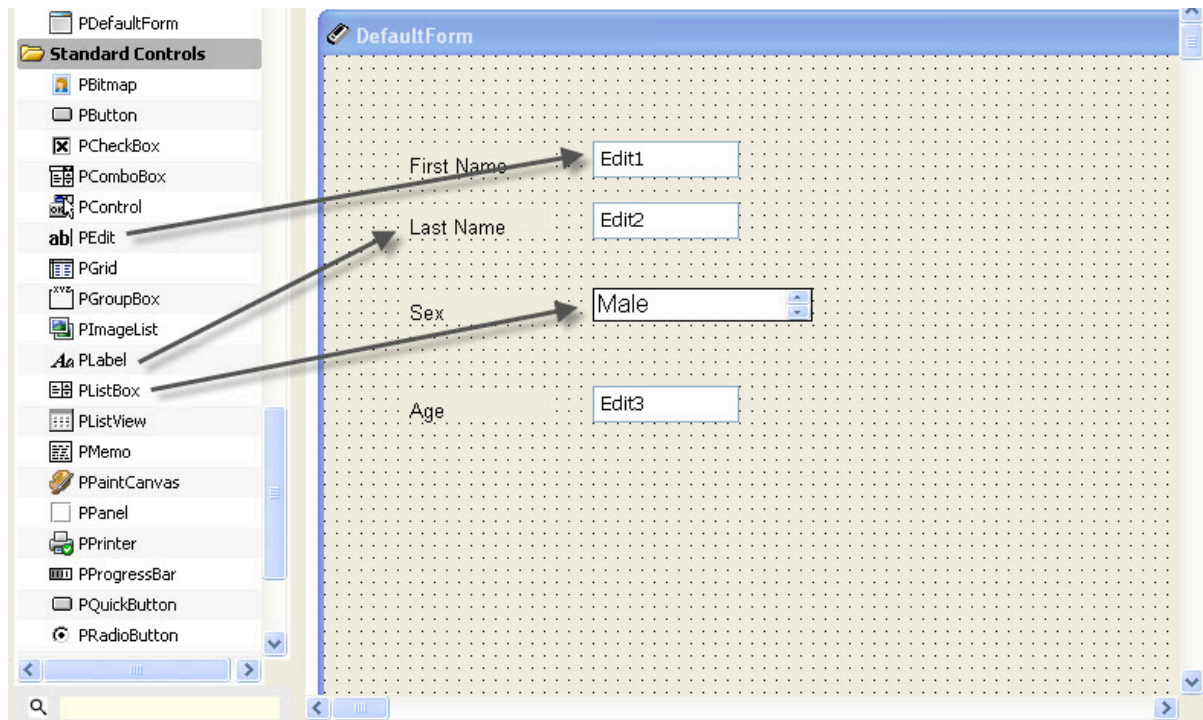


Figure 24: Final placement

- Lastly, we will add a **PButton** and set its caption property to **Submit**. A click on this **PButton** will lead to the processing of the given information.

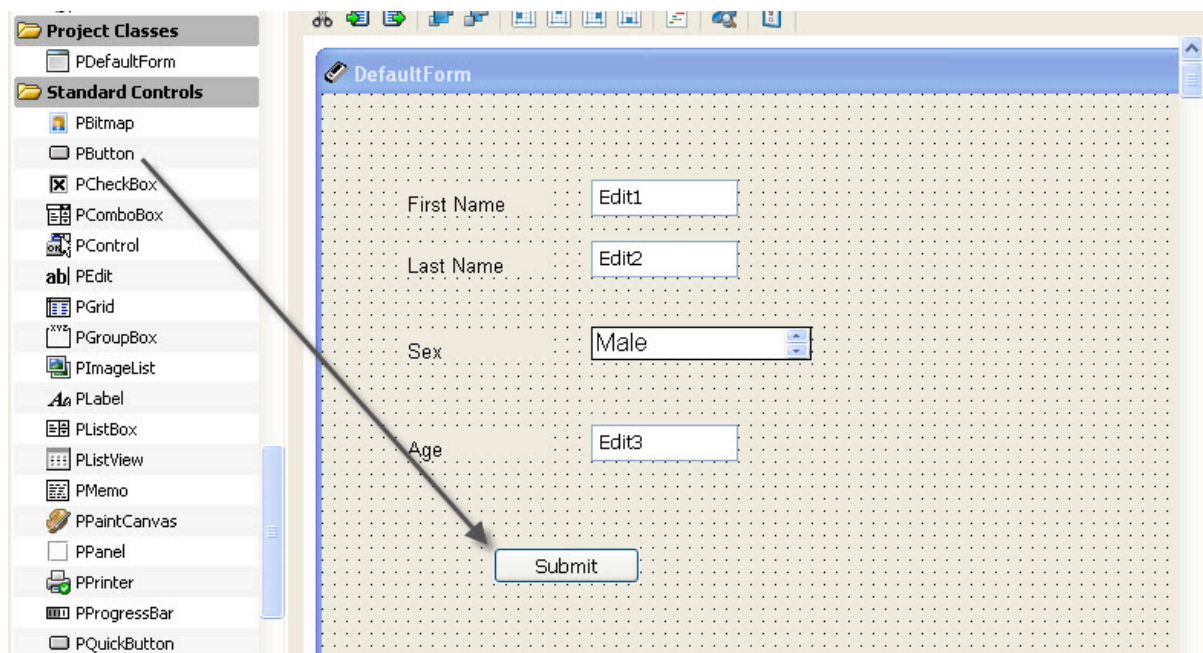


Figure 25: PButton

As the form is now complete, we will now look at how to submit various input values so that we can print the required output.





- Drop the **Edit1** to the **Expr** property of newly created variable and select the **Text** from the auto code complete box. Doing this will assign the text written in the **PEdit1** object to the variable.

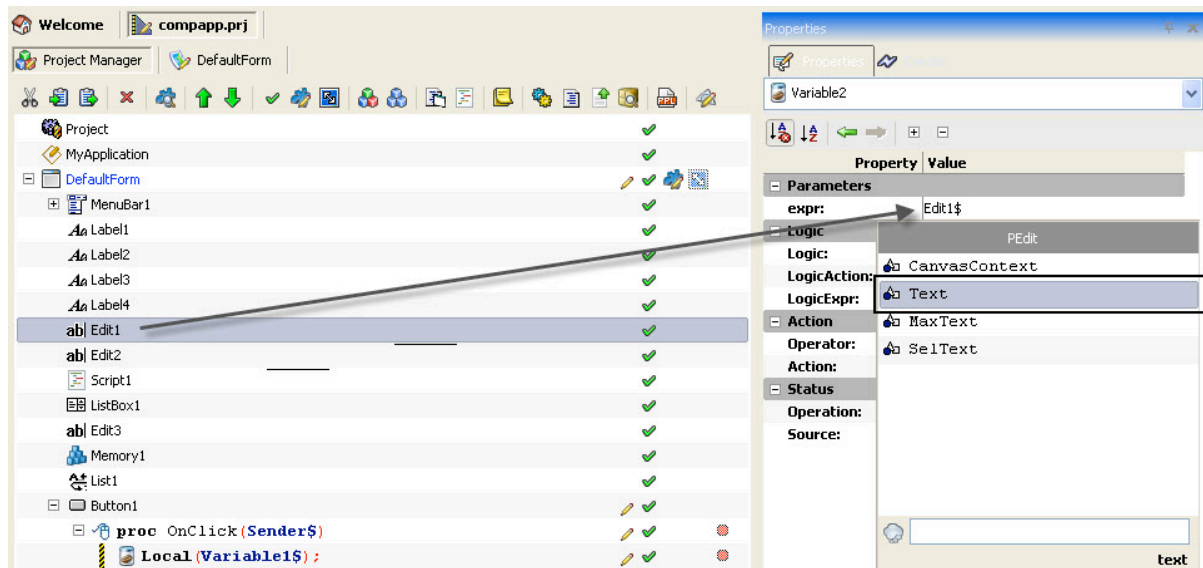
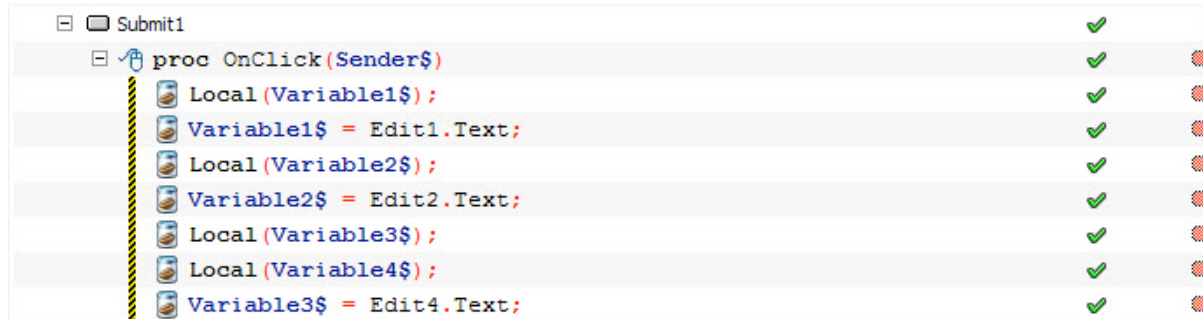


Figure 29: Drag PEdit to expr

- Do likewise for other **PEdit** objects also. After all the variables are created, make sure you have all the **PEdit** objects in the respective **Expr** property of the declared variables like you did with the first **PEdit** objects.



- For the **list box**, drag a new **PVariable** to the **OnClick** event of the button and then drag the newly created Local variable again to the **OnClick** event while holding the **Alt** key. This would create a new variable.
- Drag the **ListBox** object to the **Expr** property of the newly created variable and select **ItemText** from the code complete window.



*If anytime while programming, you feel that you are redoing some steps, you can also select the concurrent statements by pressing the Shift key and copy the statements. Then paste them to create more such statements.*

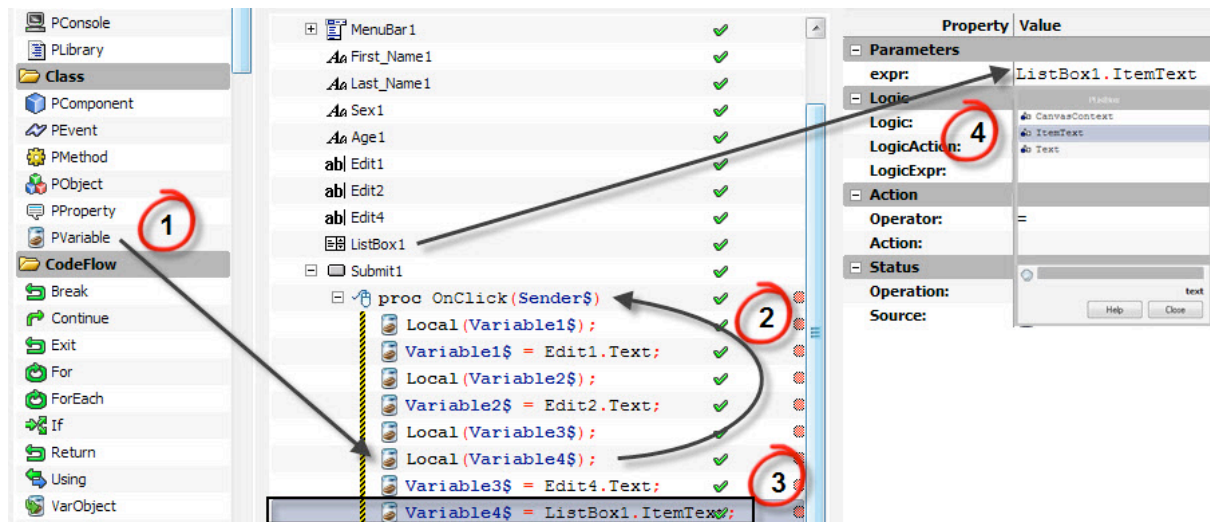


Figure 30: Steps from 1 to 4

- Now, we will have an **If** condition on the **PEdit** objects that inputs the age of a user. This **If** condition will display a custom message depending on the age entered by a user.
- For creating an **If** condition on **PEdit4**, drag and drop **PEdit4** on **OnClick** proc. As the **Code Completion** window appears, select **Text**. This will create an **Edit4.Text** entry. Click on the entry and press **Ctrl+I** (This will make it an **If** condition).

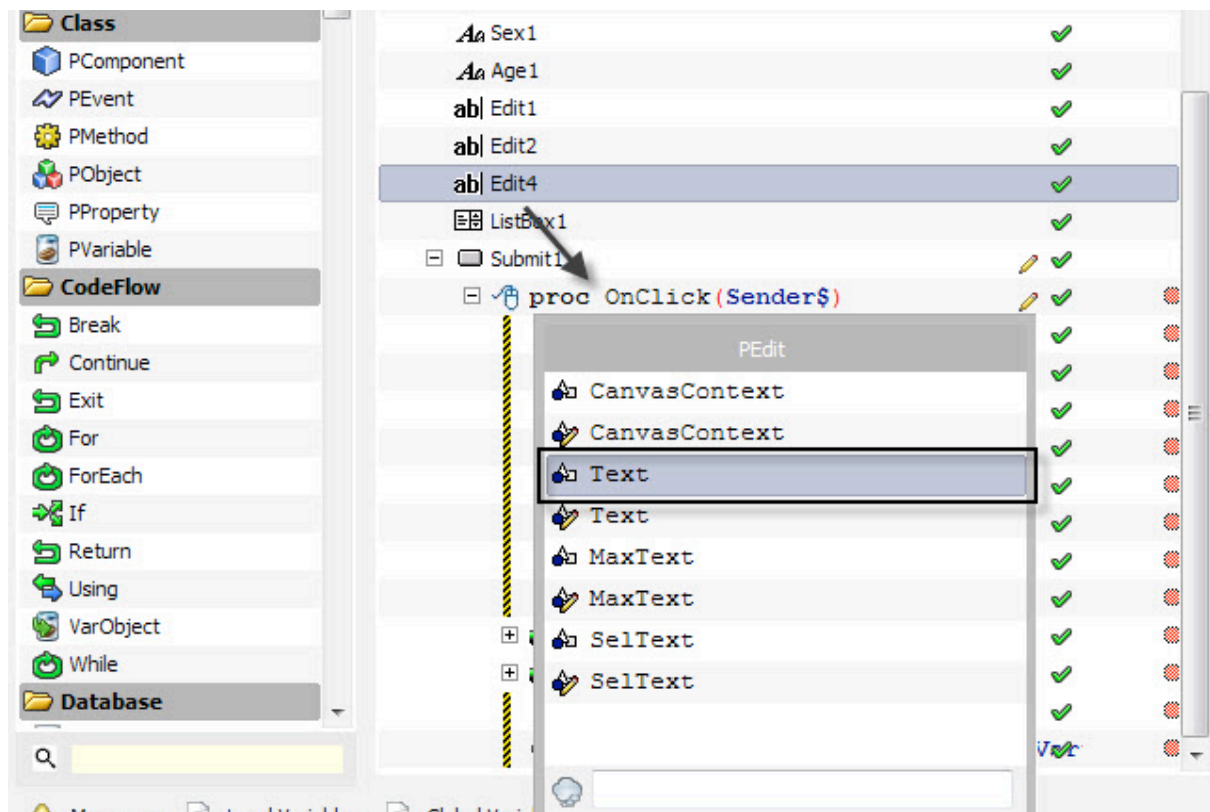


Figure 31: Drag operation

- Change the **LogicExpr** property of the **If** condition to **<20**. Doing this will check whether the age input by the user is less than 20.

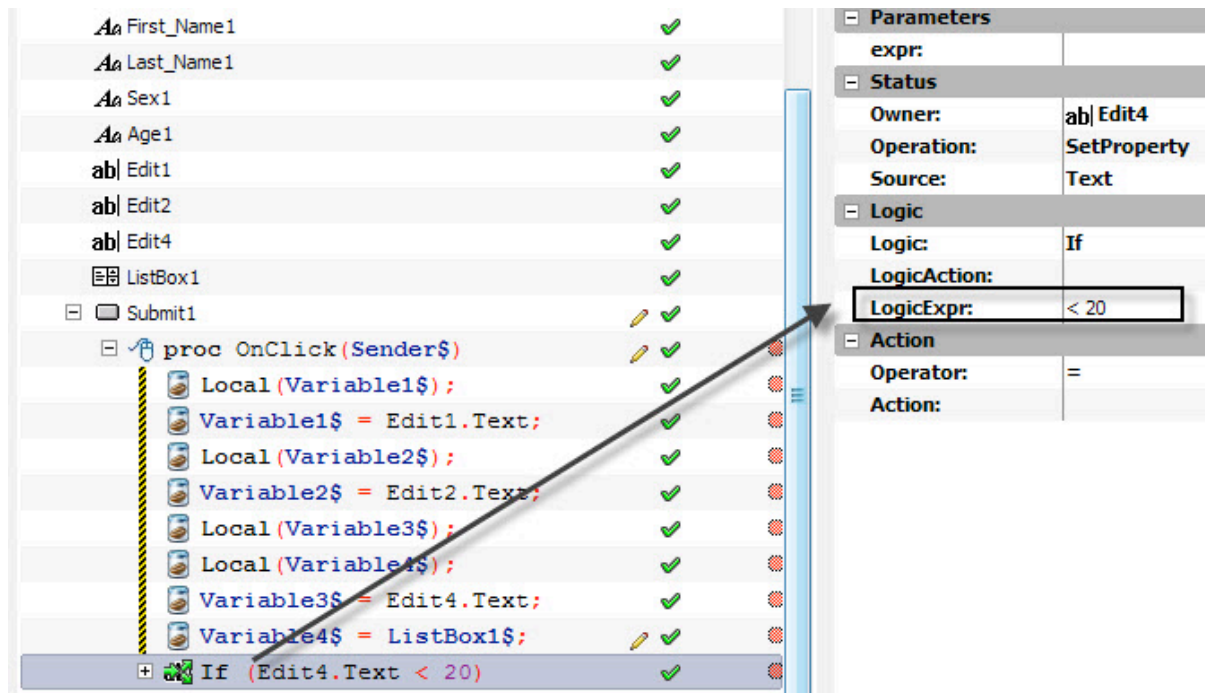


Figure 32: Change property

- If our **IF** condition holds true, we will print “*eligible for the program*”. For doing this, create a new variable by dragging and dropping a **PVariable** to the **OnClick** Proc. Now drag and drop the newly created variable back to the **OnClick** procedure while holding the **Alt** key. Once a new variable is created, drag this variable on the **If** condition. In this variable change the **Expr** property to “*eligible for the program*”. Doing this will enable you to display this message if the condition holds true.

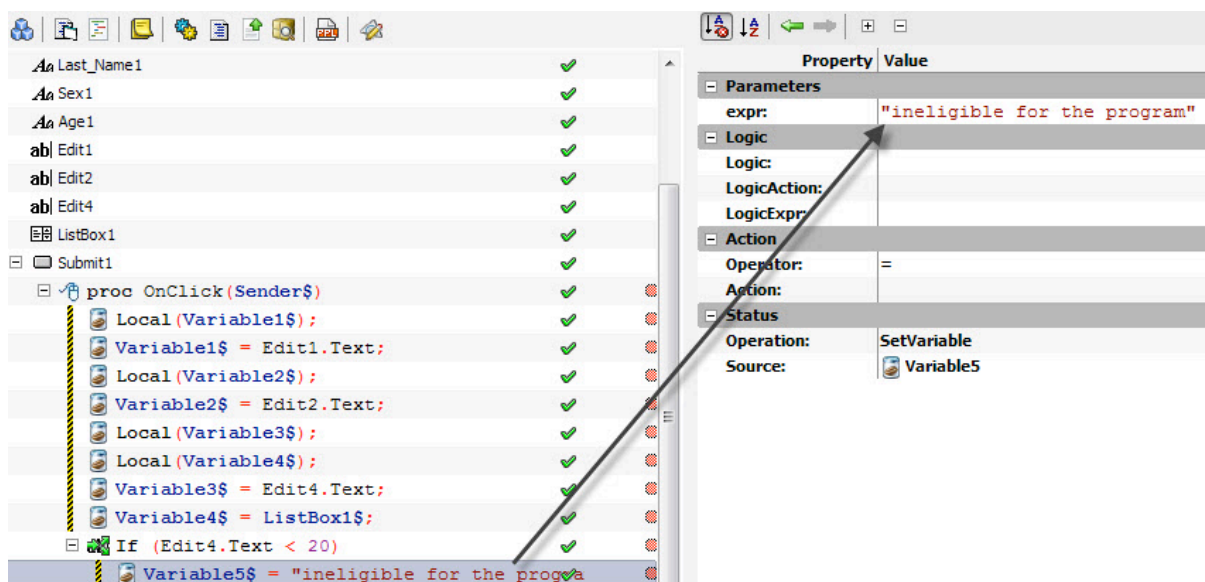


Figure 33: Change Expr property

- Just like the If condition we created in the step above, we will create one more If condition that would check if the age entered by a user is smaller than 20 and display a message accordingly.

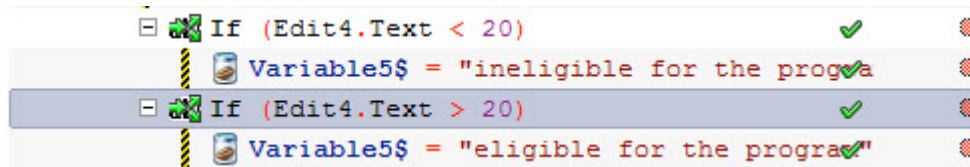


Figure 34: Condition

- Now that everything is done, all we need to do is create a dialog box that would print the message in a preformatted way. We would use **ShowMessage()** for this. Go to **OnClick** proc and press **Ctrl+Space** to open the **Code Completion** window. In this window, select **ShowMessage()**, and change its value to the variables you want to display. For our example, we will use **(Variable1\$+" "+Variable2\$+" is a "+Variable4\$+" and is "+Variable5\$)**.

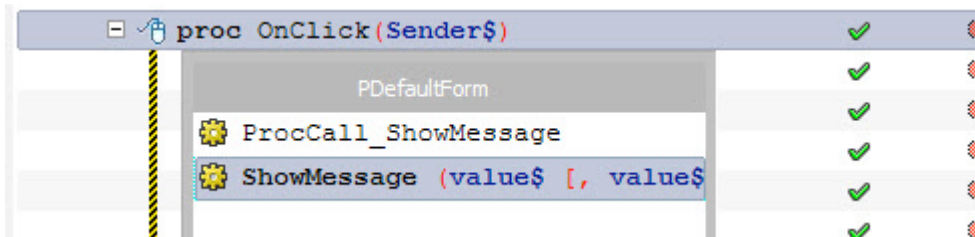


Figure 35: ShowMessage in Code Completion window

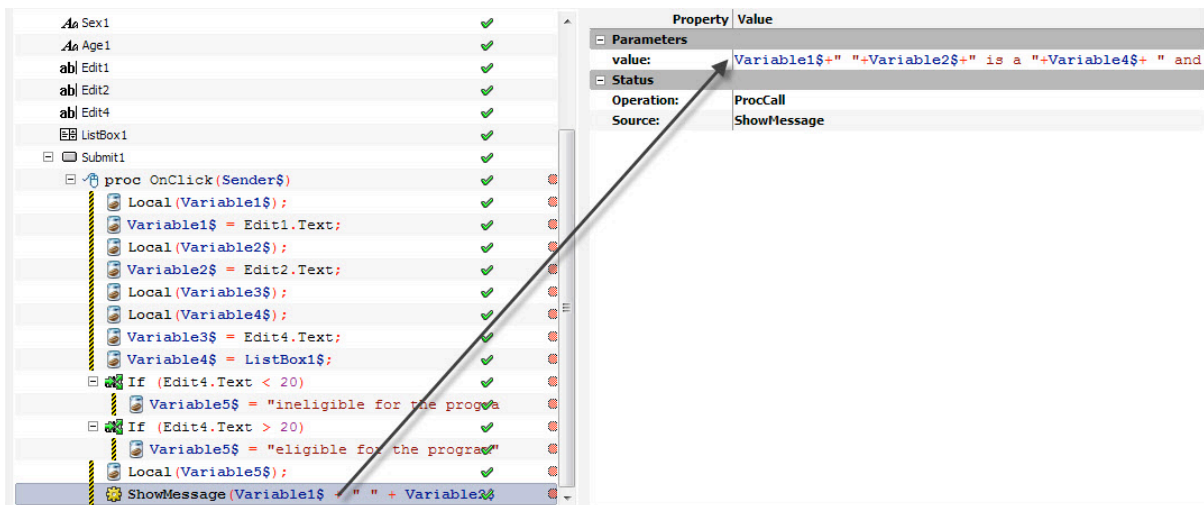


Figure 36: Change Value property

- Now, run the project and see it in action.

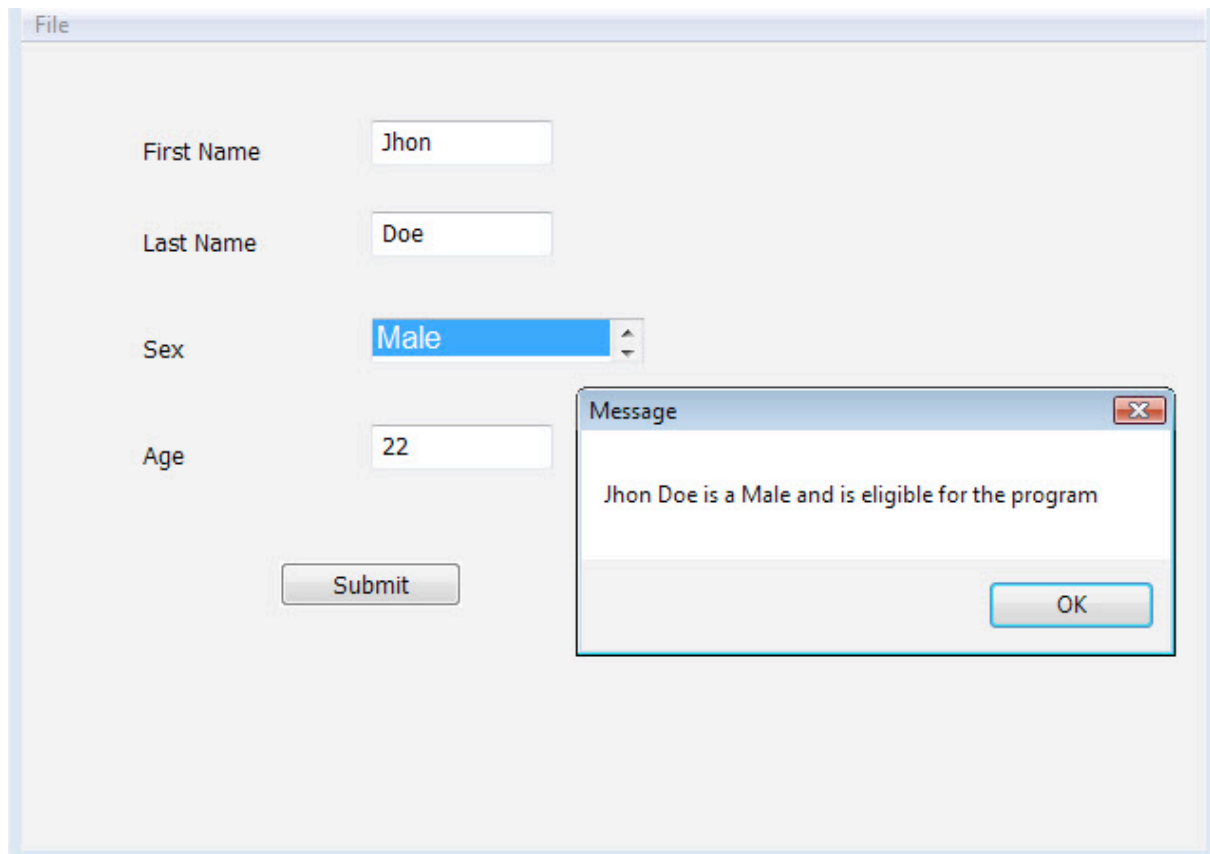


Figure 37: Output



*In PIDE, you can have variables as well as strings in a same statement by using + sign and enclosing the strings with quotes. For example “**How are you today**”+Name\$+”?” may be printed as **How are you today chuck?***



## Creating Packages And Using Them In Your Applications.

Packages are important part of a PIDE project. Users can use PIDE to create applications with various sources like pictures, videos, graphics etc that are stored in a single file. A package will contain all the resources that a PPL application will contain. Given below are the steps to create a package so that it can be used in your PPL project.

- Create a new **Desktop Form** project by selecting it from the **Select New project Type...** window. For this press **Ctrl+N** key or Go to **Files>New**.

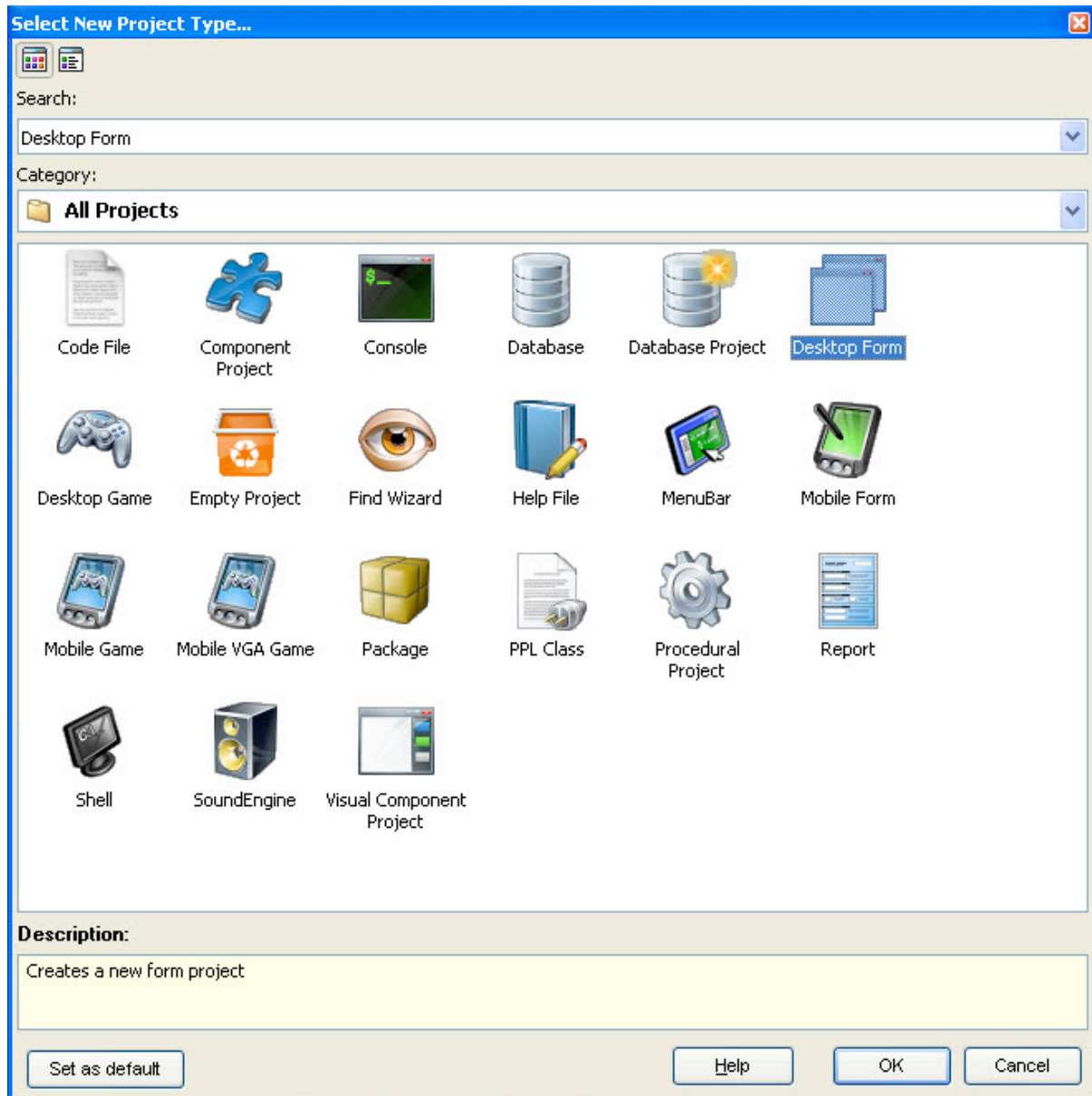


Figure 38: Create New Project

- Drag a **PPackage** object to the project area.

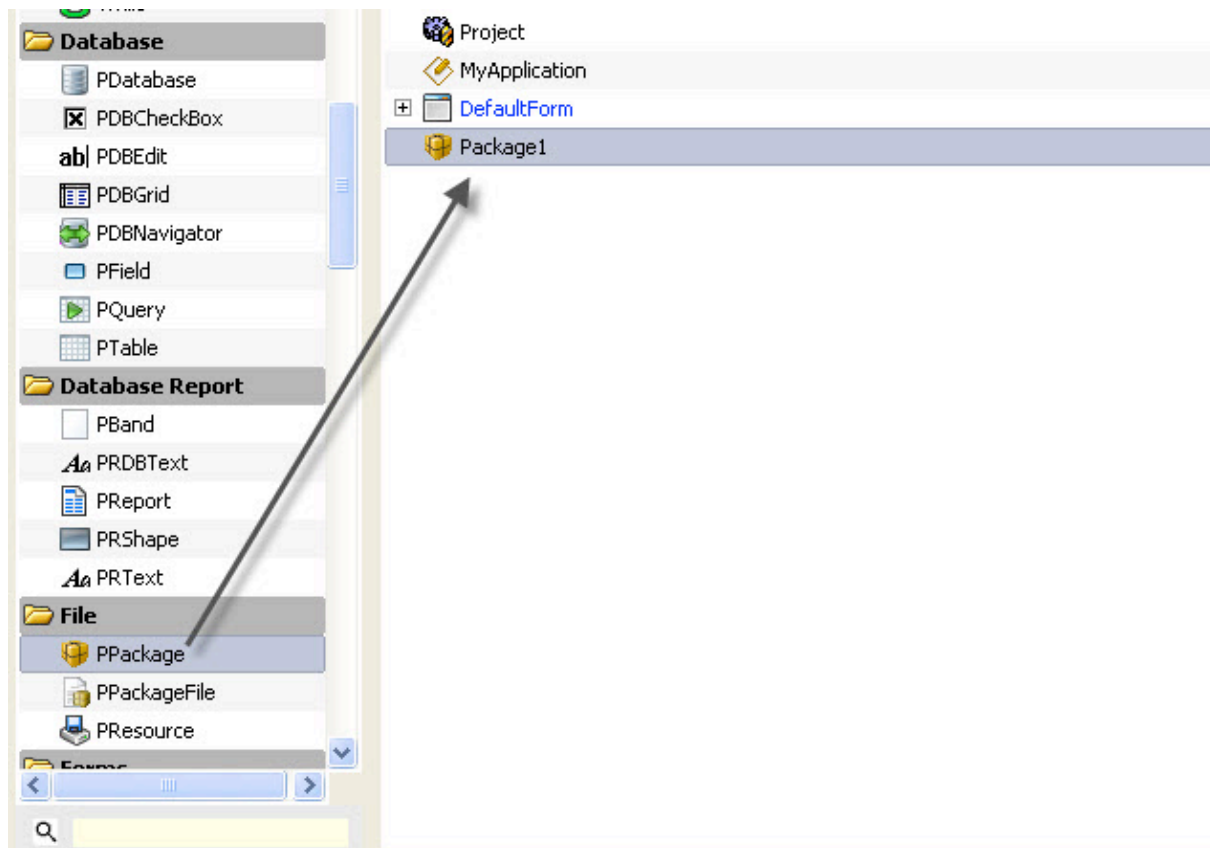


Figure 39: Drag and Drop PPackage

- Give a target location in the **PackageFilename** property of **PPackage** object

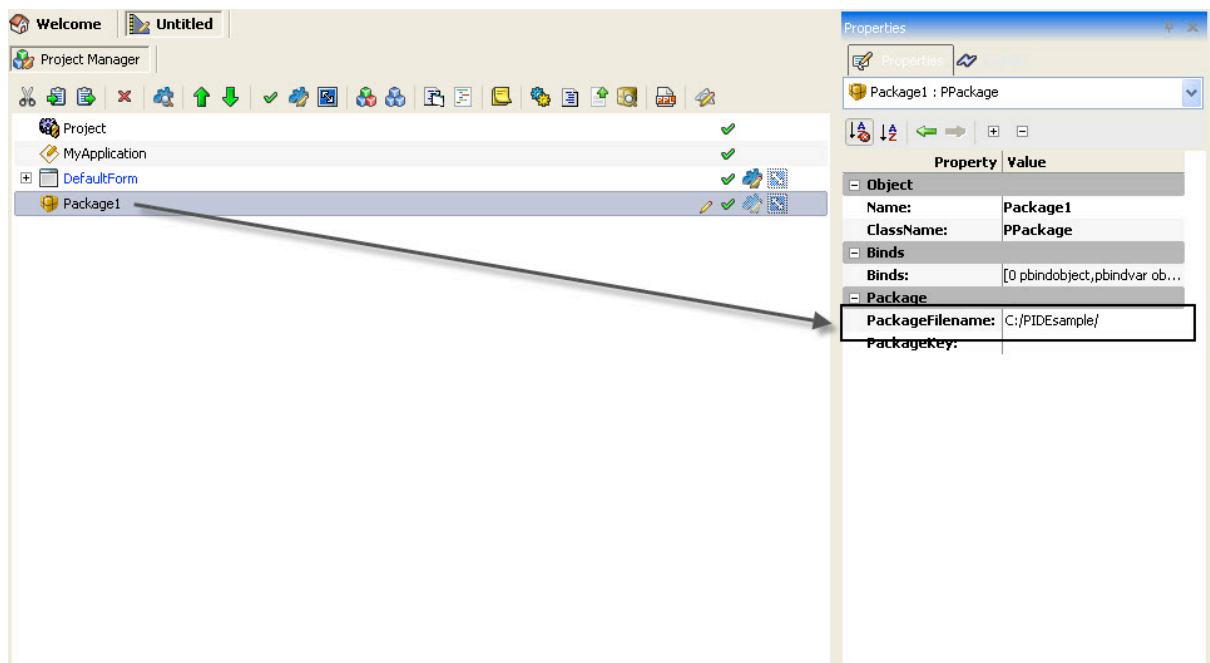


Figure 40: Change Property

- Specify a package encryption key in the **PackageKey** property under package section

- Drag a file from the windows explorer to the **PPackage** object and select the component type as **PPackageFile**. Likewise, drag and drop as many files as you wish to the **PPackage** object and choose the **PPackageFile** option in the component type drop down box.

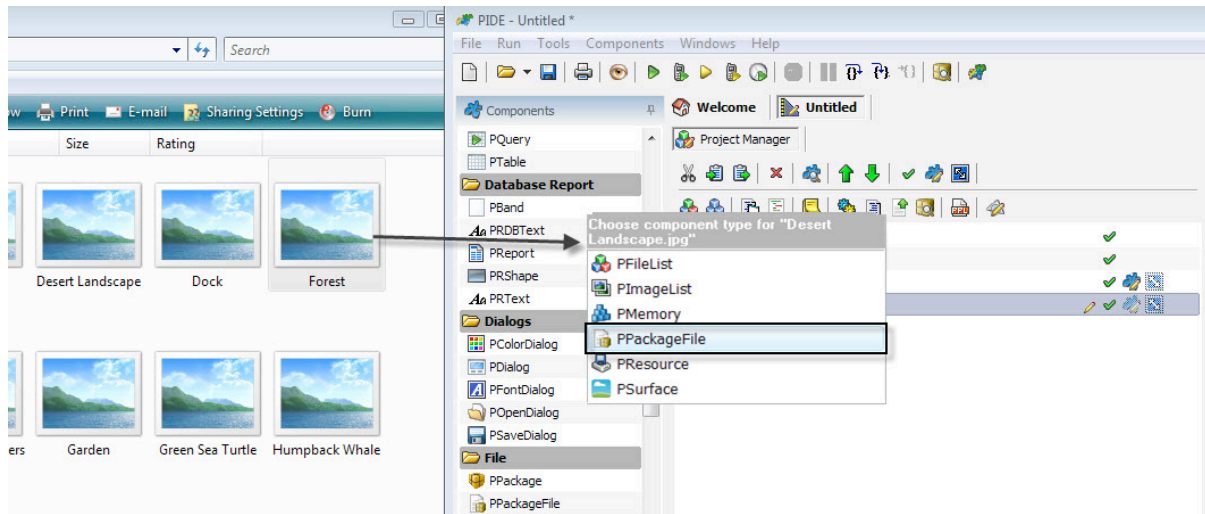


Figure 41: Drop Files

- Using the “+” sign will expand the package and show all the constituents of the package.

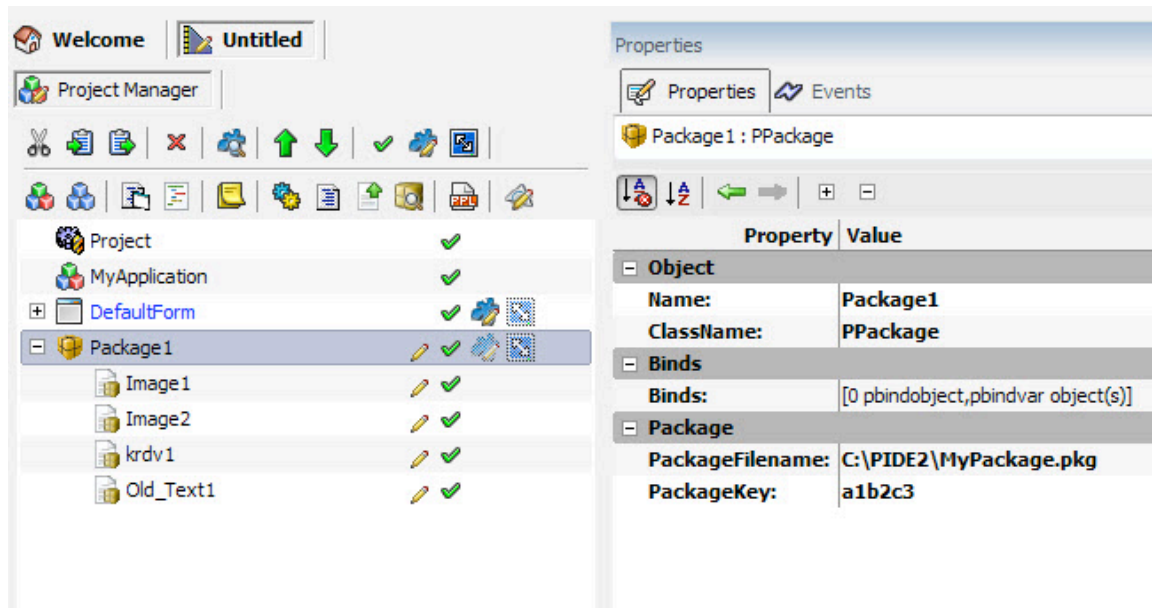


Figure 42: Expand for constituents

- After all the files have been added to the package, a user needs to compile the project. Go to the **Run** menu and click on the compile option. Alternatively, users can also use the **F7** key to compile the project. After a project is compiled, the package file of the project is recreated at the location where it was specified in the **PackageFilename** property.



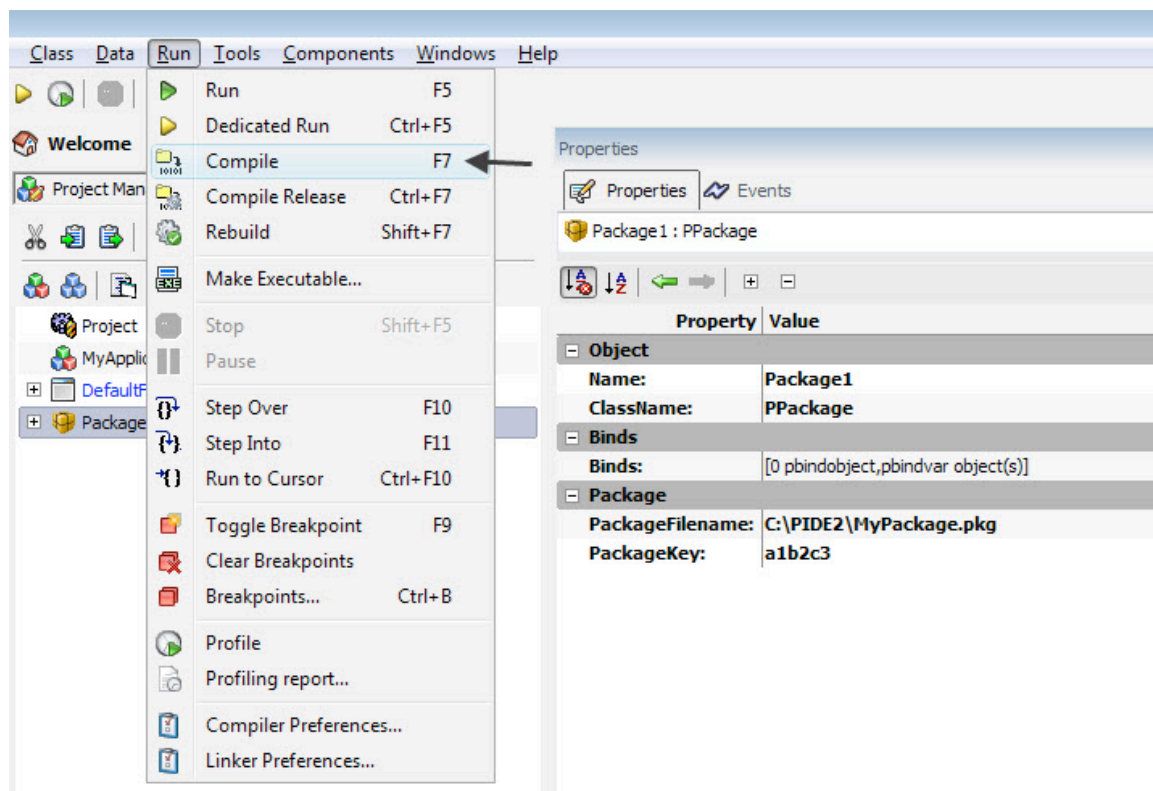


Figure 43: Run the Project

## Creating Databases And Tables

Creating databases and tables with PIDE is a really easy job. A couple of drag and drop operations is all you need to create your database driven applications.

There are three options to build a database with PIDE, users can either use a database, a database project or a **Desktop Form** with a database attached to it.

**Database** - This project comes just with a database, tables and fields. Using this database, you can add a database to your existing projects.

**Database Project** – This type of PIDE project comes complete with a database, tables, query as well as a report. It also includes a **Default Form** object so that users can start right away with their database enabled forms.

### Creating Table In A Desktop Form Project

A **Desktop Form** is one of the most useable applications of PIDE. With a **Desktop Form**, not only does it becomes very easy for a person to navigate through the application, it is easier to understand and also serves the exact purpose why a **Desktop Form** is made in the first place. By combining a **Desktop Form** with database, programmers are able to create applications that can have dynamic nature and can work really well with data.

Given below is the procedure to create a table within a **Desktop Form** and use it to store values.

- Create a **New Project** in PIDE by selecting **File>New** from the file menu and select **Desktop Form** from the **Select New Project Type...** Alternatively, you can also press **Ctrl+N** to bring the **Select New Project Type...** window.

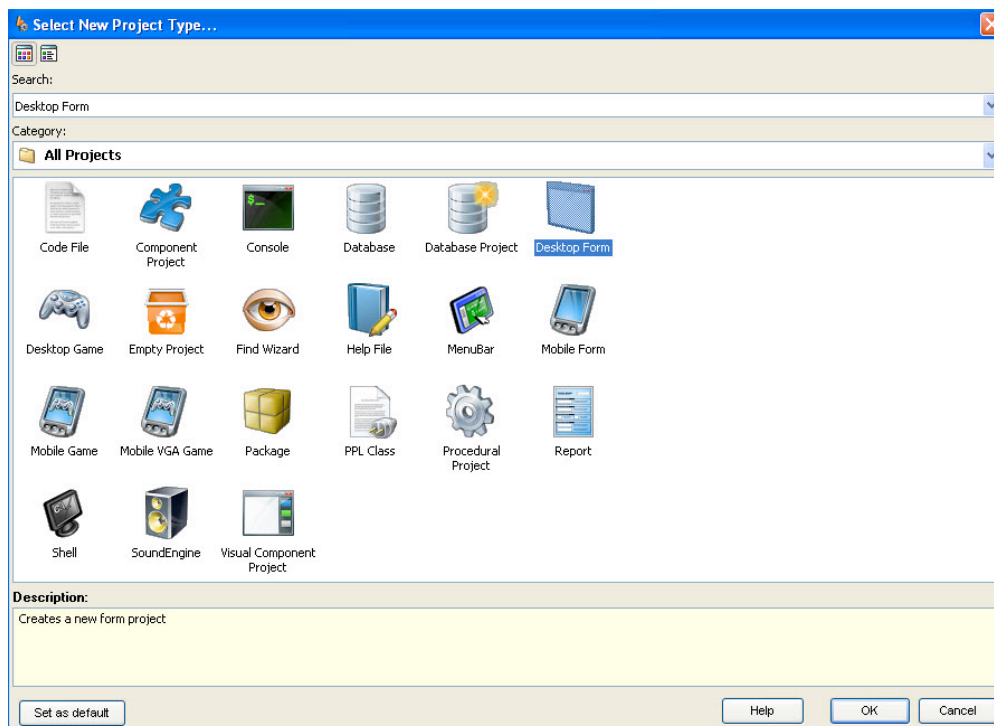


Figure 44: Creating new project

- Select a **Desktop Form** project and open it in the **Project Manager** view
- Once a new **Desktop Form** project opens, drag **PDatabase Object** from the **Components Panel** to the **Project Manager**

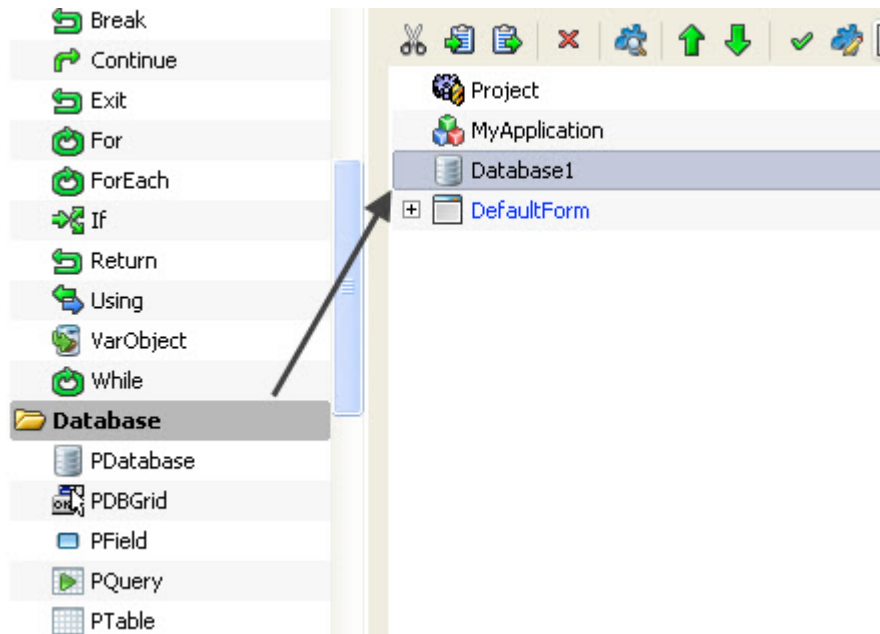


Figure 45: Drag a database

- Drag a **PTable Object** from **Components Panel** to the newly created **PDatabase Object**

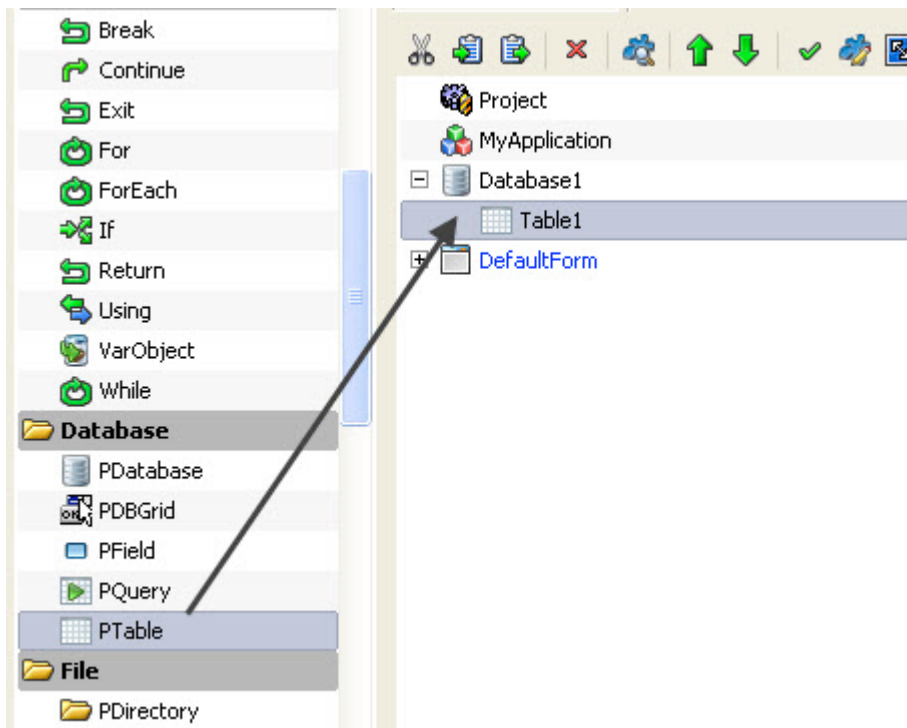


Figure 46: Drag a PTable

- Click on the **PDatabase Object** and set its filename property to a location where you would like to store the database you are creating. Remember to save this file with a **.db or .sdb extension**

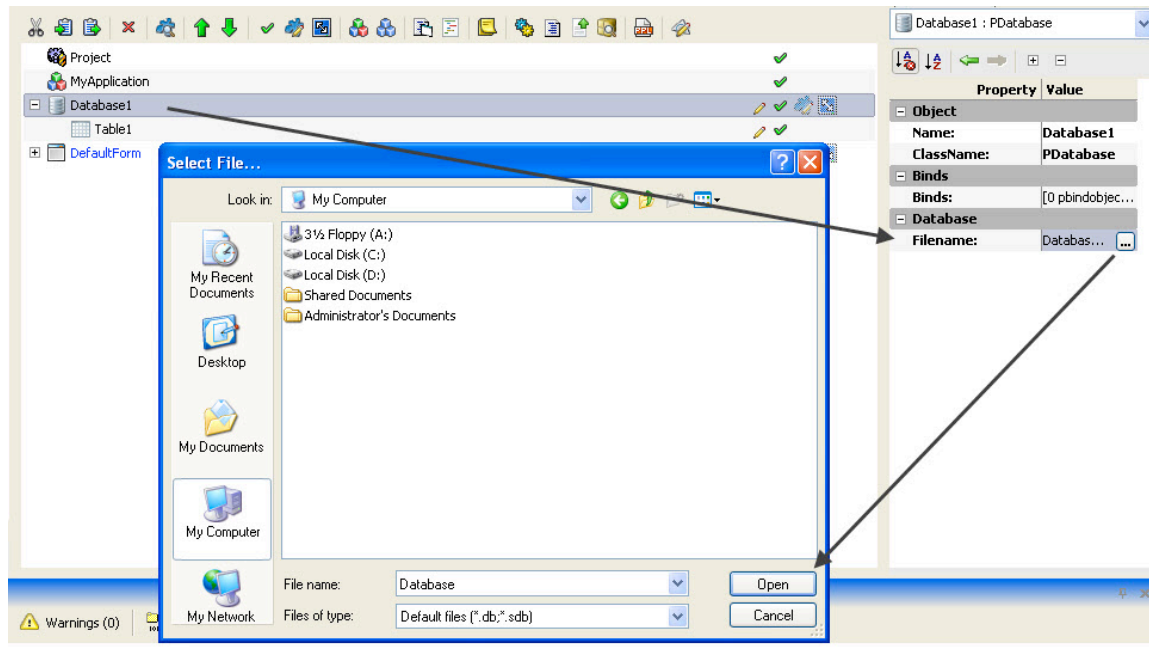


Figure 47: Specify a filename

- Now, in the **TableName Property of PTable Object**, give a name to your table

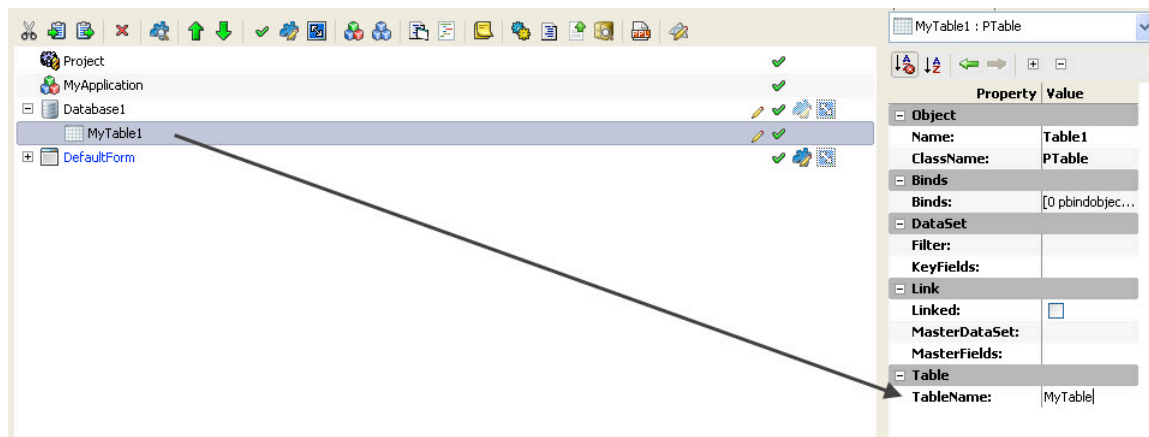


Figure 48: Give TableName

- Drag some **PField Objects** to the table object to add fields to it.

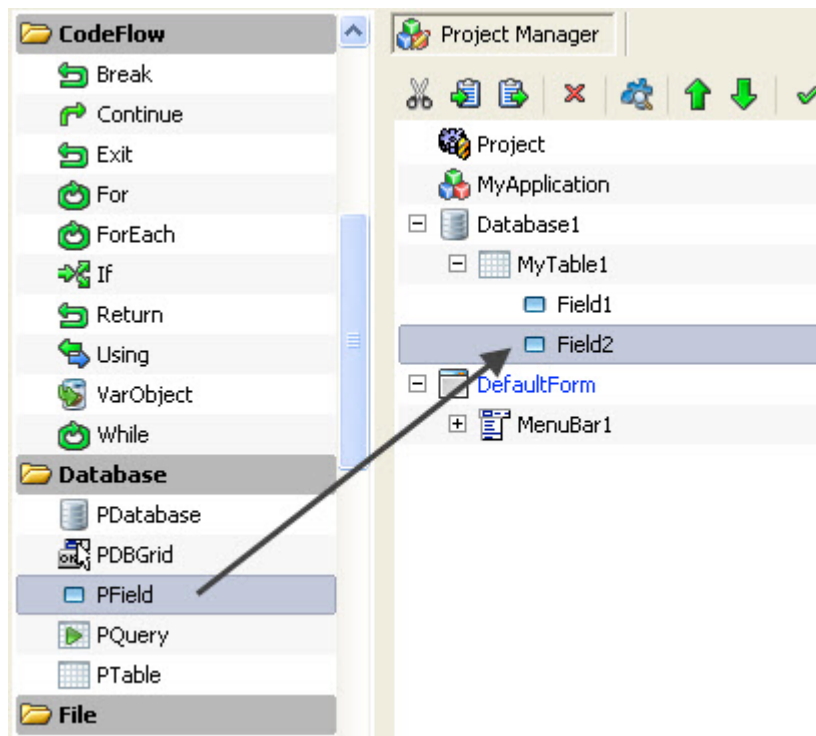


Figure 49: Add PField object

- Click all the **PField Objects** and change their **FieldName**, **FieldSize** and **FieldType** Property according to your needs.

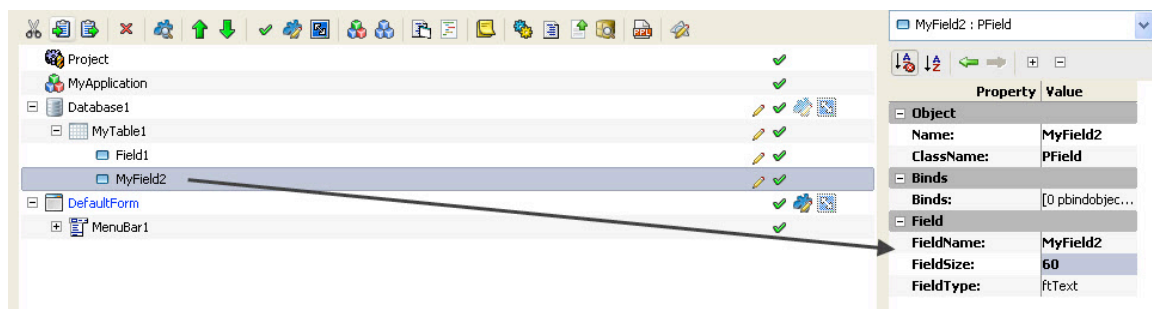


Figure 50: Specify Property

- In the data menu, Click on **Create Database** to create your database file in the location specified earlier in the **Filename** Property. Alternatively, you can also right click the **PDatabase** Object and select **Create Database** from **Data** to create a database. Doing this would create a physical database that is blank and contains all the tables and fields.

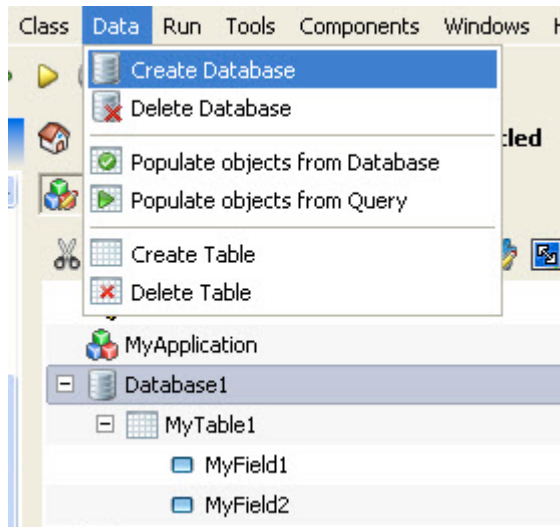


Figure 51: Creating Database

- Once you have created a database, you can double click the **PTable Object** to edit the field values.

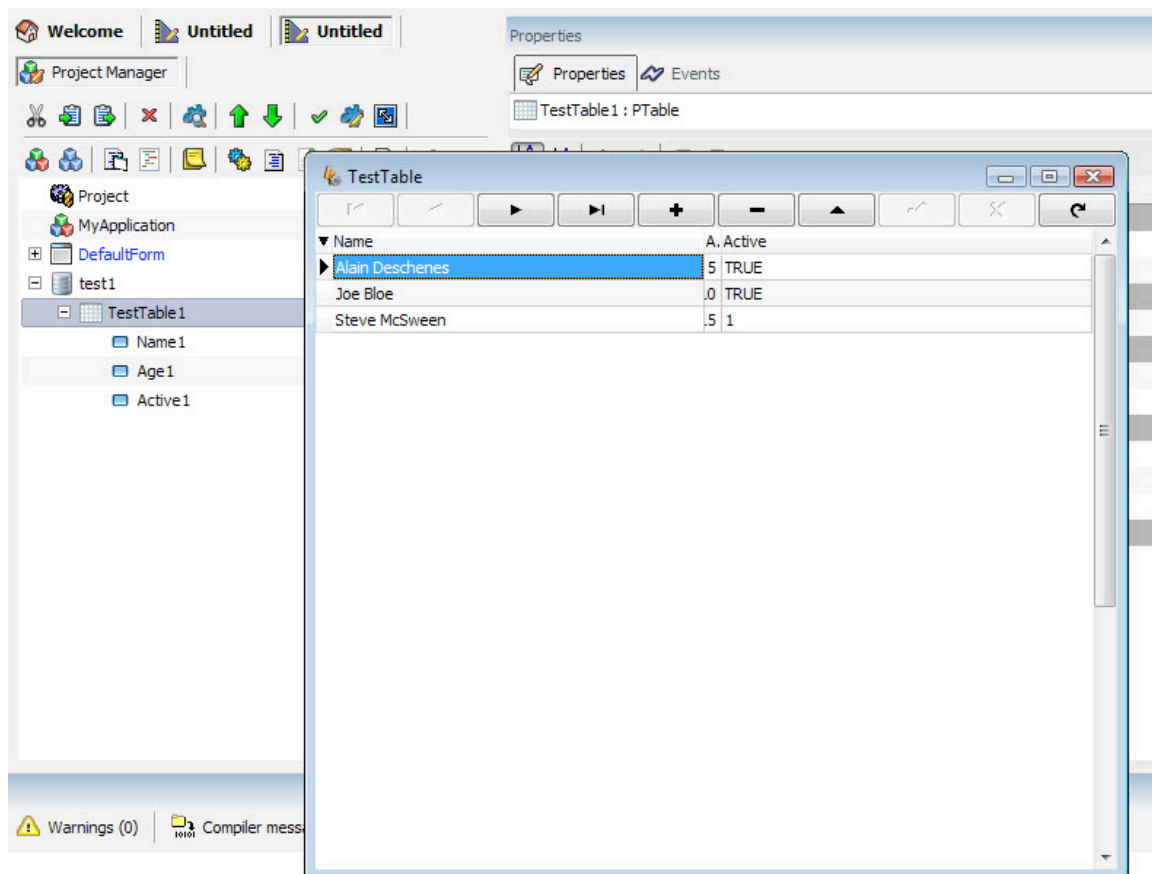


Figure 52: Viewing Table

- If you want to build your SQL query visually, you can also drag a **PQuery Object** from the **Components Panel** into the **Project Manager** or skip the next few steps to print everything on the form by default through **PDBGird**.

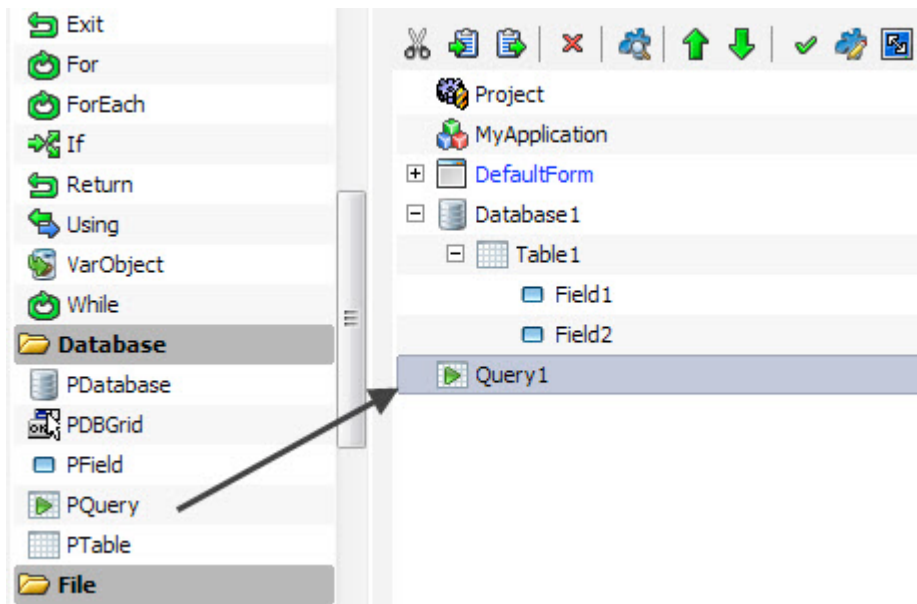


Figure 53: Drag PQuery

- After dragging and dropping the **PQuery** to the **Project Manager**, go to the **Properties Panel** and change the **Database Property** to a database that is already created.

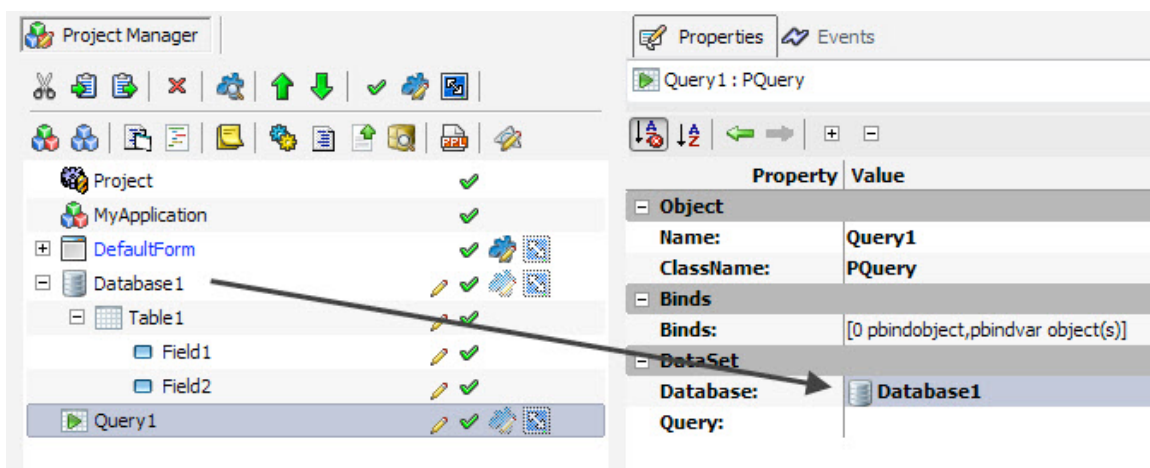


Figure 54: Change Database property

- Now double click the **PQuery Object** to create a **SQL query visually**. Refer to the next section(Using Visual Query)for in-depth knowledge on using the visual query editor for writing visual SQL queries.



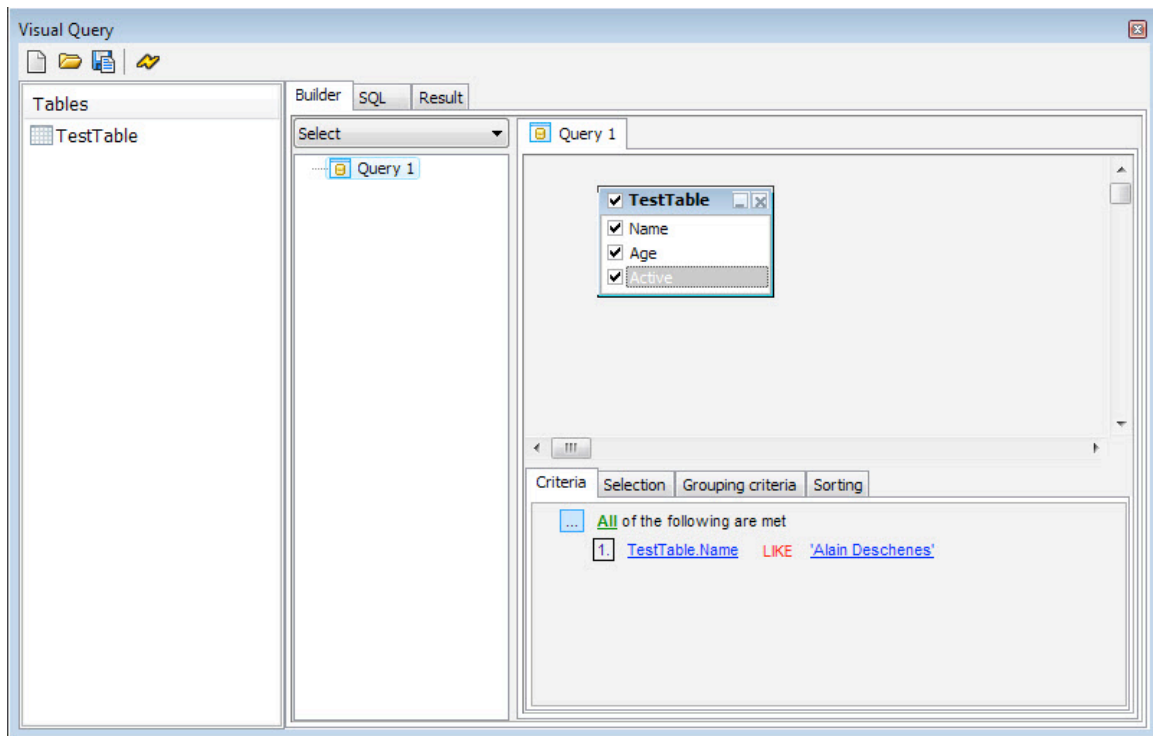


Figure 55: Visual query editor

- The database along with table and the query that we need to perform has been created. Now we can use other objects like **PDBGird** to display the table the way we want.

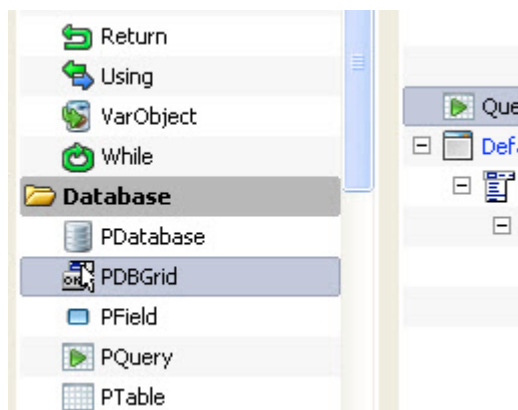


Figure 56: Insert PDBGird

- After creating a database, you can drag the **PTable** to a **PForm**. The drag operation will initialize the **SmartMove** feature which will ask you about the action that you would like to accomplish. Here, Select '**Add Grid to Control**' to create a table with a grid view in your form.



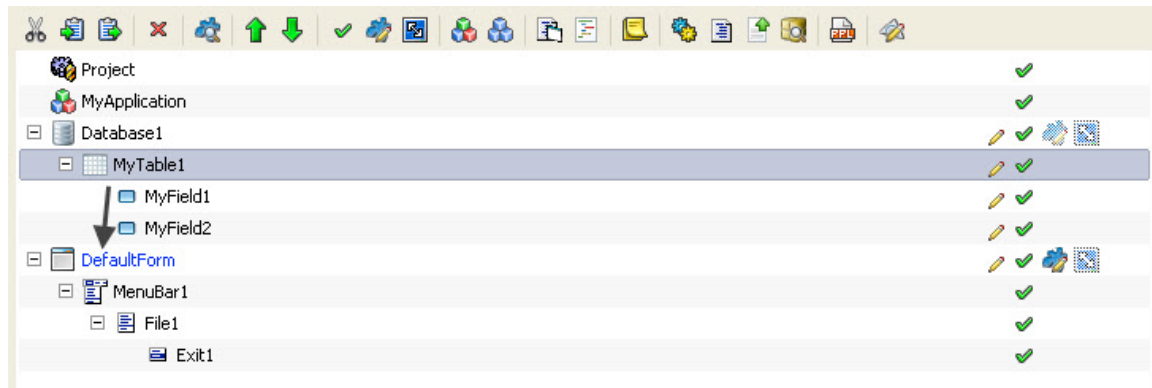


Figure 57: Drag table to form

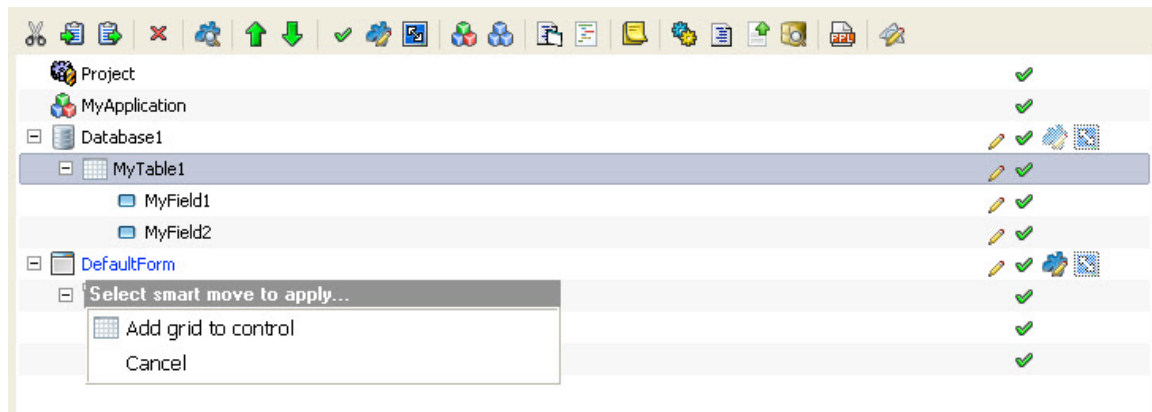


Figure 58: Smart move



*SmartMove feature in PIDE is very helpful in selecting the actions that would be performed. Likewise, code completion window is a unique feature that would allow you to better program while following visual programming.*

## Using Visual Query(PQuery)

With **PQuery**, users can easily change the way their database is handled. Users can easily create **Select**, **Insert** as well as **Delete** and **Update** SQL statements to manipulate their database the way they want. Users need to run these SQL statements in your form and **PQuery** is an object that will help them doing this.

In the example given below, we will use the database created in the above example and work upon it to create an SQLquery visually. Follow the given steps:

- Drag a **PQuery** object present in the **Components Panel** to the **Project Manager**.

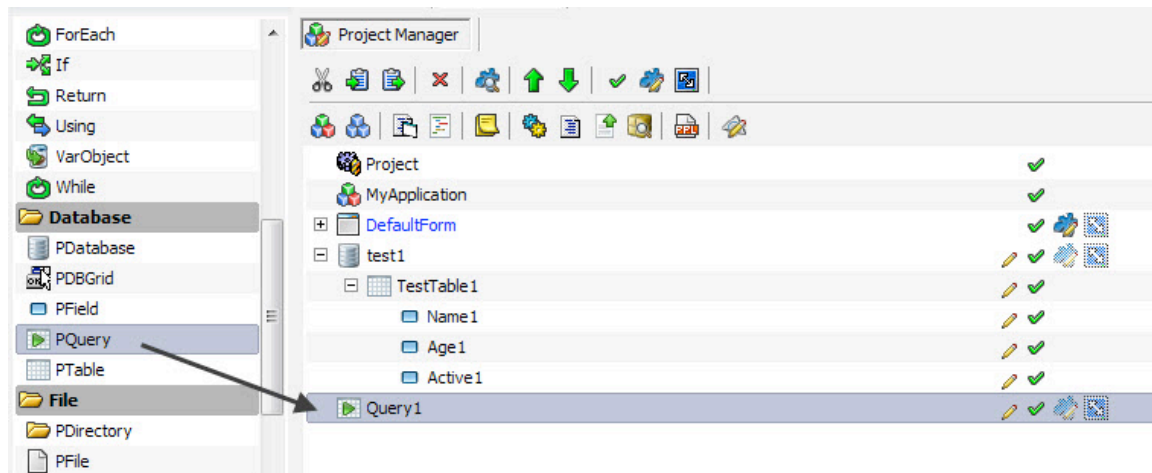


Figure 59: Drag PQuery

- Select your database name in the **Database Property** of **PQuery**. To do this, select the **PQuery**, and select the database name i.e test1, in the Database property.

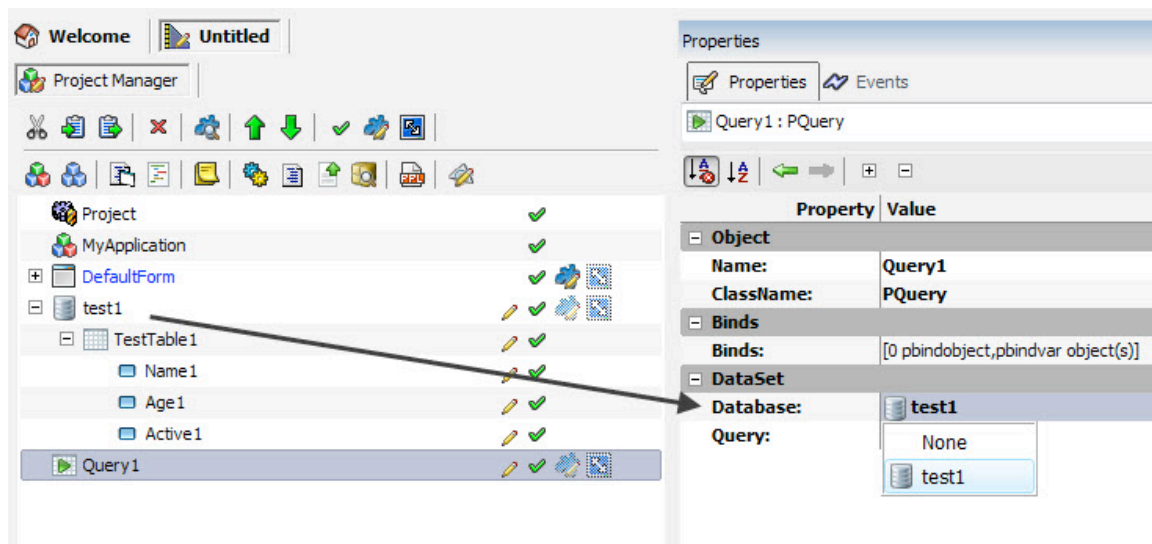


Figure 60: Select appropriate database name

- Double click the **PQuery Object** to open visual query editor.

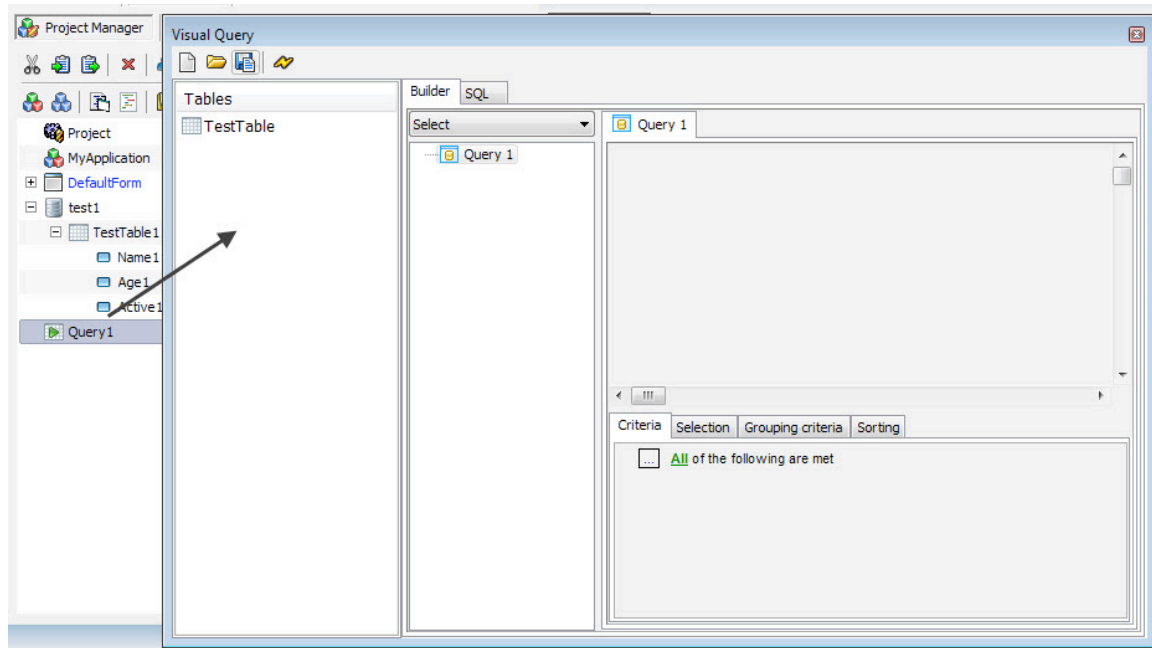


Figure 61: Double click PQuery

- At the top, **visual query editor** has four buttons that are used to perform tasks like **new, open, save, and run**.

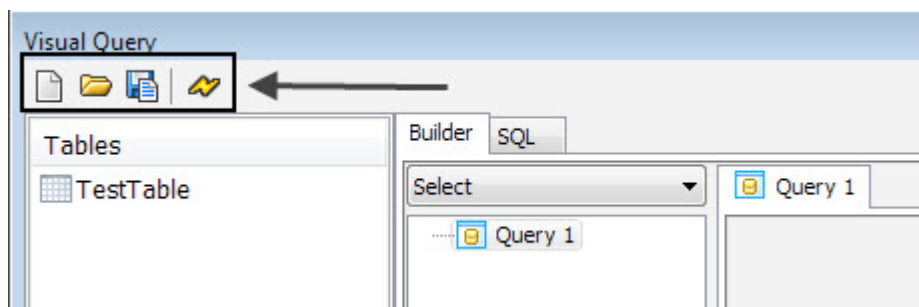
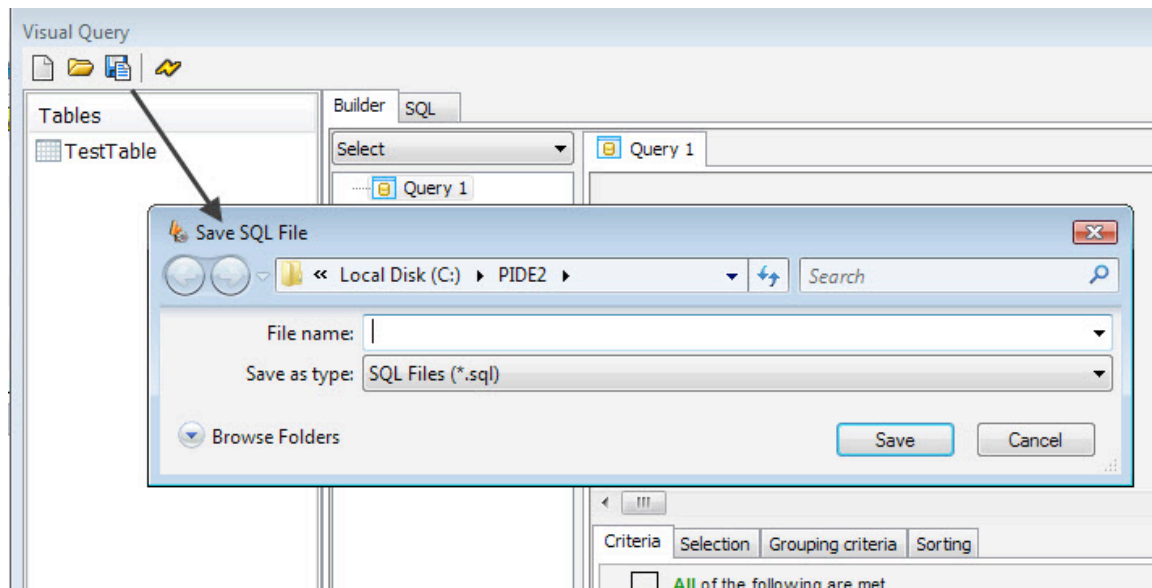


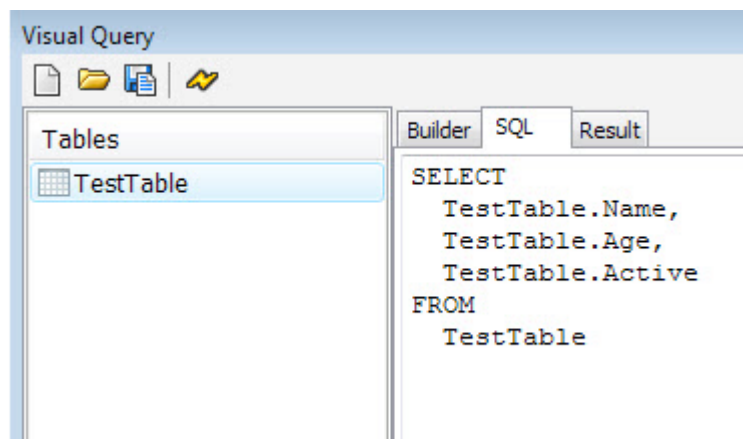
Figure 62: The four buttons of visual query editor

- While the **New Button** creates a new query, the **Save Button** is used to save the existing query in a .SQL format. All the SQL queries saved in .SQL format can be opened in the visual query editor through the **Open Button**. Lastly, the **Run Button** is used to execute an SQL query.



**Figure 63: Save file button**

- The main view of **visual query editor** consists of a **Tables** pane and three tabs. The tables pane contains the names of all the tables present in the database that was associated with a **PQuery Object**.
- The first tab is the **Builder Tab** that allows users to build SQL query visually.
- The second tab is the **SQL Tab** that allows a user to write SQL query or modify a SQL query created with the help of builder tab.



**Figure 64: The SQL Tab**

- The third tab is the **Results Tab** that shows the result of a query. After a query is created, it should be executed by pressing the run button. The result of the query is displayed in the **Result Tab**.

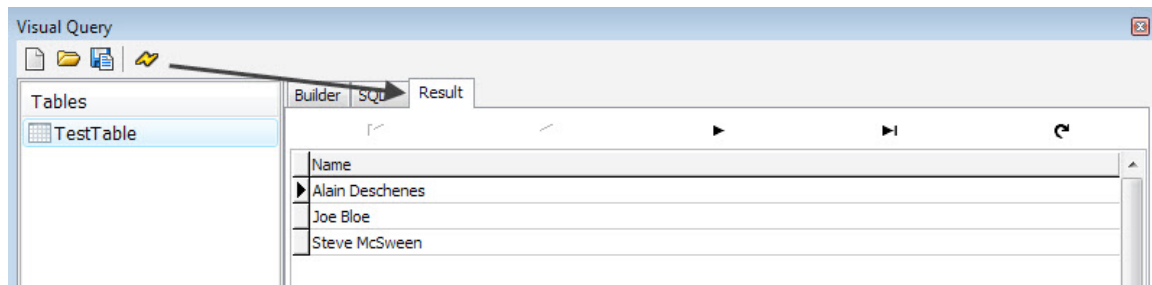


Figure 65: Results Tab

- Users of PIDE2 can use the builder tab present in visual query editor to build **Select, Insert, Update and Delete** SQL queries.

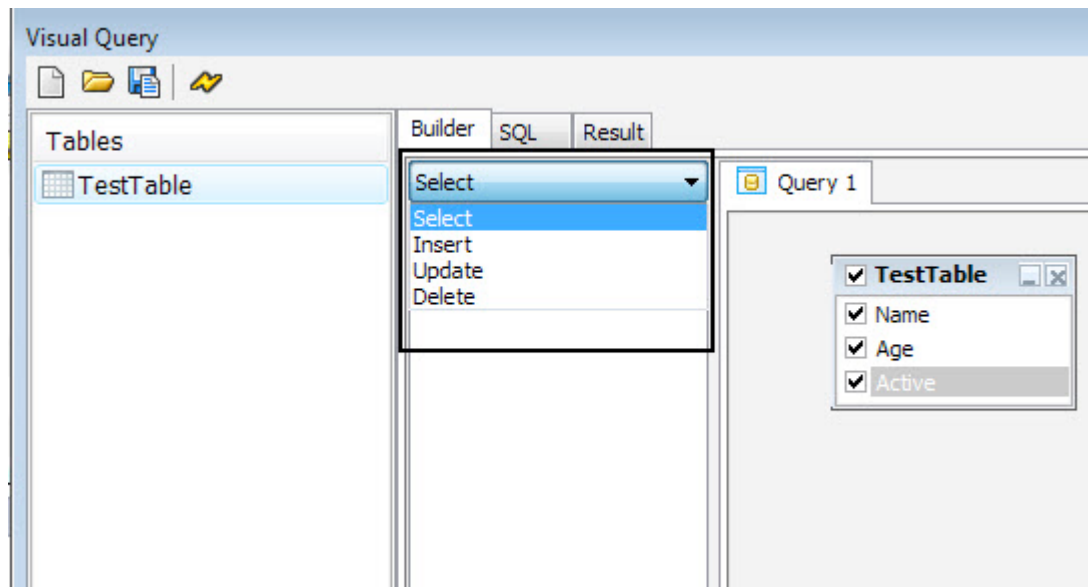


Figure 66: Selecting Query Type

- The **Criteria, Selection, Grouping criteria and Sorting Tabs** can be used to build advanced SQL queries.

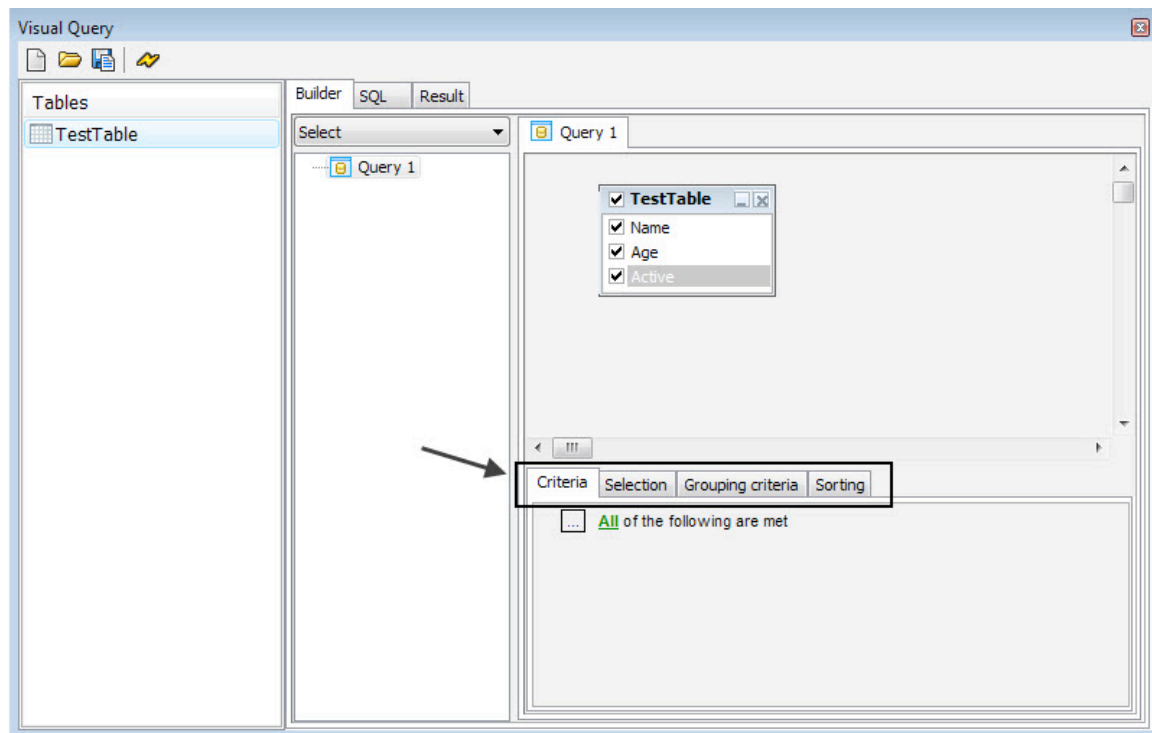


Figure 67: Creating Advanced Query

## Using A Database Project

A database project is something that contains just the basic ingredients for creating a database driven application. There are basically two ways of using a Database project.

**Method 1:** For creating customized applications and starting just with a database, its table and fields.

- In PIDE, Press **Ctrl+N** or Go to **File>New** and Click on **Database** to create a project with just a database, its table and two fields.

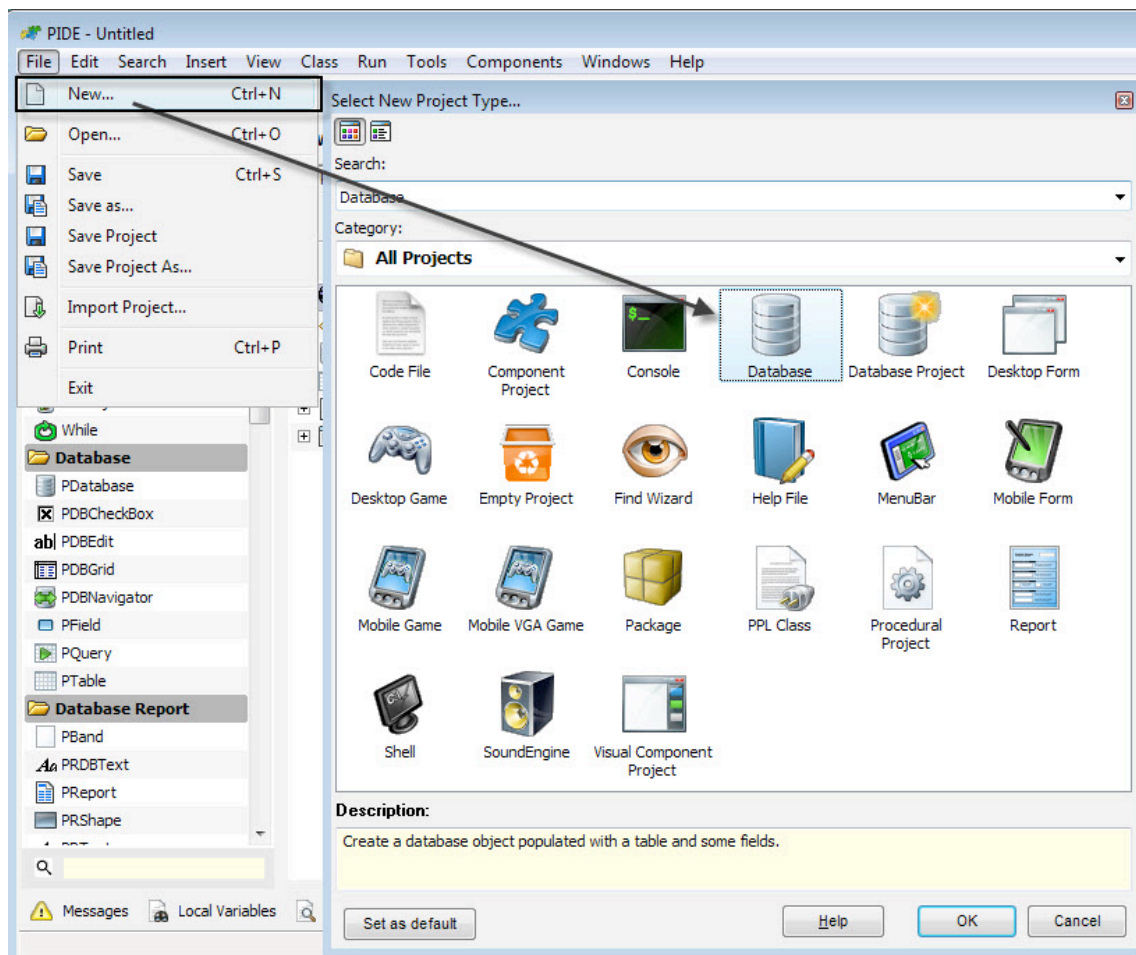


Figure 68: New Project

- Now that you have a database for your database driven application, you can easily add other objects like form or create a game if you like.



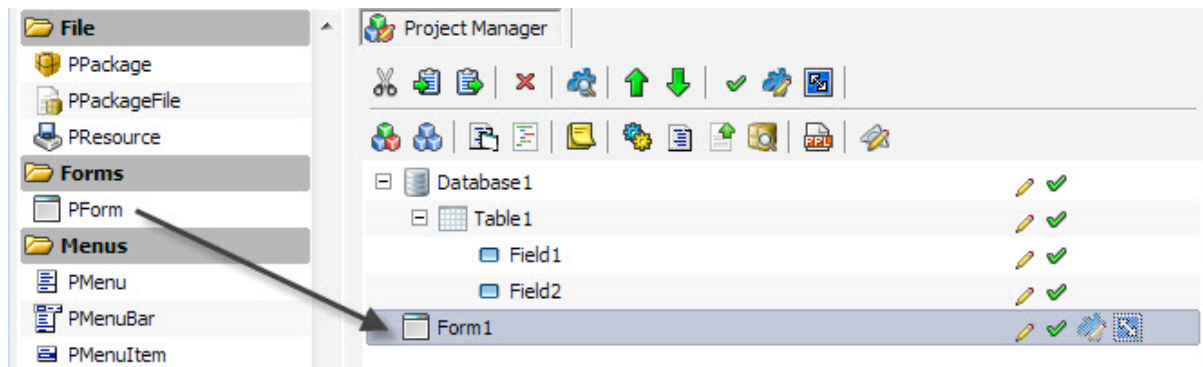


Figure 69: Drag a PForm

**Method2:** Adding a database and all the related components to an existing project.

- Many times, we start with a project and then want to add database functionality to it. Instead of adding individual components like a database, its table and related fields, PIDE provides you to add all that in a single go with the help of **Database** project. While developing a **Desktop Form** project/**Desktop Game** project or any other such project, press **Ctrl+N** or go to **File>New** and select **Database** project to automatically create a Database and all its related elements in a single go!

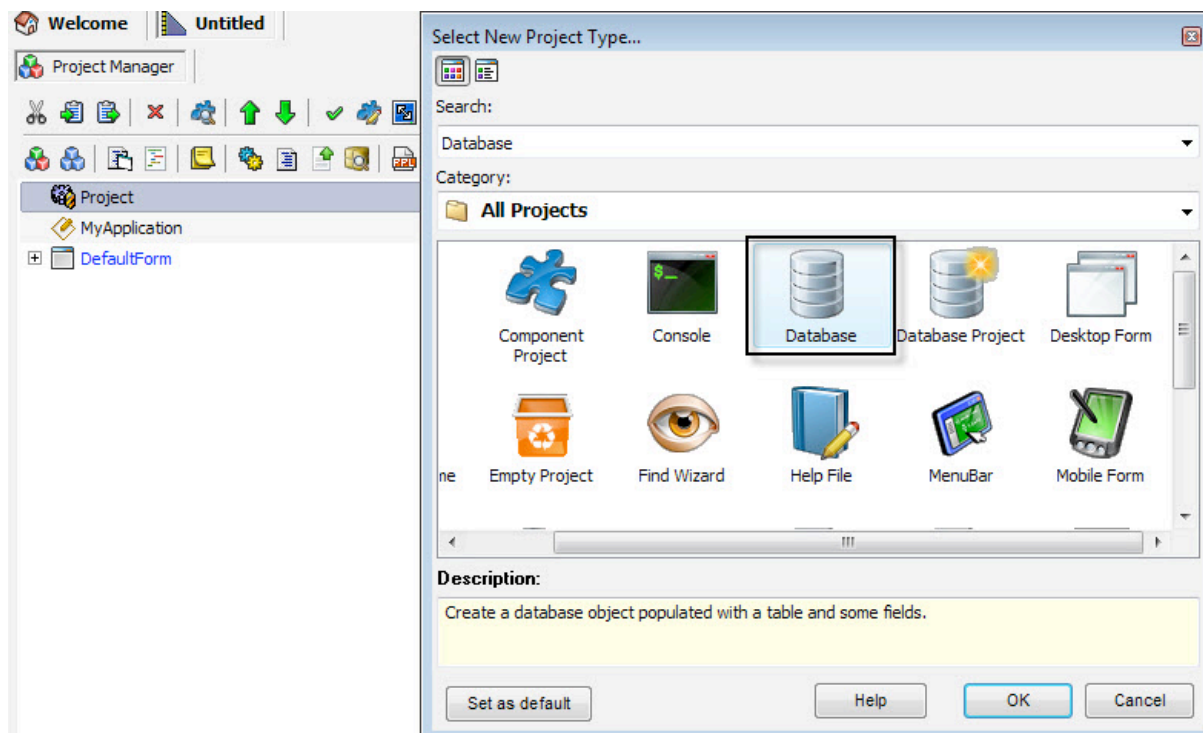


Figure 70: Create new project

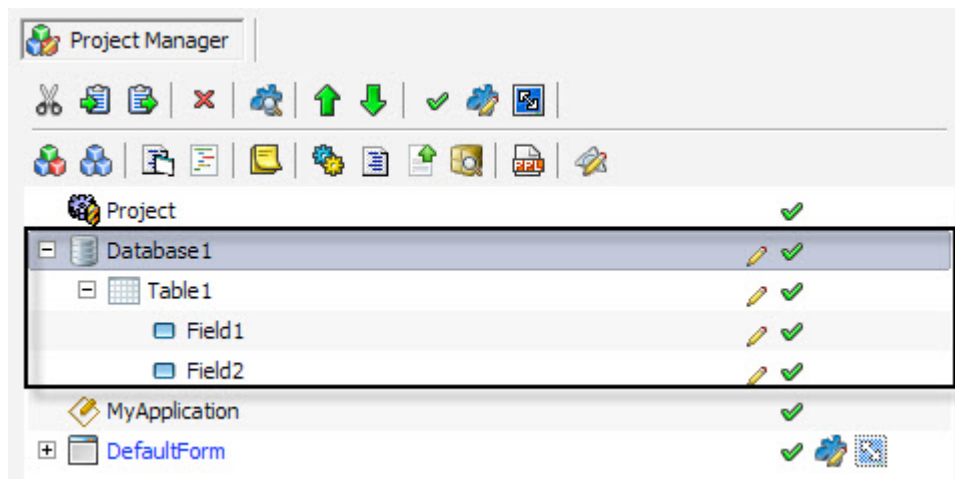


Figure 71: Get database

## Using Database Project

A Database Project is the premier choice if you want to create a database driven Windows or Windows Mobile based application. Just by a single click of a button, users can give access to all that they need to create a database driven application. A database project provides its users with Database, **PQuery** object, **PReport** Object, as well as all the basic necessities of a database like a table and fields.

- For creating a database project, Press **Ctrl+N** and simply select **Database Project** from the “Select New Project Type” window.

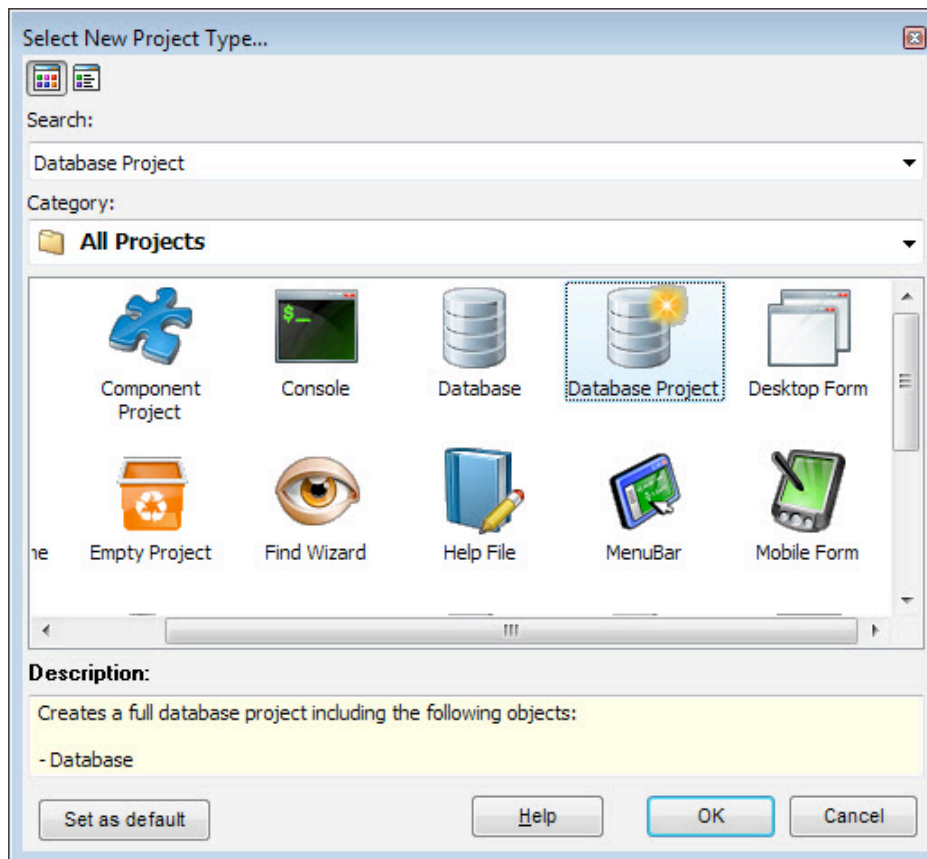


Figure 72: Create new project

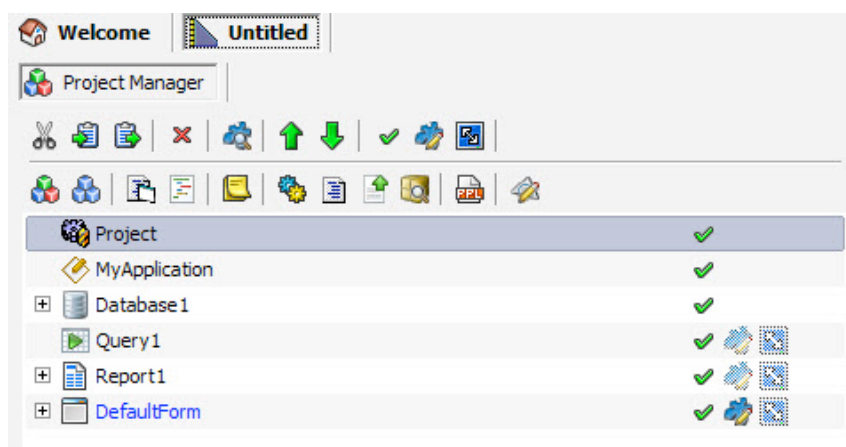


Figure 73: Database Project

## Using The PGird And PDBGird

**PGird** and **PDBGird** are objects in PIDE that are used to impart a grid type layout to a certain text. While they both are used to perform more or less the same thing, their application differs as **PGird** is more focused on the layout of a text whereas the layout of a **PDBGird** is related to the layout of table elements present in a database.

**PDBGird** is a very important object that is used for visually aligning table elements in an easily understandable format. Given below are the steps that are required to create a **PDBGird** as well as **PGird** objects in a PIDE application.

### *Creating a PGird Object*

Creating a **PGird** Object is a simple to use procedure that both easy and effective one. With the help of a **PGird**, we can add grid functionality to the application and use it for efficient layout of data.

Given below are the steps to create a **PGird** Object in a database form:

- Start by creating a new **Desktop Form**
- Now drag and drop a **PGird** object on the **Default Form**
- For filling out the values, you can use the **Text** property of **PGird**.
- That's it! Press **F5** to run your application.

While we have created a **PGird** in the above steps, this **PGird** will do nothing because it has not been programmed to do so. In the example below, we will create a **PGird** that mimics the actions of a spreadsheet that can calculate the average of various columns with numbers.

To explain **PGird**, we will create an application that would allow a user to enter details about some students and certain values based on the column headings and get a sum total.

Follow the steps given below:

- Open PIDE2 and create a new **Desktop Form** project from the **Select New Project Type...** window. Using **Ctrl+N** is the keyboard shortcut for displaying **Select New Project Type...** window; alternatively you can also use **File Menu>New**.

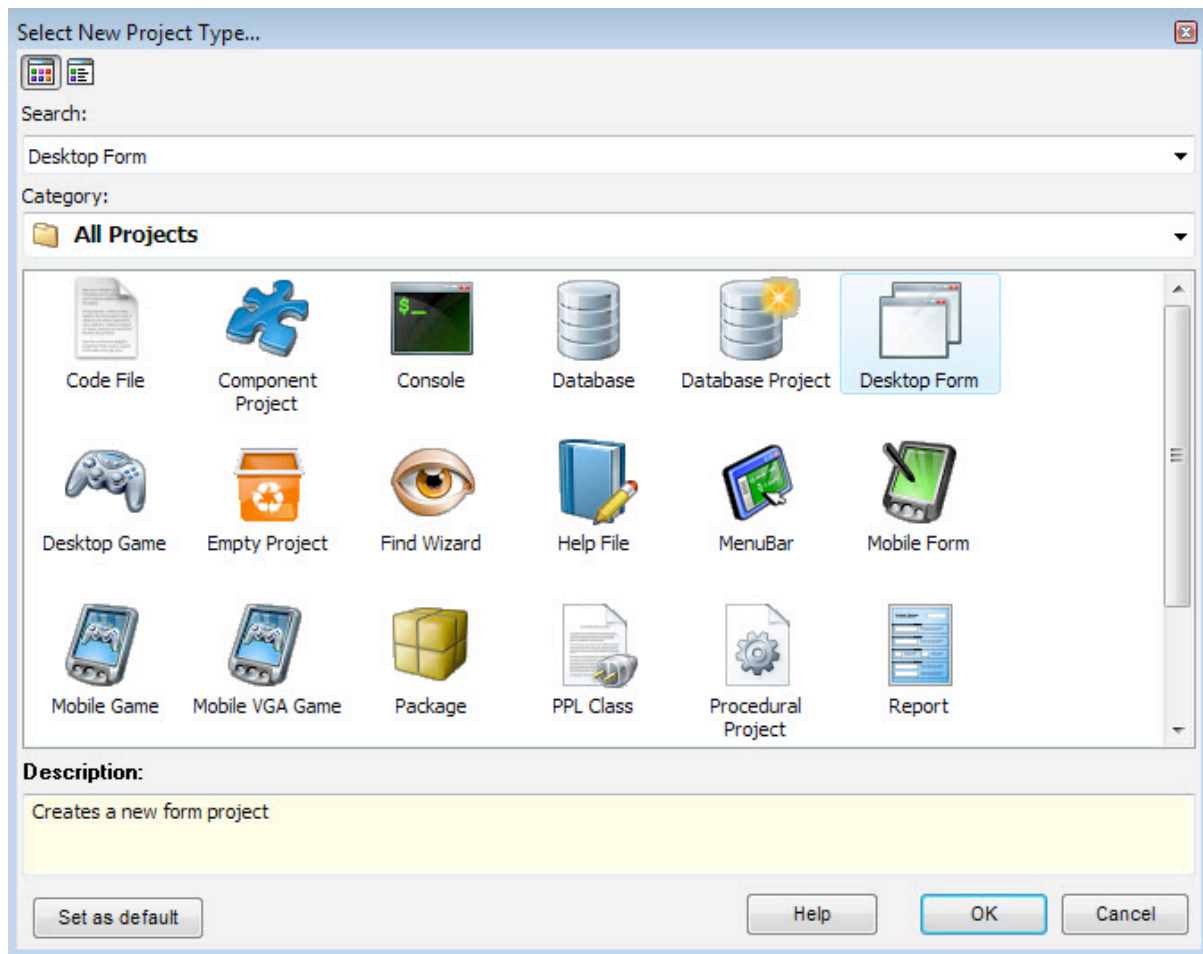


Figure 74: Create new project

- In the **Project Manager**, double click the **DefaultForm** object to open the **Form Editor**.

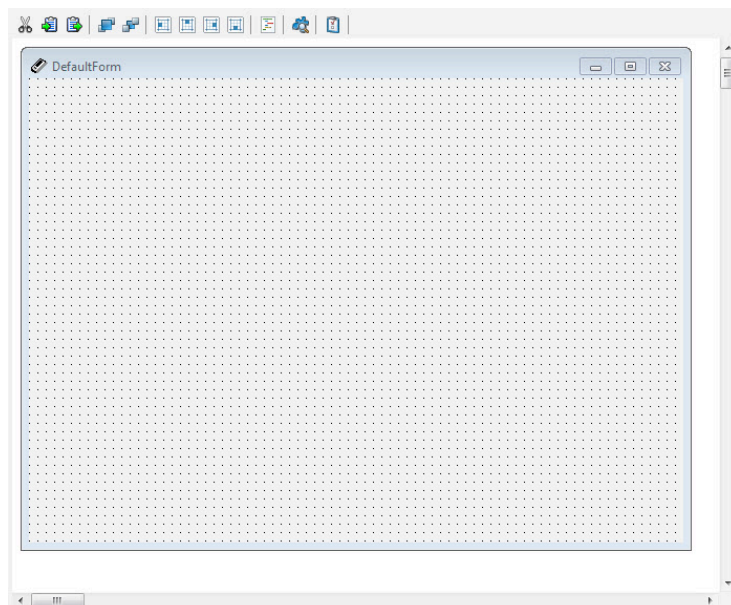


Figure 75: Form Editor

- We need to place a **PGird** on the Form. For doing this, click on the **PGird** object in the **Components Pane** and then click at the place where you want to place the **PGird** object. You can set the **PGird** according to your preference by dragging the borders to the desired size.

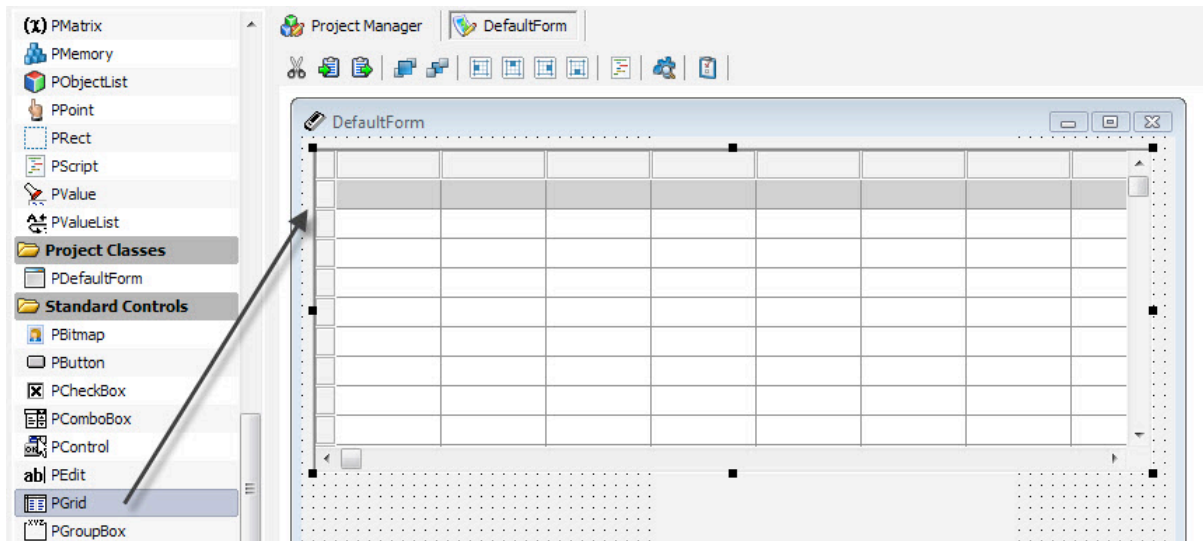


Figure 76: Drag PGird

- Now, use the **Text** property of **PGird** to place appropriate heading for columns.

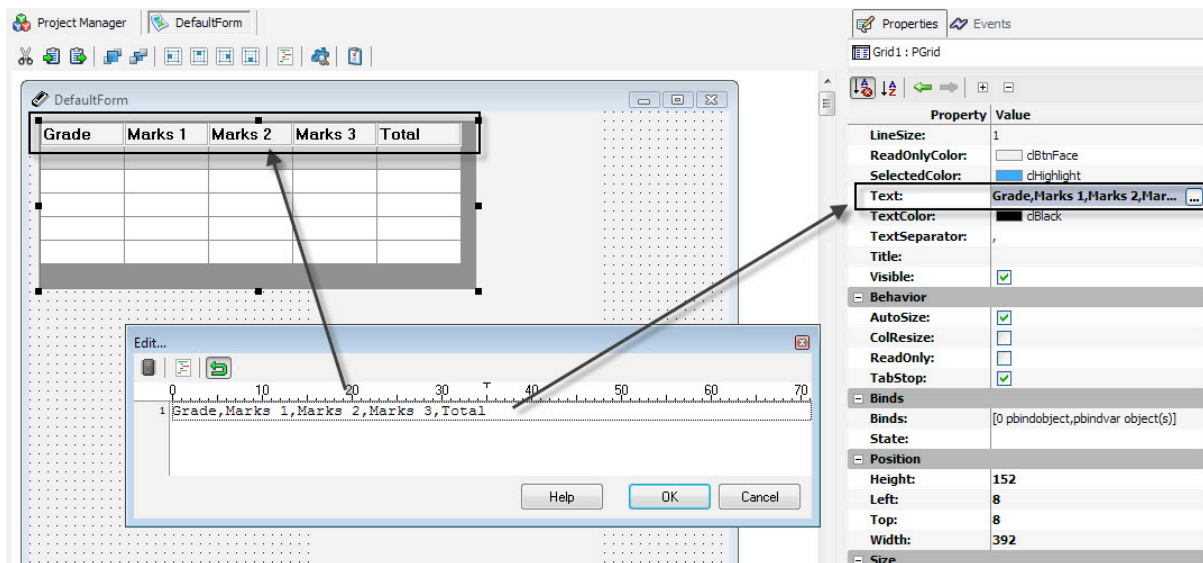


Figure 77: Text Property

- After filling out the columns in the text property of **PGird**, go back to the **Project Manager**.



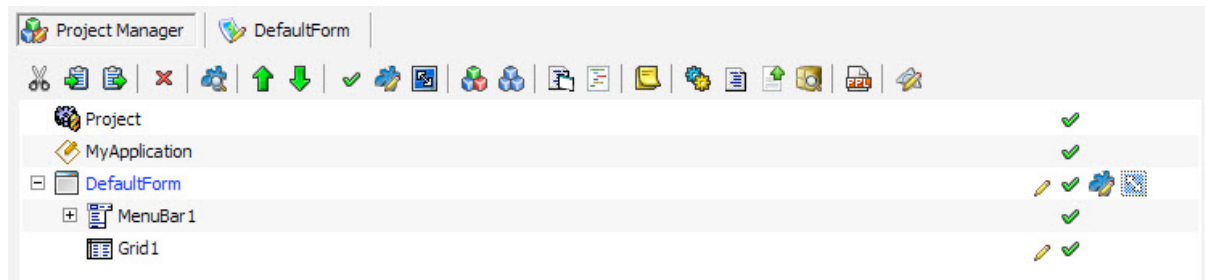


Figure 78: Project Manager

Note: You can also use the **ColCount**, **RowCount** and the **ExtendedLastColumn** property to have more control over the layout of your **PGird**.

- In the **Project Manager**, click the **PGird** object and go to the associated events by clicking the **Events** pane.

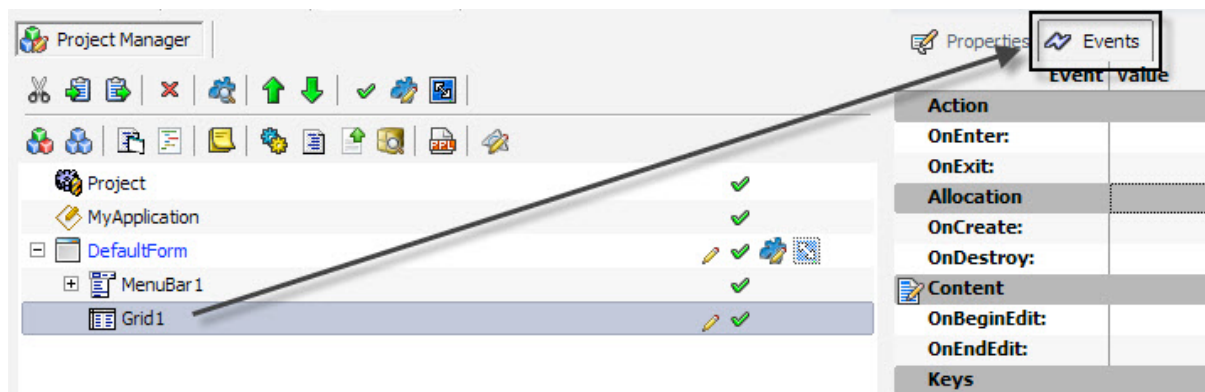


Figure 79: Events Pane

- In the Events pane, double click **OnEndEdit** event available under the Content column to create an **OnEndEdit** event.

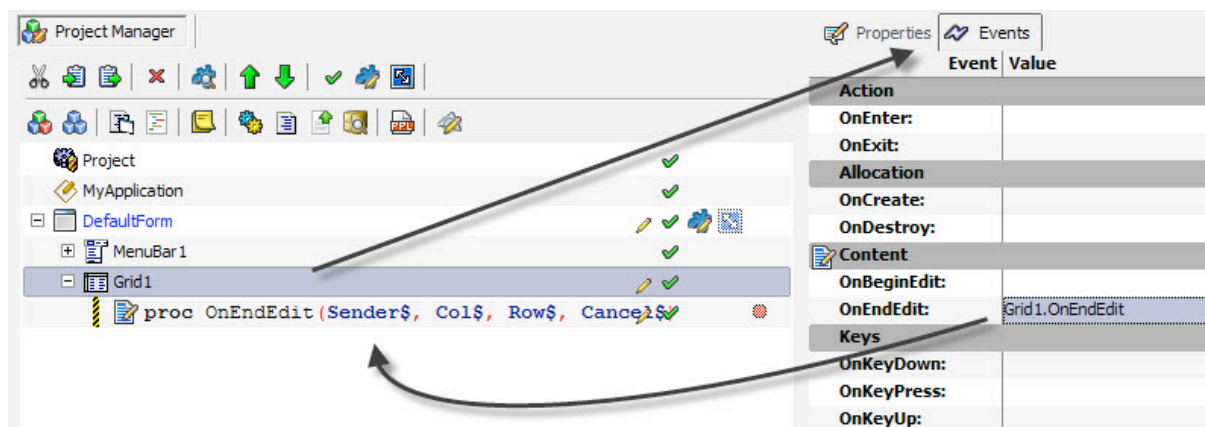


Figure 80: Create event

- Drag a **PVariable** to the **OnEndEdit** event and name the local variable appropriately.



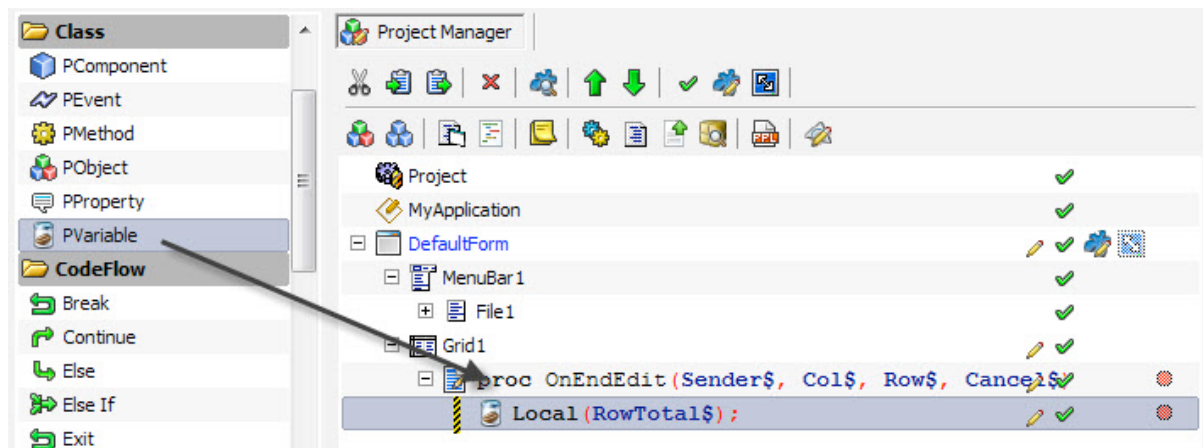


Figure 81: Drag PVariable

- Now, drag a **For** loop on the **OnEndEdit** event and drag the **PGird** object to the Finish property of **For** loop. This would result in a code complete window.

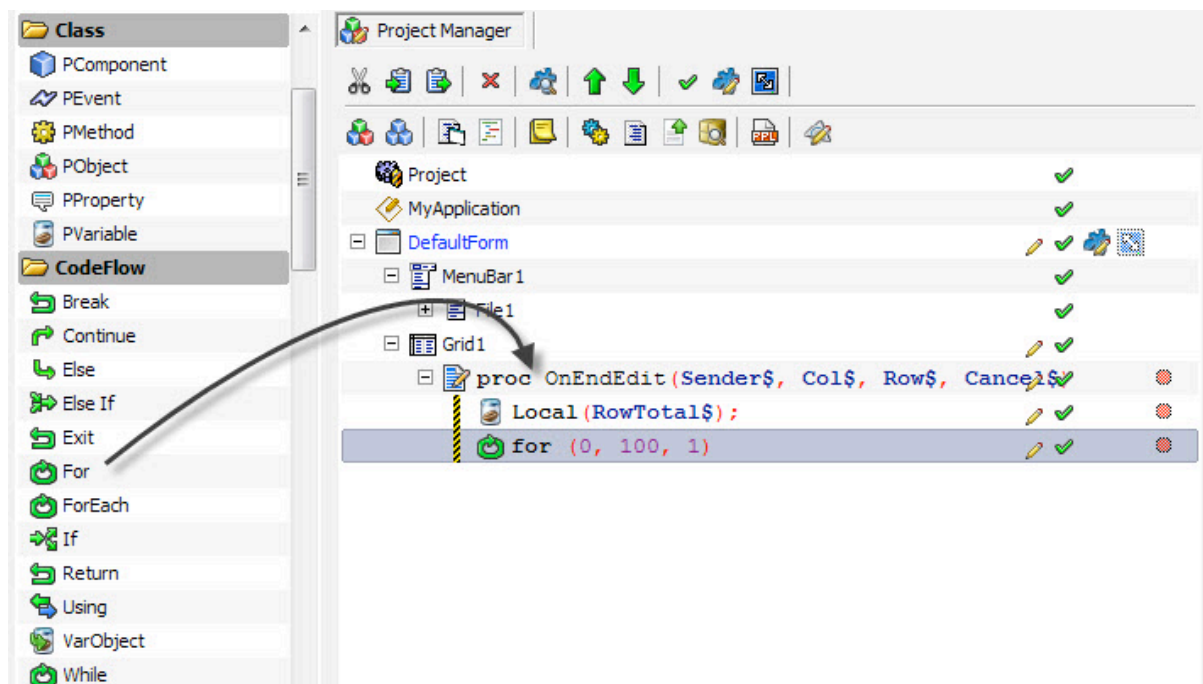


Figure 82: Drag For Object

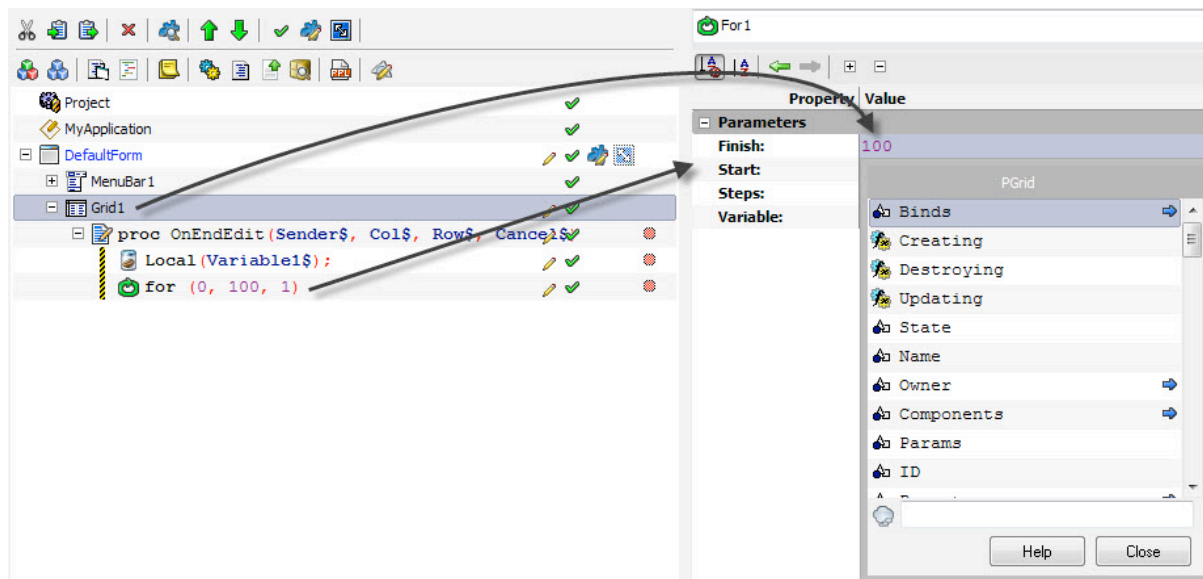
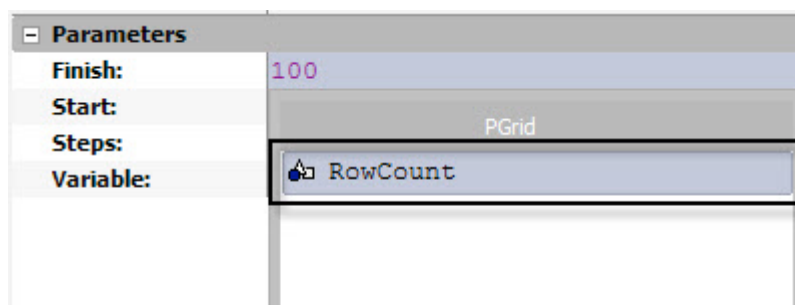


Figure 83: Set For properties

- Select **RowCount** in the code complete window.



- Select the **For** loop again and specify a variable in it. In our example we will use **y\$**.

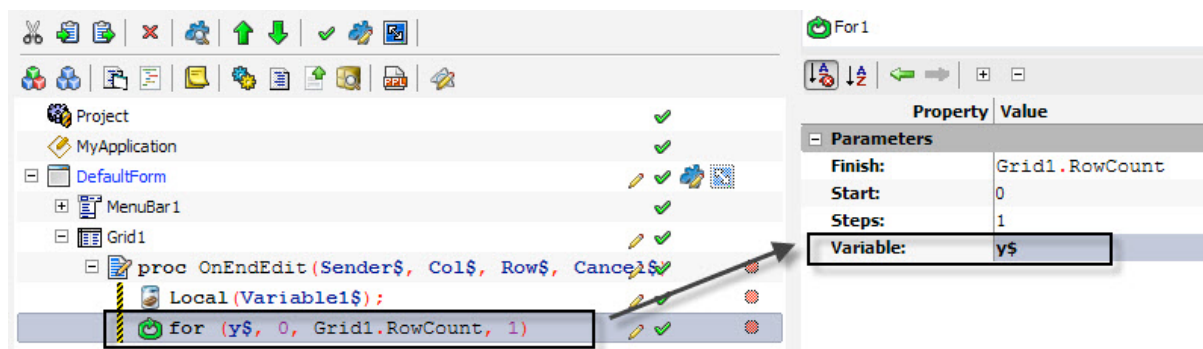


Figure 84: Change Property

- **Alt + Drag** the local variable to the **For** loop so that it comes below it. If the local variable is not below the **For** loop, drag it there.

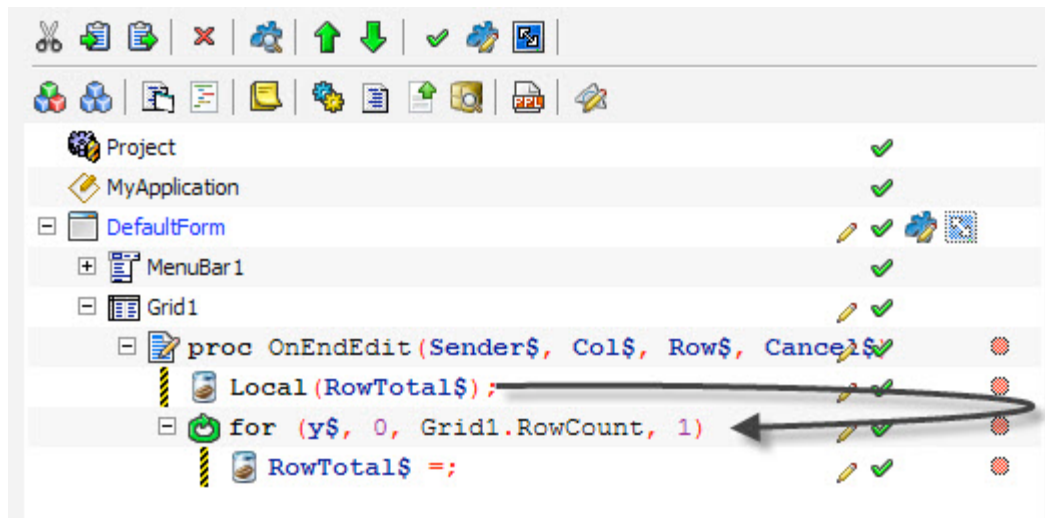


Figure 85: Alt + Drop

- Change the **Expr** property of the dragged variable to **0**. This would set the counter at **0**.

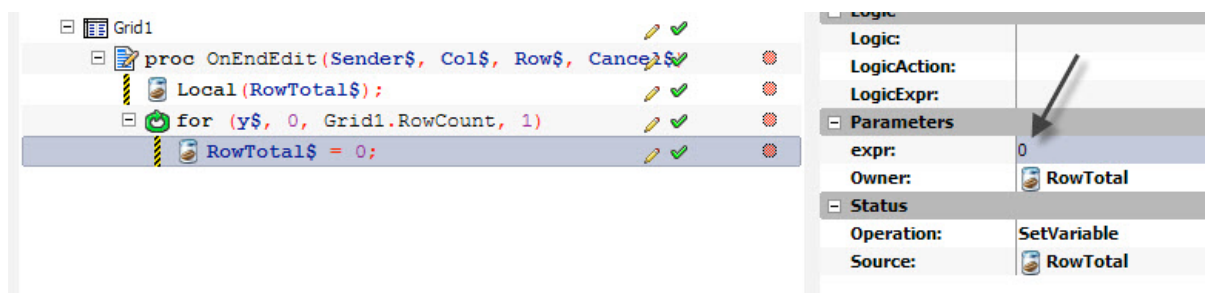


Figure 86: Change Expr

- Drag another **For** loop from the **Components Panel** on the **For** loop available on the **Project Manager**.

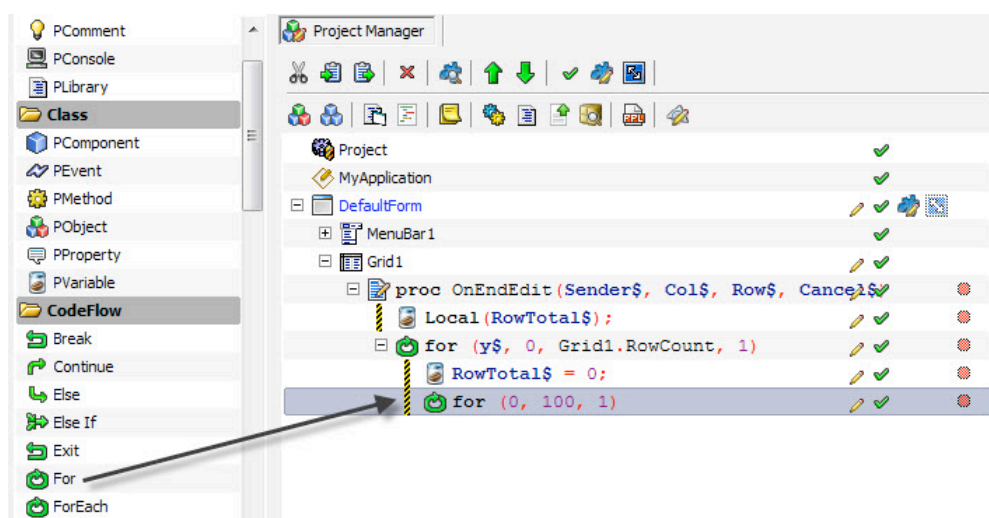


Figure 87: Drop a For loop

- In the Finish property of the **For** loop, write (**Grid1.ColCount -1**). This should be enclosed within brackets so as to specify that it is not an Expression. Alternatively, you can also drag **PGird** object, select **ColCount** option and add **-1** to it later.

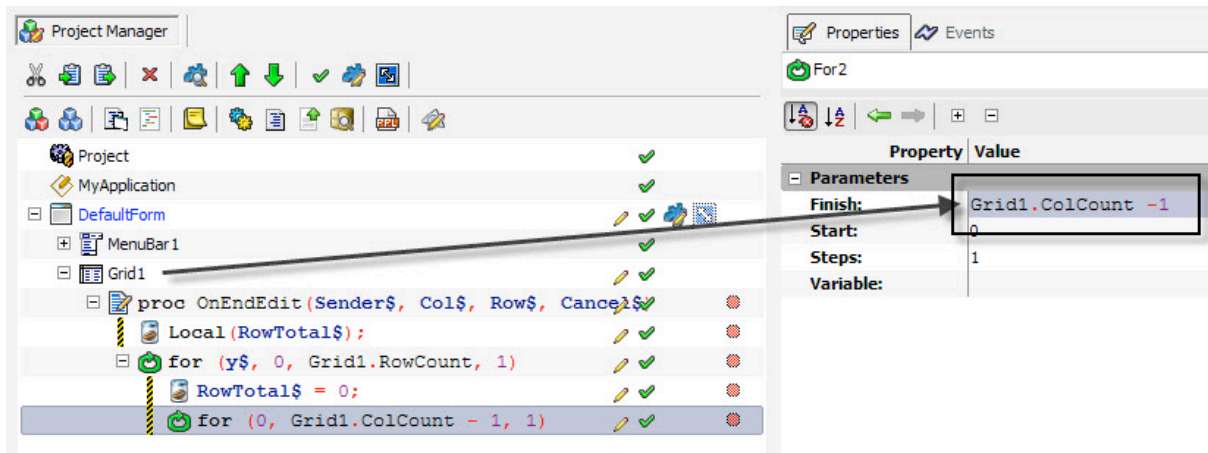


Figure 88: Change Property

- Tweak the Start property of this **For** loop to **2** and set the variable property to **x\$**.

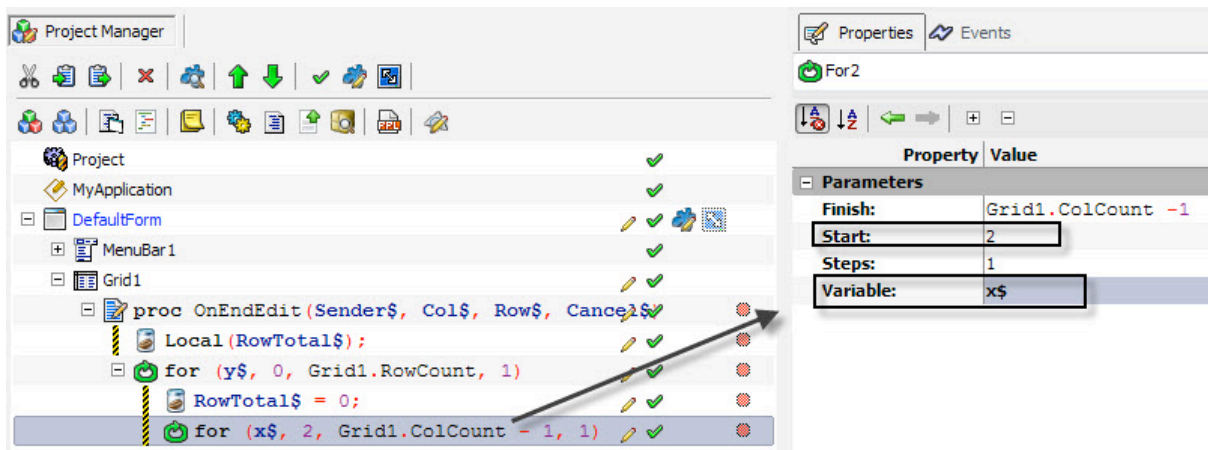


Figure 89: Change Property

- Drag the **PGird** object to the **For** loop. This would result in a **Code Completion** window.

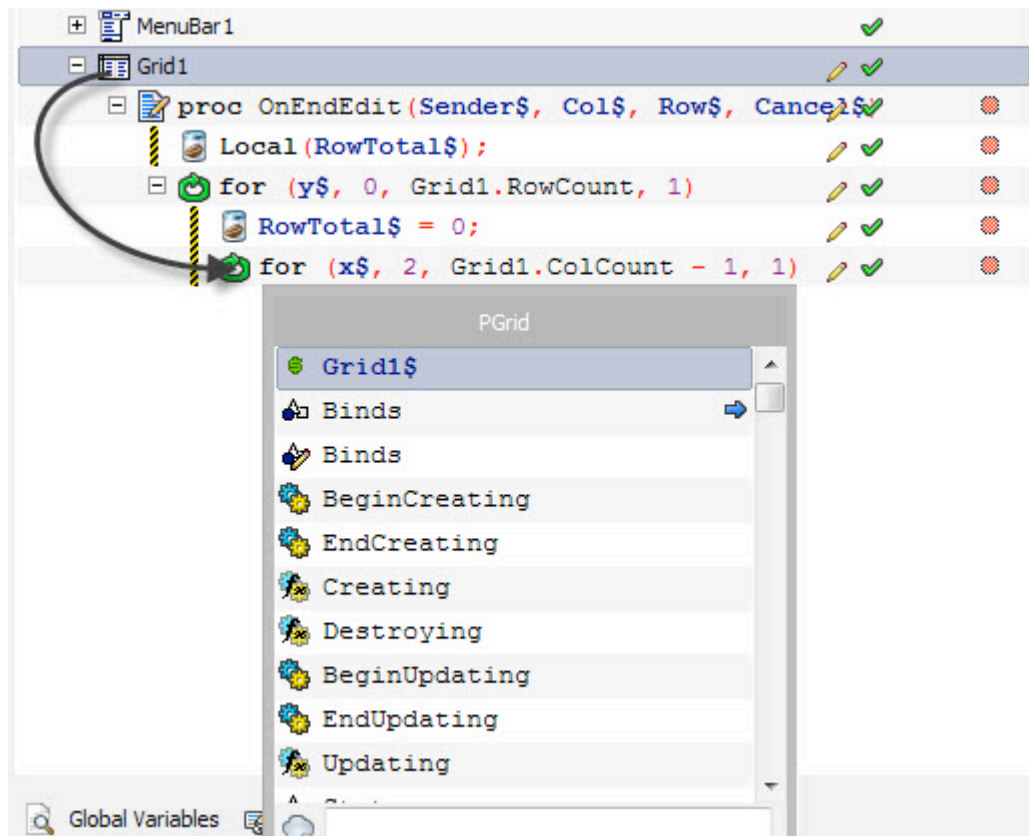


Figure 90: Code Completion window

- Select **Get(col\$,row\$)** from the **Code Completion** window.

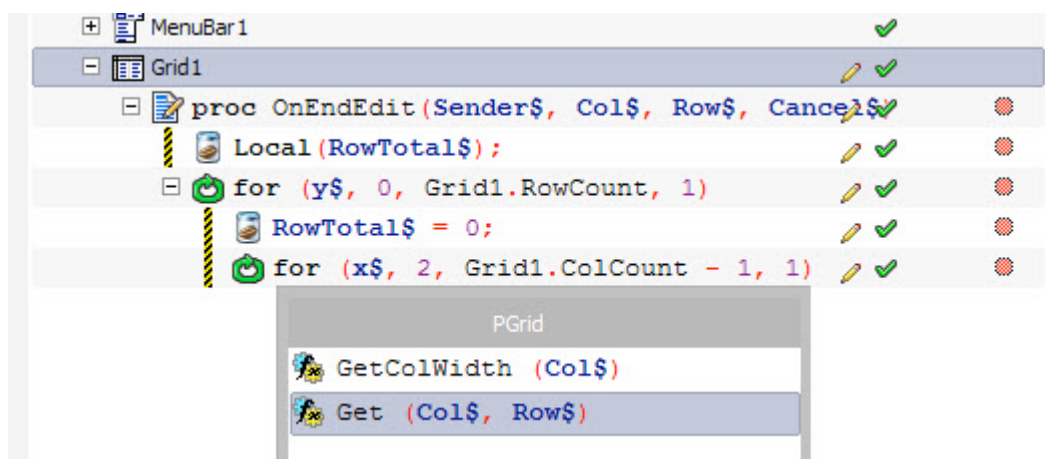


Figure 91: Select option

- Select the new **Grid1.Get(col\$,row\$)** and set its **col** as well as **row** property to **x\$** and **y\$** respectively.



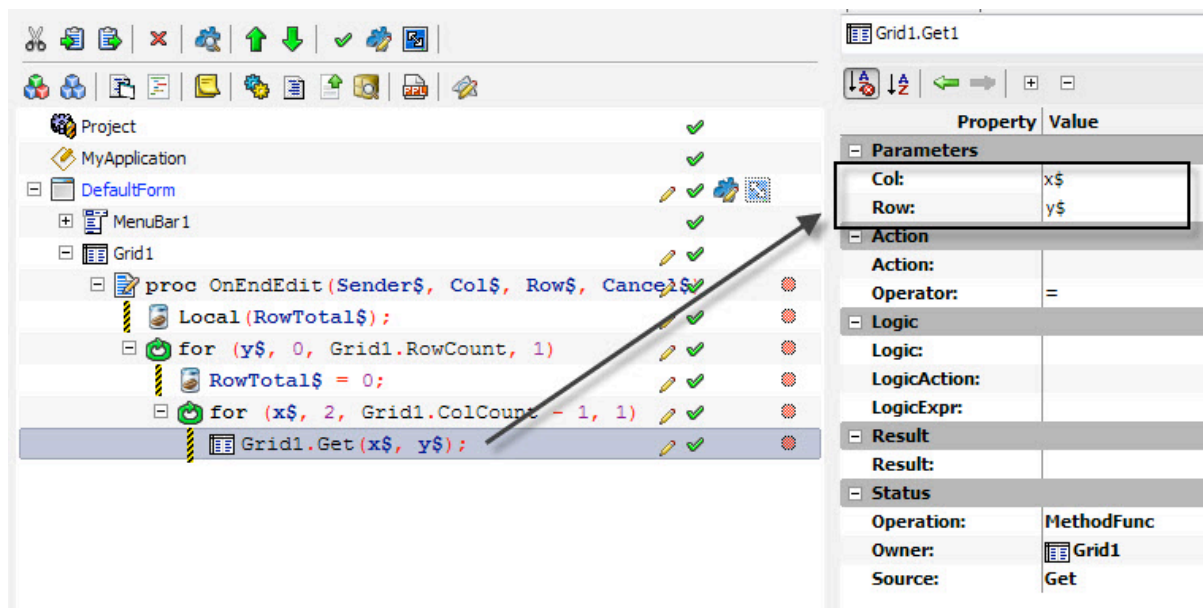


Figure 92: Change property

- Now, change the **Operator** property of **Grid.Get(x\$,y\$)** to **+=** and drag the **RowTotal\$** variable to the result property. This would increment and store the result to this variable.

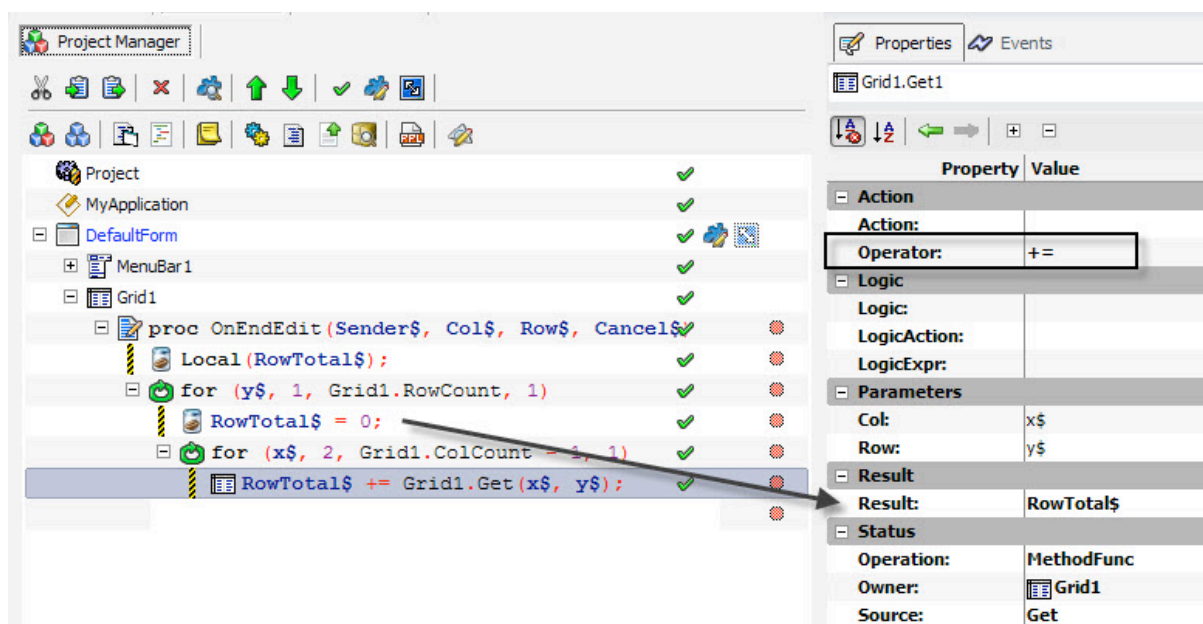


Figure 93: Drag RowTotal\$

- Drag the **Grid1** object to the parent For loop and select the **Set(col\$, row\$,value\$)** option. This would result in a **Grid1. Set(col\$, row\$,value\$)** code.

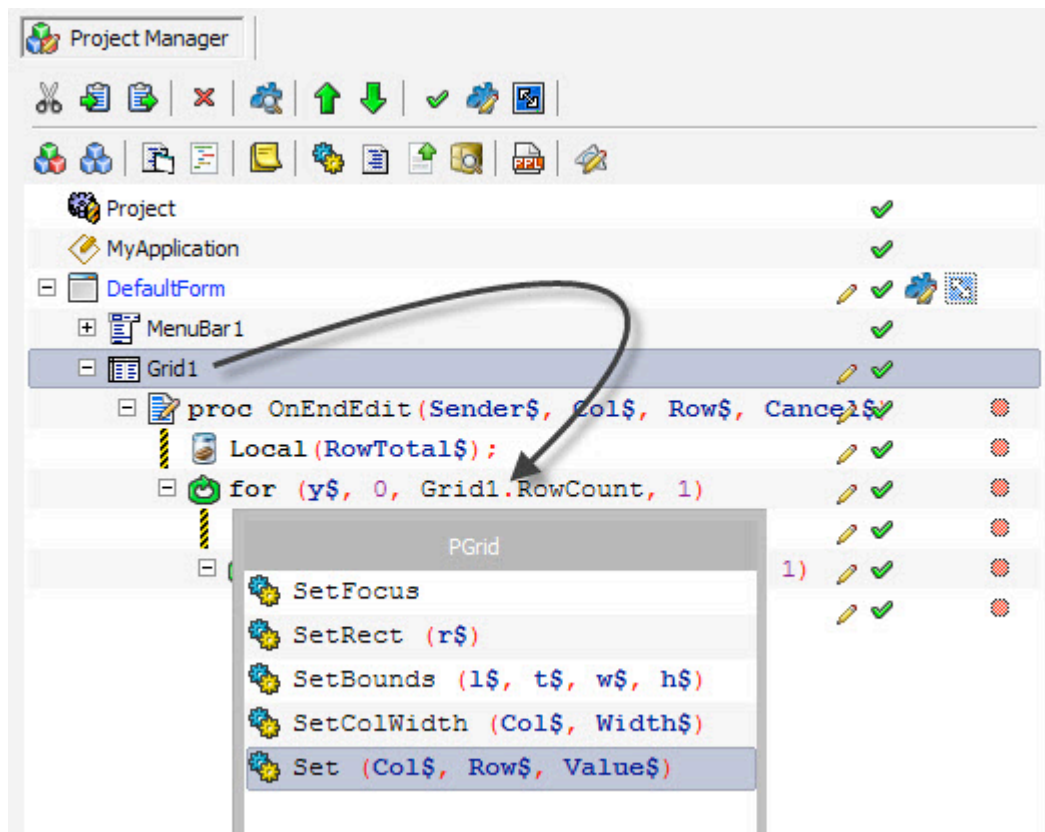


Figure 94: Drag and Drop

- Select **Grid1.Set(col\$,row\$,value\$)** and drag the **PGird** object to the **Col** property and select **ColCount** from the **Code Completion** box that appears.

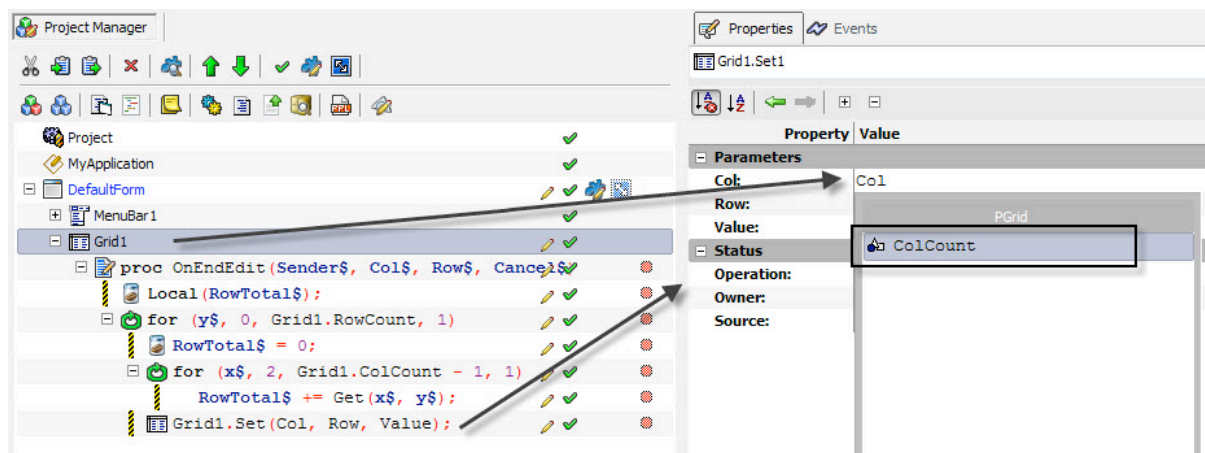


Figure 95: Drag operation

- Change the **Row** property to **y\$**



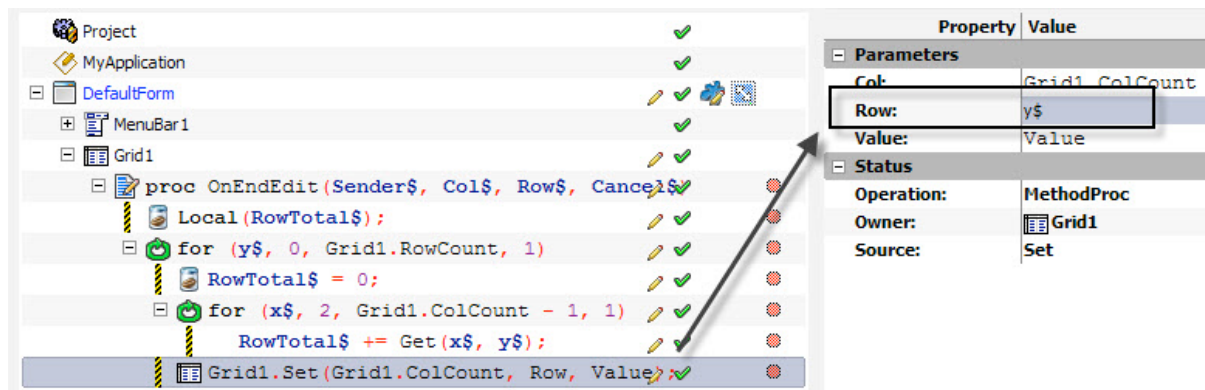


Figure 96: Change Row property

- Change the **value** property of **Grid1.Set(Grid.ColCount,y\$,value\$)** to **RowTotal\$ / (Grid1.ColCount - 2) + "%"** . This will divide the value in the **RowTotal** variable with that of the total number of columns with numerical values in our grid.

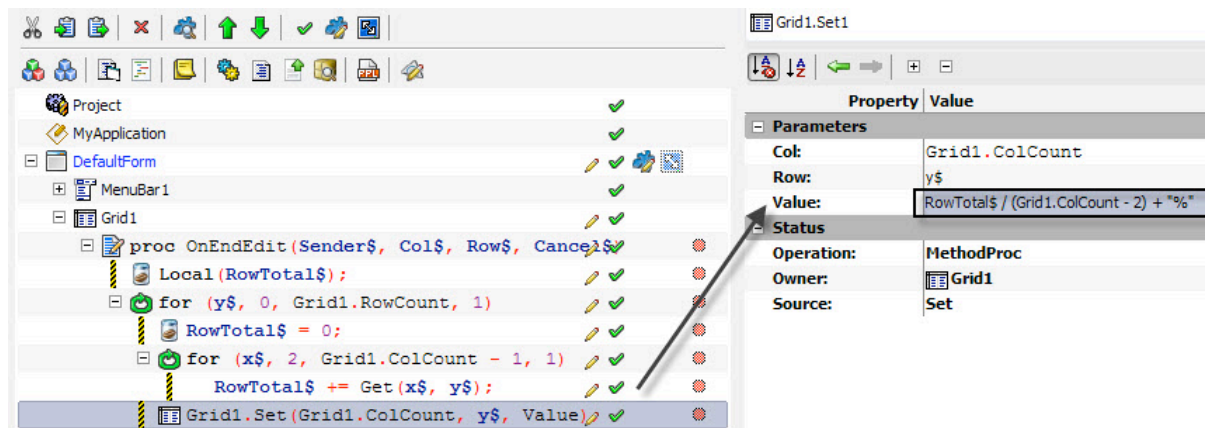


Figure 97: Change Property

- We will also need to round off the values and for that we will use the **RoundEx** method. To round the value to nearest two decimals, we will enclose the **value** property with **RoundEx(value,2)+ "%"** .

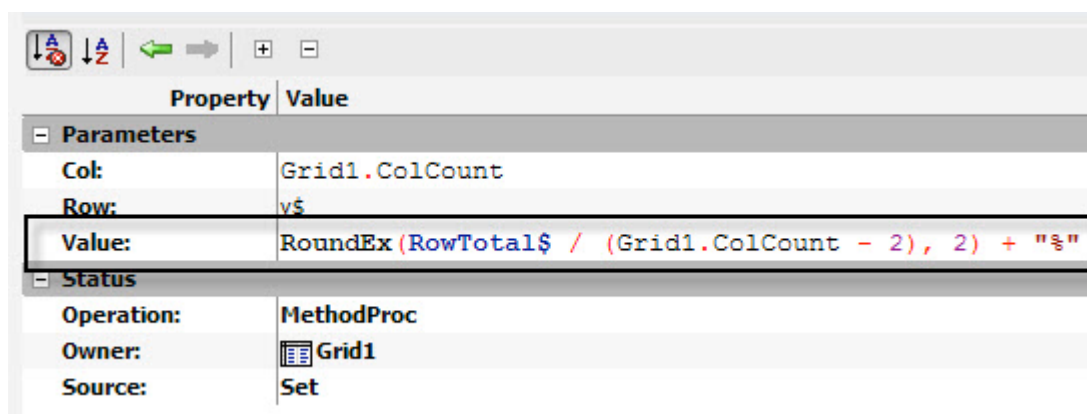


Figure 98: CHange Value property

- Save your project by pressing **Ctrl+S** or go to the **File Menu>Save**. Give an appropriate name in the **Save As..** window and press **Save** to save the file.

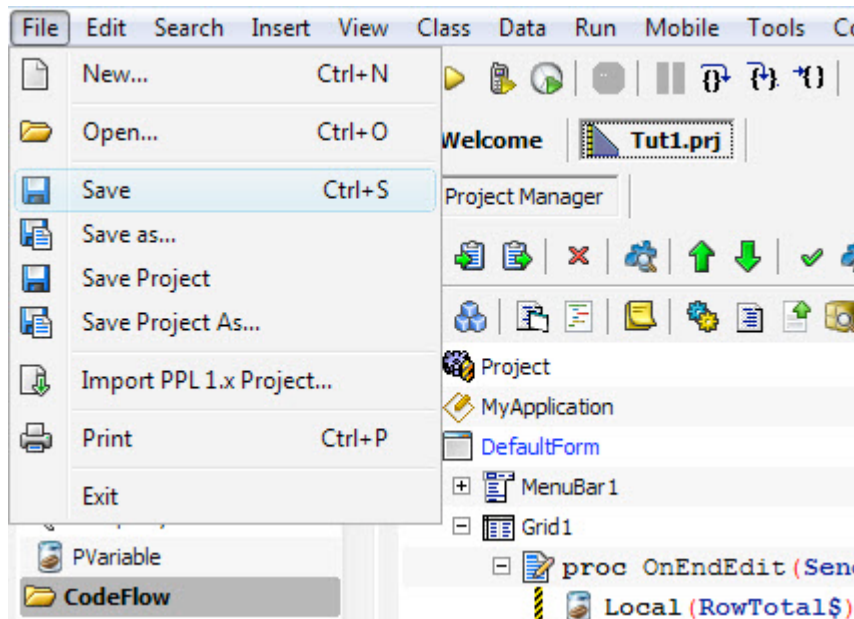


Figure 99: Save Project

- Run the project by pressing **F5** on the keyboard.

DefaultForm

File

| Grade | Marks 1 | Marks 2 | Marks 3 | Total  |
|-------|---------|---------|---------|--------|
| 1     | 80      | 70      | 60      | 70%    |
| 2     | 77      | 66      | 55      | 66%    |
| 3     | 87      | 65      | 68      | 73.33% |
| 4     | 54      | 72      | 94      | 73.33% |
| 5     | 62      | 55      | 71      | 62.67% |

Figure 100: Output

## Creating a PDBGird object

PIDE Users can use a **PDBGird** object to add grid access to a table and its columns.

**PDBGird** not only allows a user to determine the layout of the data contained in a table, it also allows for easier readability of the data for a user. Not only is a user capable of showing data present in the table as a grid, he/she also gains the capability to display headings, column headings and manipulate table data with the help of **PQuery** object.

The **PQuery** object can be used to create queries that run on a table, these queries can be attached with **PDBGird** so that data returned by a query is displayed in a **PDBGird** as a programmer wants. Follow the steps given below to create a **PDBGird** object:

- Start with creating a **Desktop Form** project. For doing this, press **Ctrl+N** or go to **File>New** and select **Desktop Form** project from the **Select New Project Type..** window.

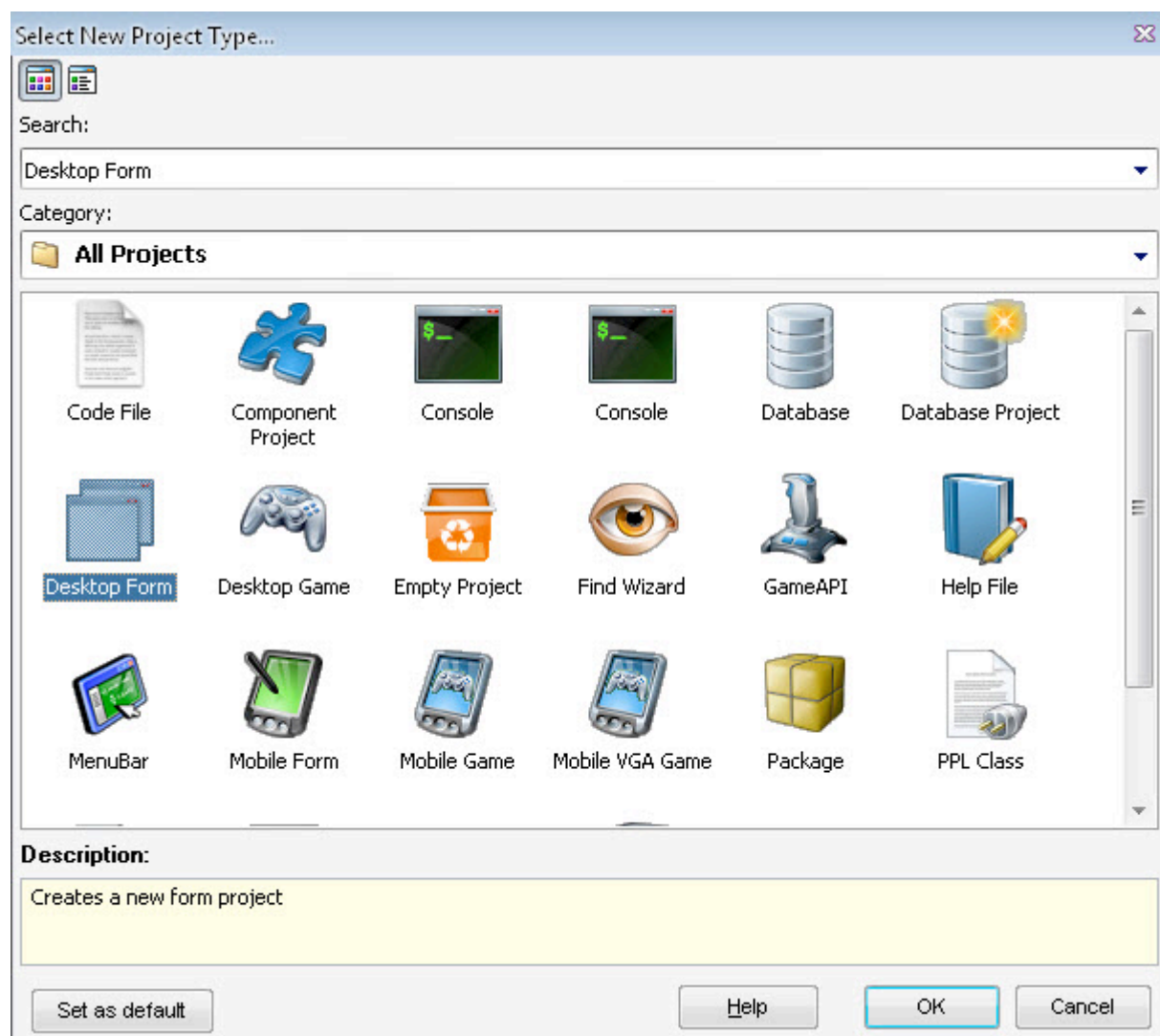


Figure 101: Create new project

- Create a new database and then attach a table to it. Refer to tutorial given above for creating a table with data. In this example, we will use the same database created before in the earlier example.

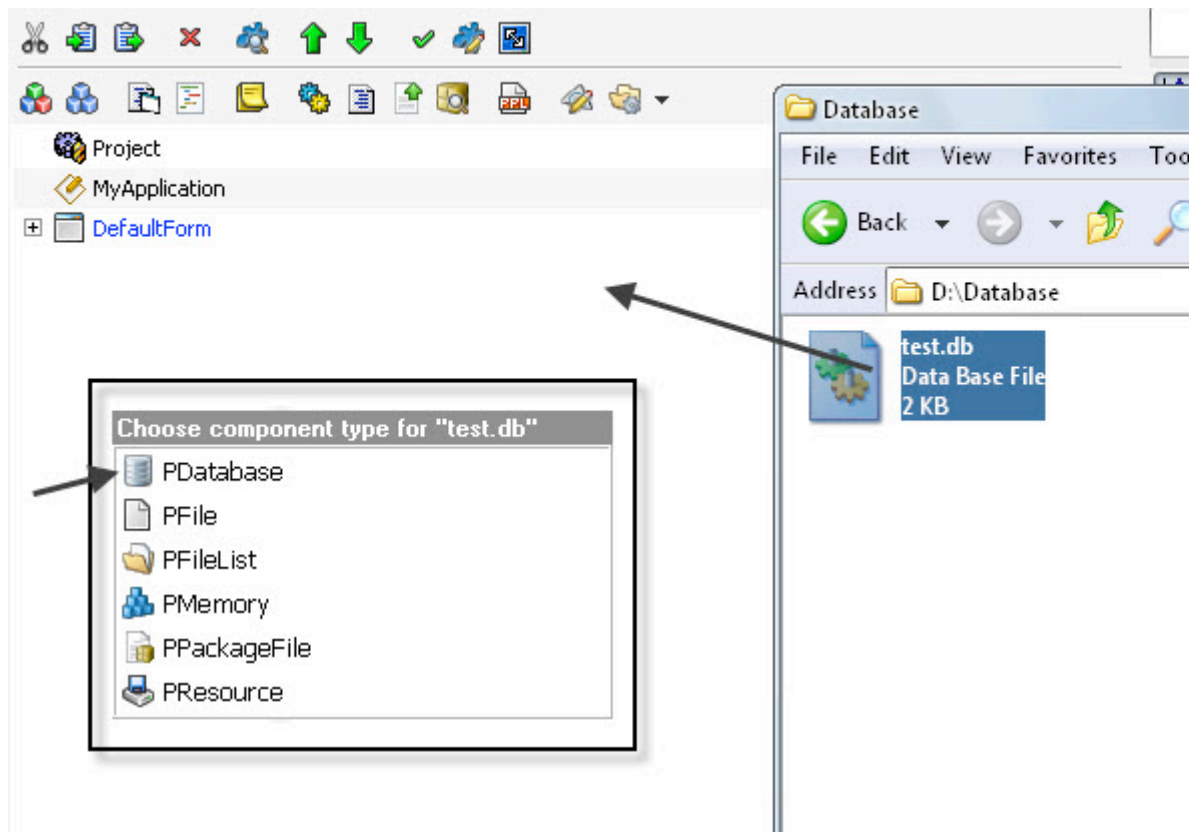


Figure 102: Drag Database

- Importing our database to the project will result in placement of the database on the project.

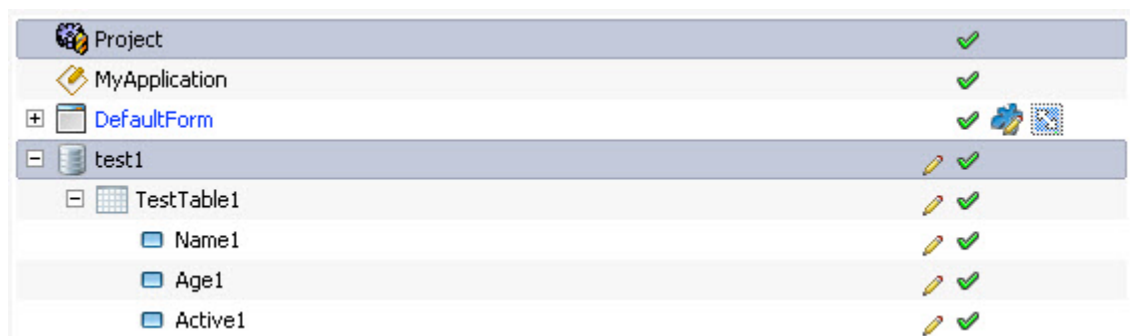


Figure 103: Database with tables and columns

- PDBGird** provides users with the facility to print elements in a table on screen in a grid like format. For printing our table in the database, we will drag the table on the DesktopForm object and select **Add Grid to control** from the context menu that appears.

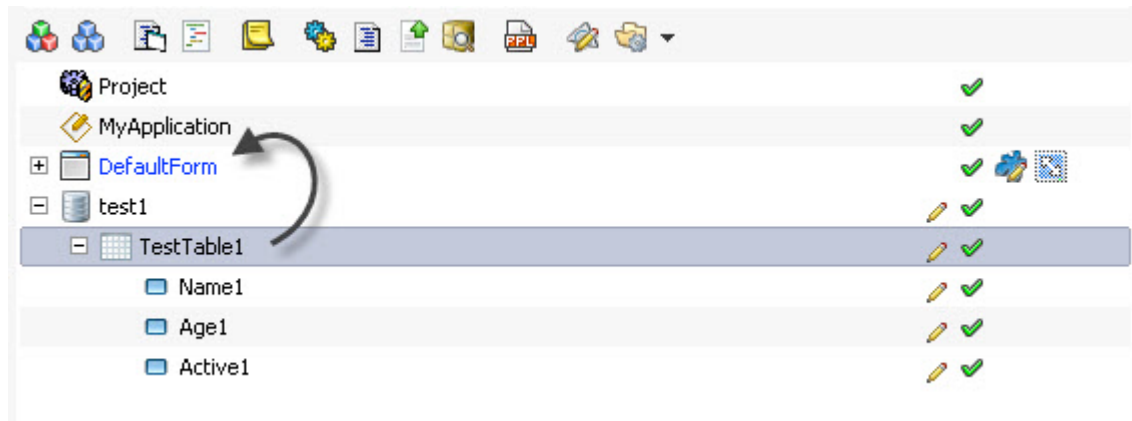


Figure 104: Drop database to Default Form

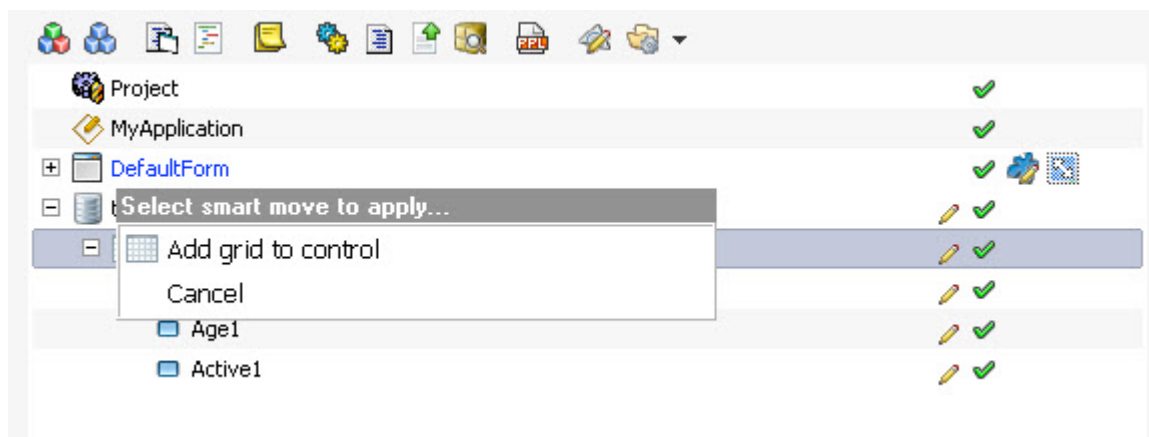


Figure 105: Smart Move

- Once **PDBGird** is available on the form, you can edit its properties to make it more suited for your application.

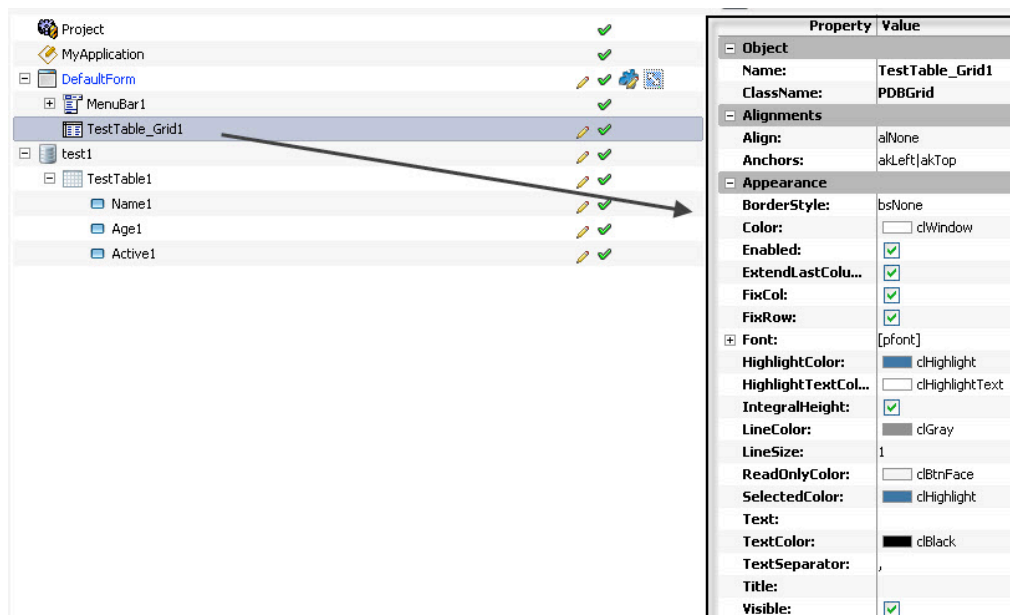


Figure 106: Change properties

- That's it! Save your project by pressing Ctrl+S or go to File>Save As.. to save the file on the hard disk.

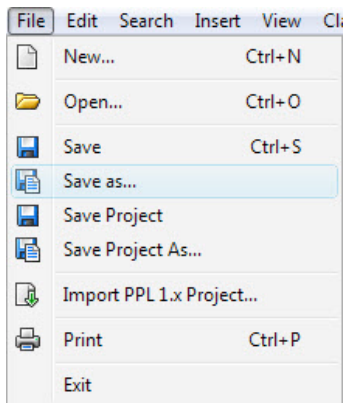


Figure 107: Save project

- Press **F5** to run the program.

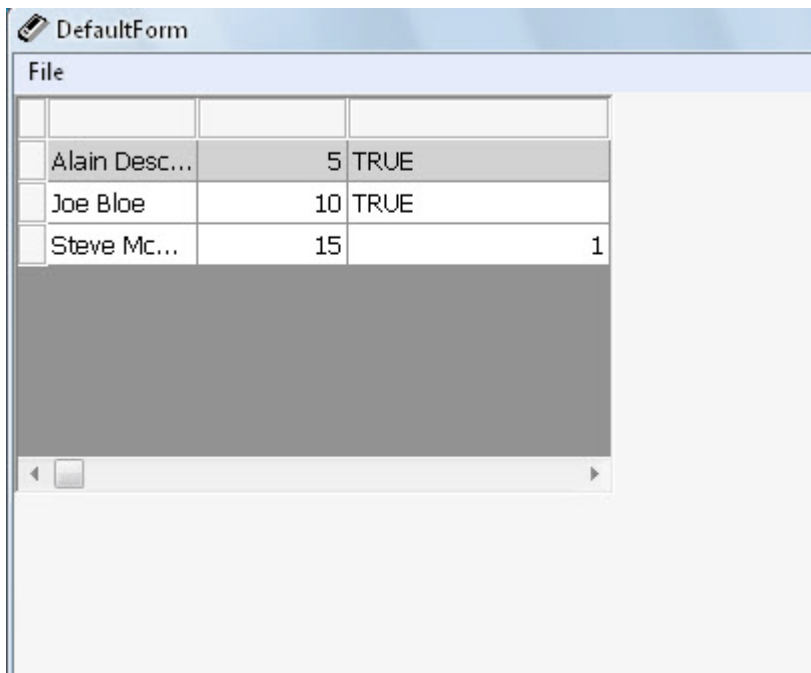


Figure 108: Output

## Using The Report Builder

Report builder is a tool that allows a user to create reports based on the data present in a database. After a database is created, we can easily create a report and then use that report to display data in a summarized form. The Report Builder contains various bands that are used to display data in the desired format. Given below are the various bands used with a report builder:

- **PageHeaderBand:** This is the band that describes the page Header for a report
- **HeaderBand:** This Band determines the Header section of the report
- **DetailBand:** Shows details
- **FooterBand:** As the name suggests, this band is present at the footer of the report
- **PageFooterBand:** Use this band for describing the page footer in a report

There are two ways to create a report builder:

**Method1:** Creating a standalone report builder and using it to create further projects.

Start PIDE and press **Ctrl+N** or go to **File>New** to create a new project.

- Now in the create **Select New Project Type** window, click on the **Report** icon.

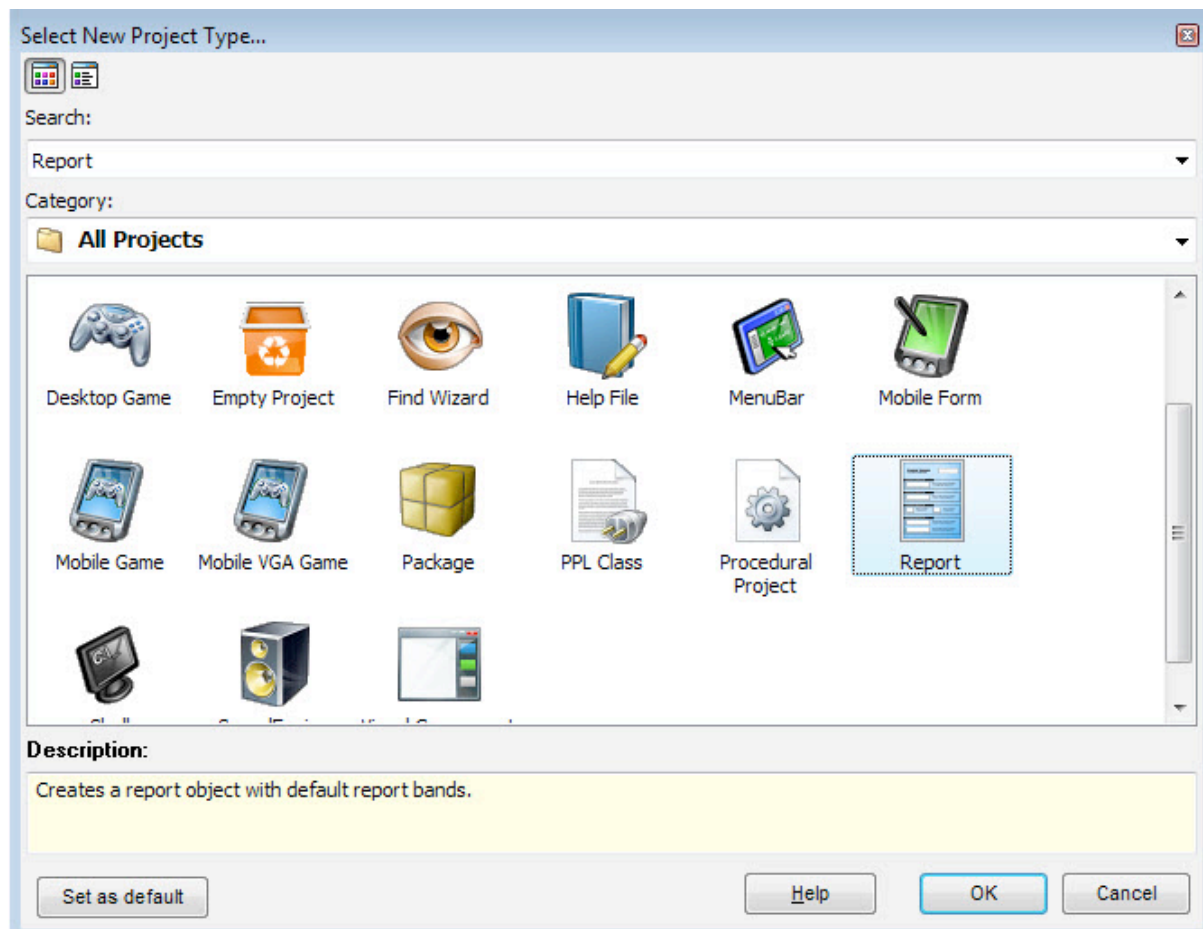


Figure 109: Create new project



- This will create an empty project with just the report and its associated bands.

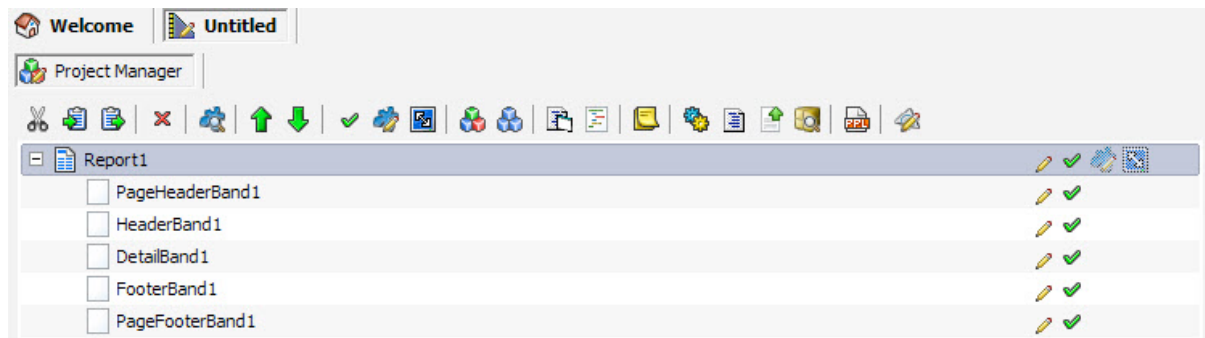


Figure 110: PReport

Now, you can drag various objects from the **Components Panel** and use them in conjunction with the report builder to create a solution with custom reports.

**Method2:** If you are using a current project and need to add a report to it, you can press **Ctrl+N** and select the **Report** option in the **Select New Project Type** window to create a report object with the existing project.

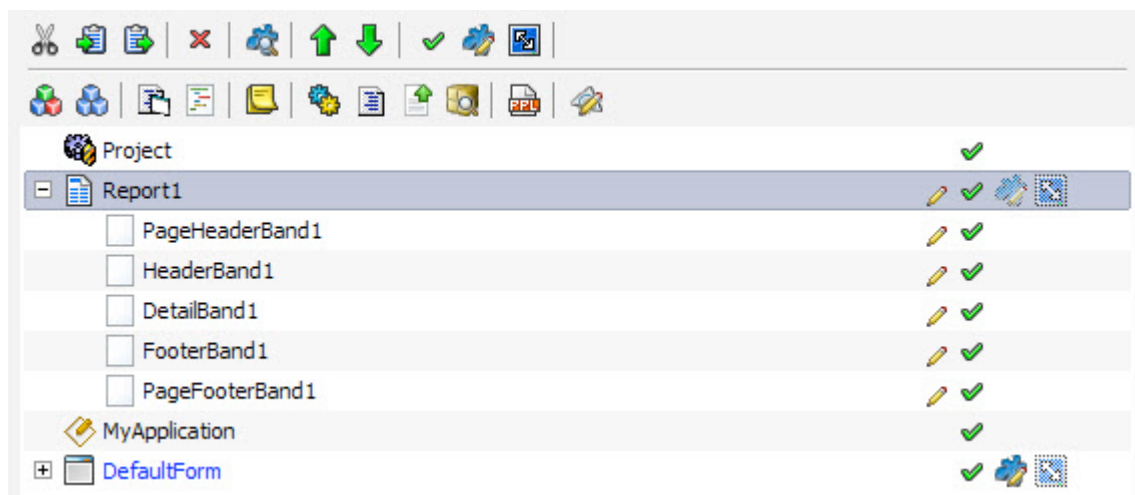


Figure 111: PReport in existing project

**Method3:** Using Database Project. With database project, users will be able to create automatically generated copy of reports in their database driven forms. This method requires minimum configuration and is best suited for creating one click database applications.

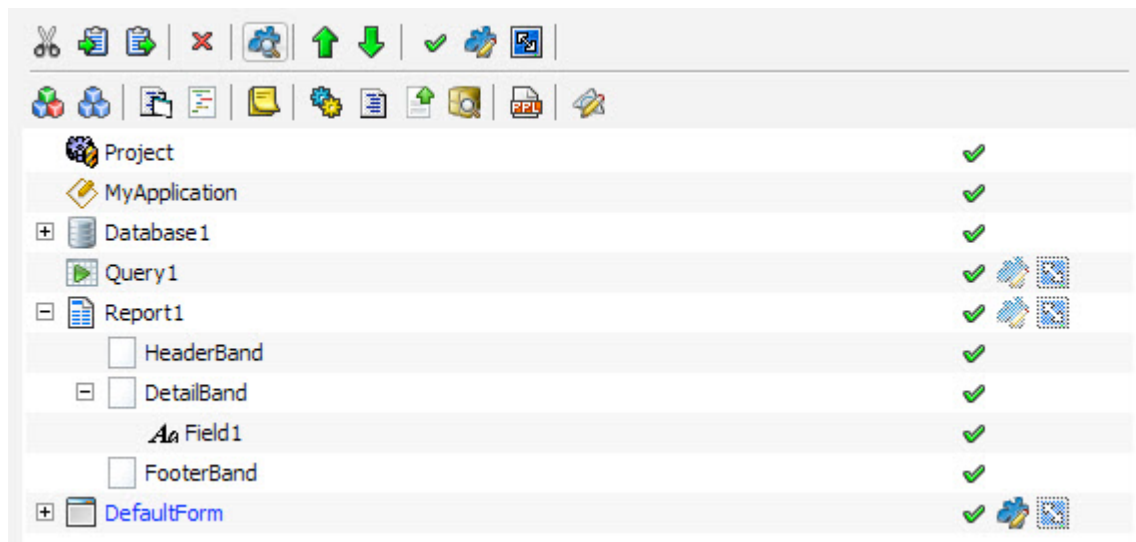


Figure 112: PReport with Database

## Using PValue, PList, PValueList And PObject Objects

### Using PList

A **PList** is an object that can be used to hold a number of string or numerical values. Users can use **PList** to insert, append as well as delete values as they like. Apart from this, users can also use **PList** object to search, sort as well as get the values as text format. Given below is an example that would allow you to understand **PList** in a better way. In this example, we will sort five values according to their alphabetical order:

- Open PIDE2 and create a new Console project from the **Select New Project Type...** window. Using **Ctrl+N** is the keyboard shortcut for displaying **Select New Project Type...** window; alternatively you can also use **File Menu>New**.

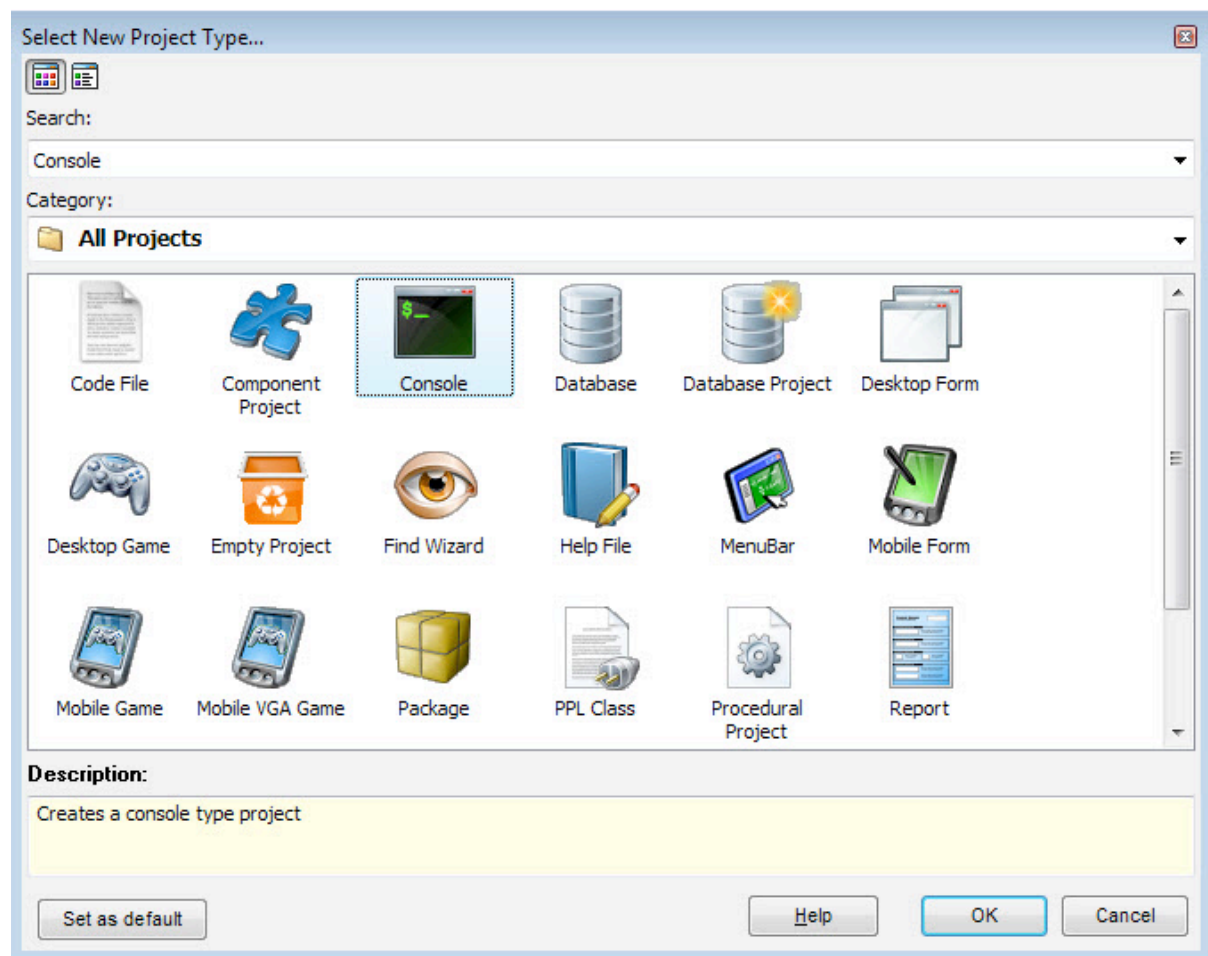


Figure 113: Create new project

- In the **Project Manager**, drag a **PList** object from the **Components Panel** and place it above the project **DefaultConsole**.

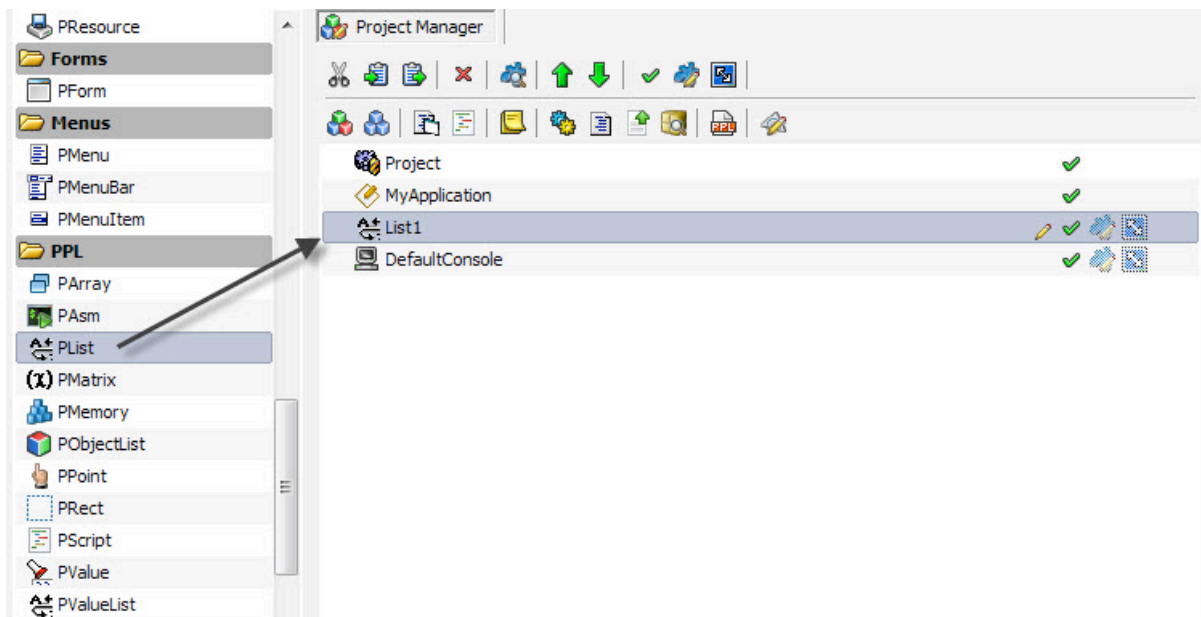


Figure 114: Drag PList

- Now we will create an **OnCreate** event on **DefaultConsole**. We need to create an event that will trigger an action after our objects have been created thus, we will use an **OnCreate** event. **OnCreate** event can be created by two ways; either by double clicking the **DefaultConsole** object or by selecting **OnCreate** in the Events section of the right click context menu of **DefaultConsole**.

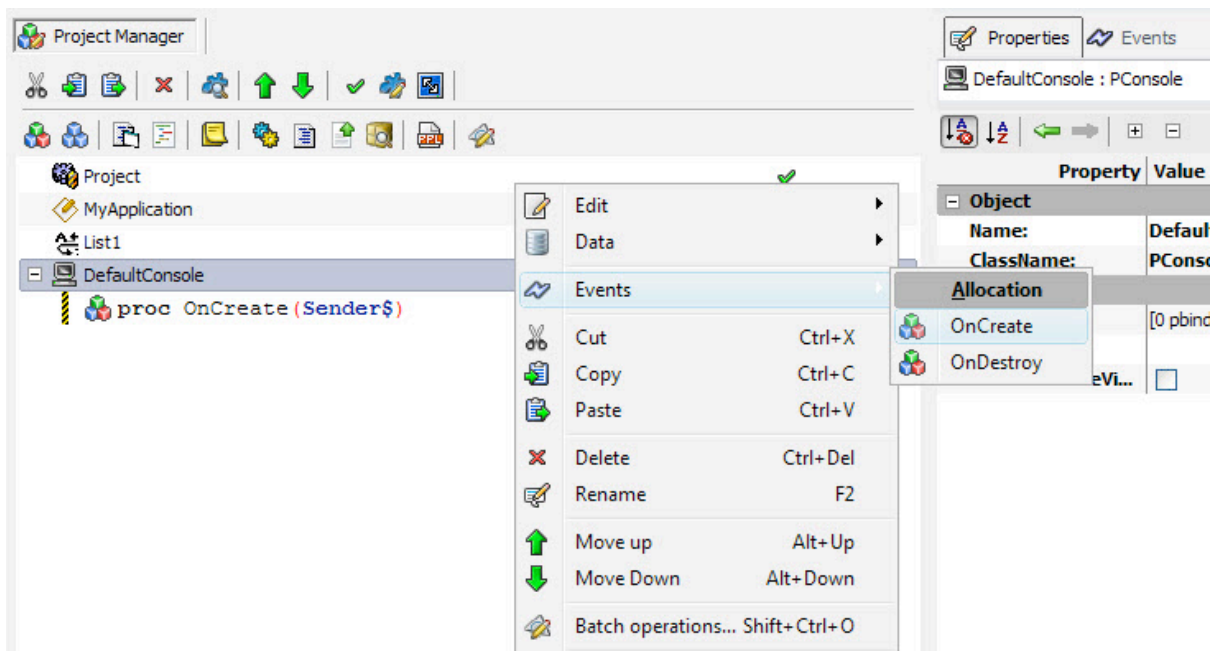


Figure 115: Create Event

- Now drag the **PList** object on **OnCreate** event. This will lead to a **Code Completion** window.

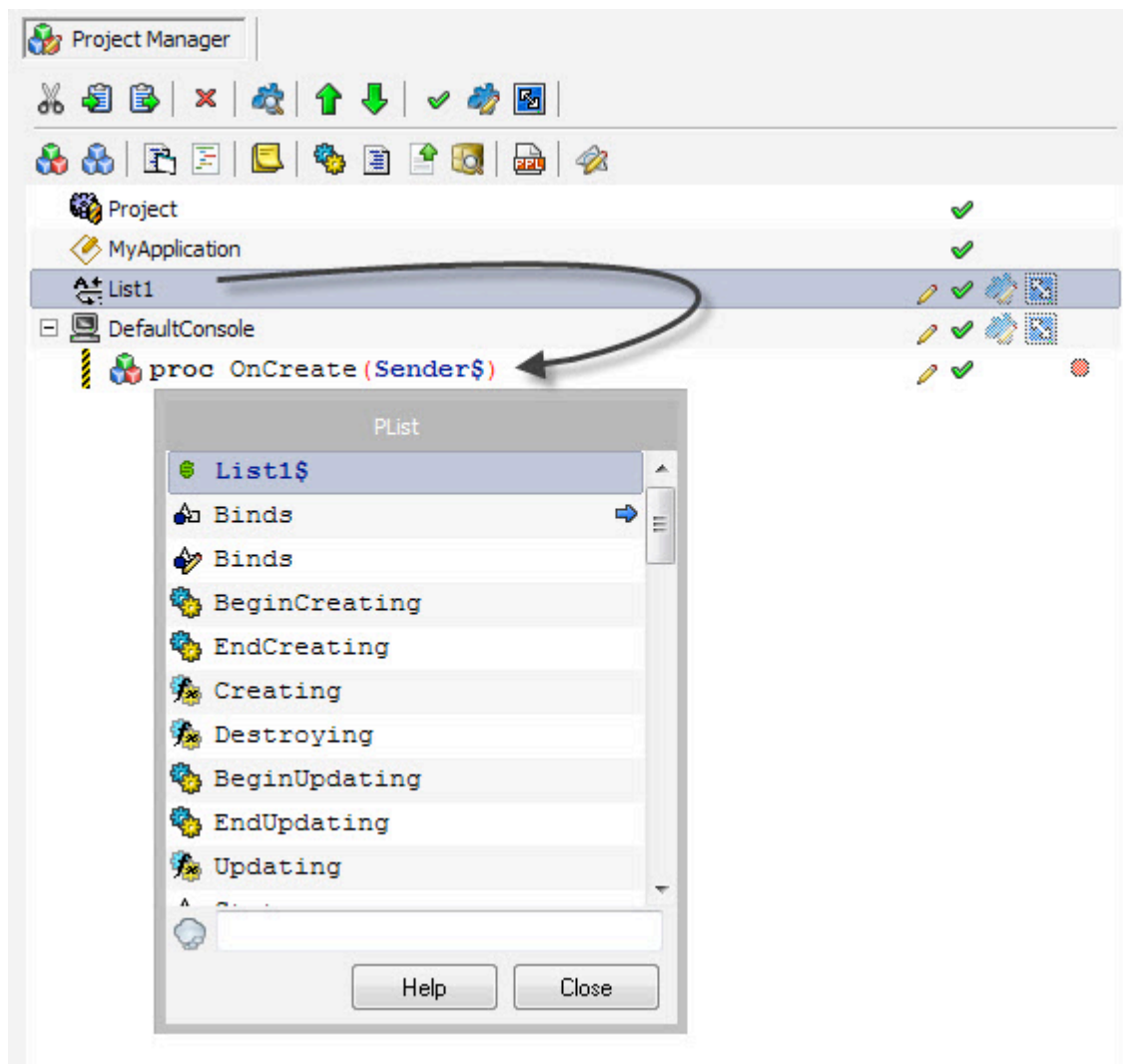


Figure 116: Drag PList

- Select **Add** property from the **Code Completion** window.

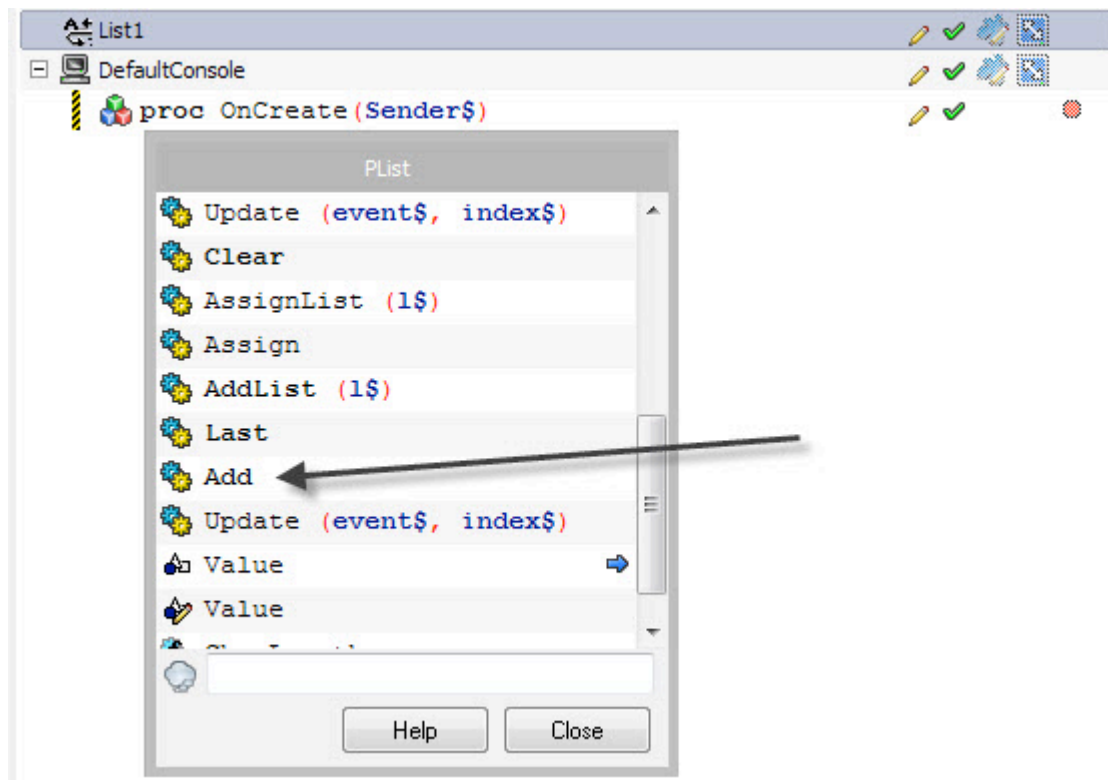


Figure 117: Select Add in Code Complete window

- Click **List.Add()** and write a value in its **Expr** property.

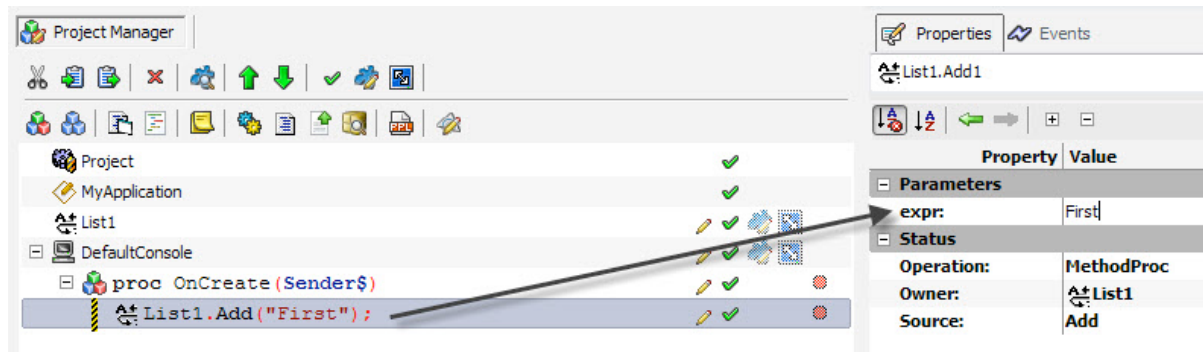


Figure 118: CHange Property

- Similarly, drag **PList** object on the **OnCreate** event and select the **Add** property from the **Code Completion** window. Writing another value in the **Expr** property of this **List.Add()** element will result in creation of one more value in the **PList** object.

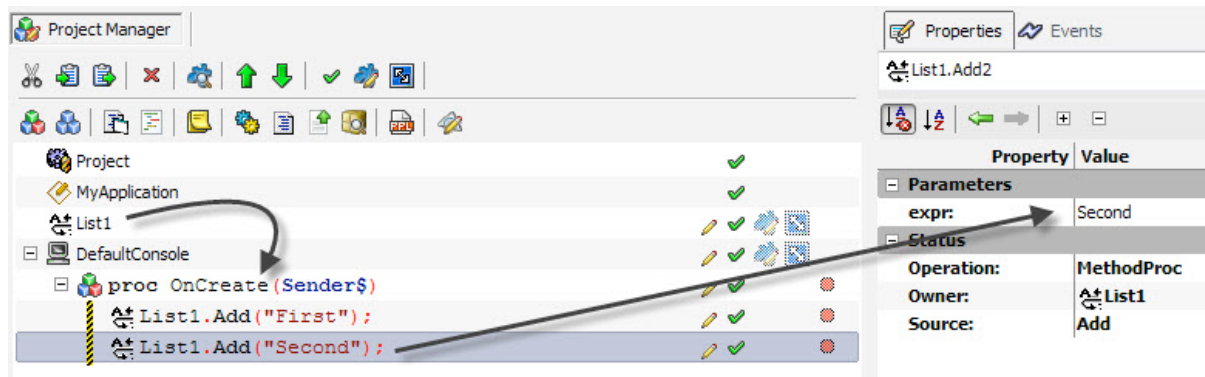


Figure 119: Change Expr Property

- We can create as many values in the **PList** object as we want.

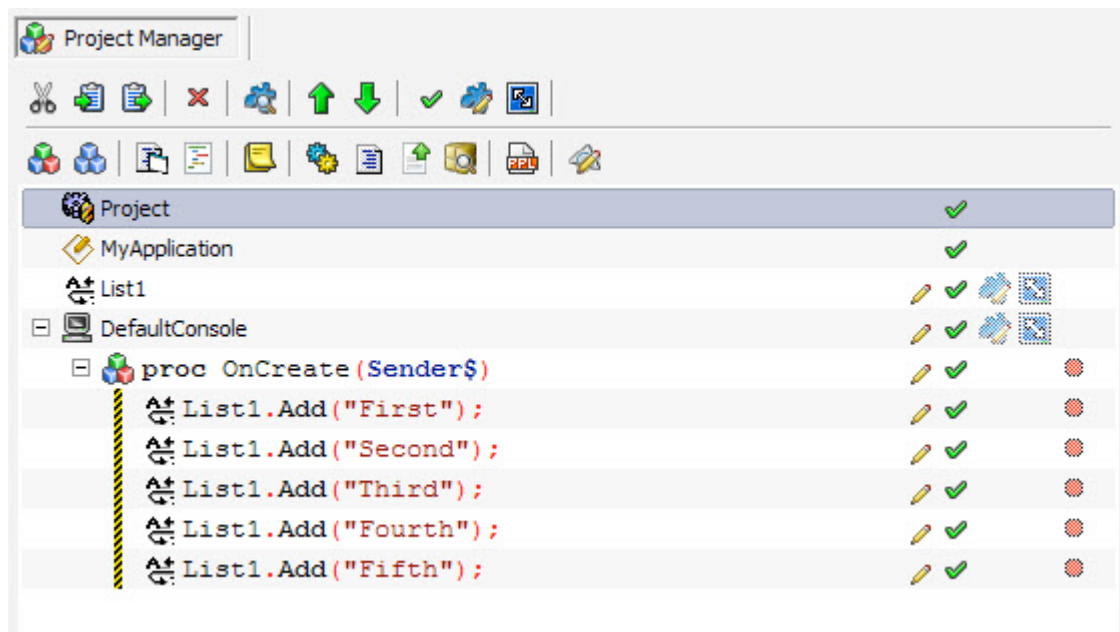


Figure 120: Add List

- Now we will sort the values stored in our **PList** object. For doing so, drag the **PList** object on the **OnCreate** event while holding the **Alt** key on the keyboard. This will allow us to use the object when the **DefaultConsole** is created. The **Alt+drag** operation will result in a **Code Completion** window.



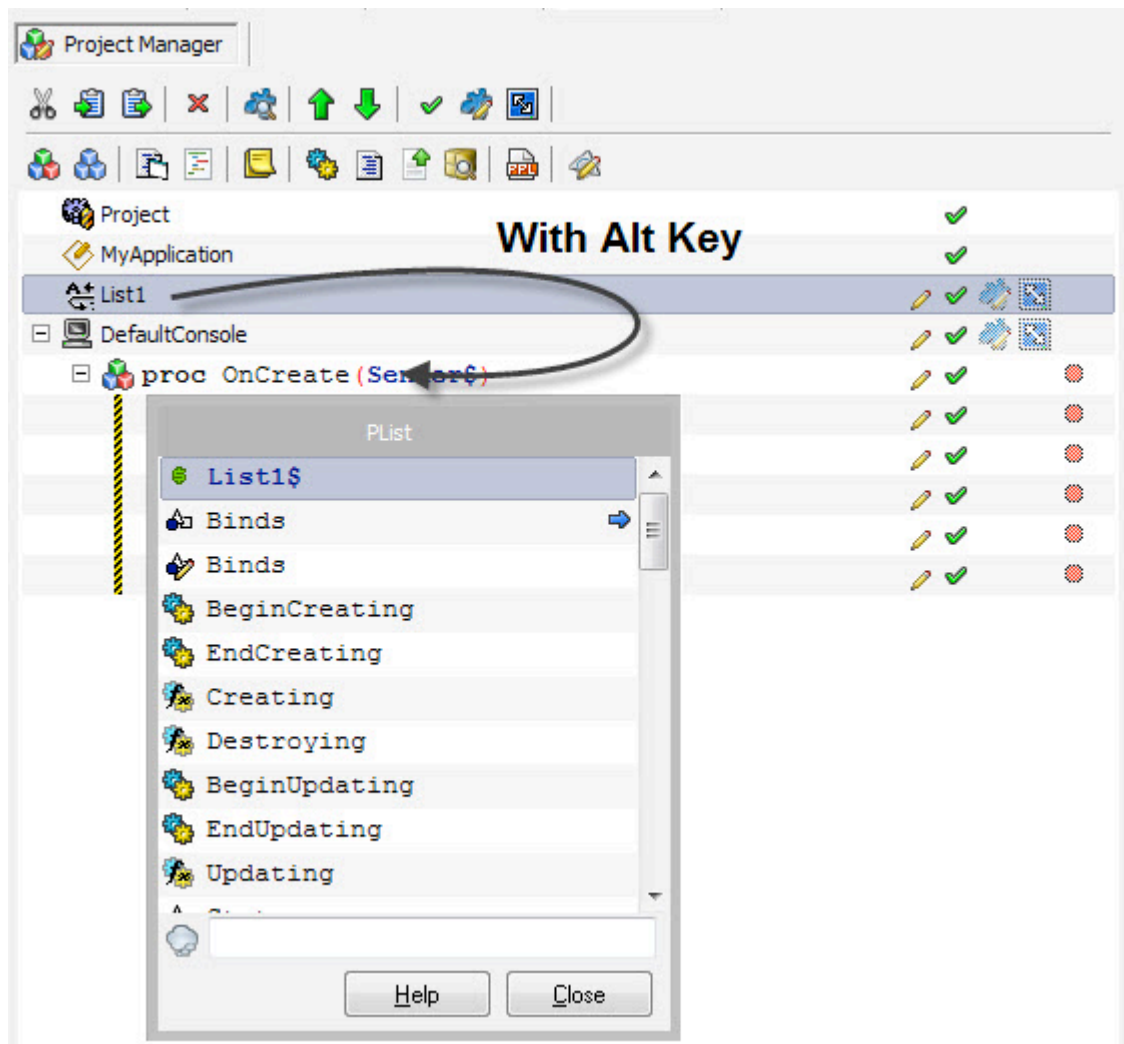


Figure 121: Drag with ALT key

- In the **Code Completion** window, select Sort property.

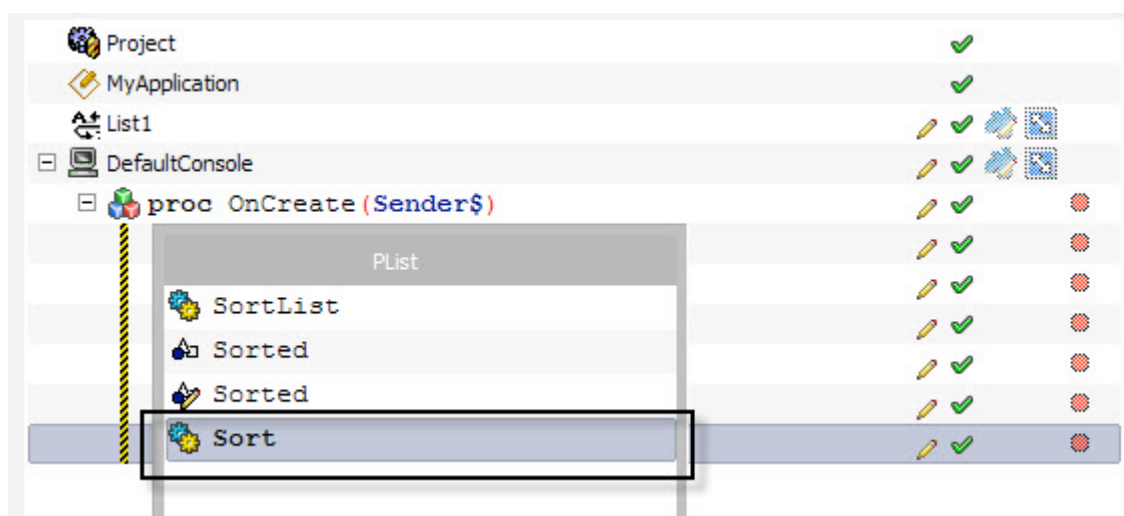


Figure 122: Code Completion window



Figure 123: Sort list

- Now we need to print our **PList**. For printing the values, we will use the help of **PrintLine** command.
- Click **OnCreate** event and press **Ctrl+Space** bar to bring **Code Completion** window.

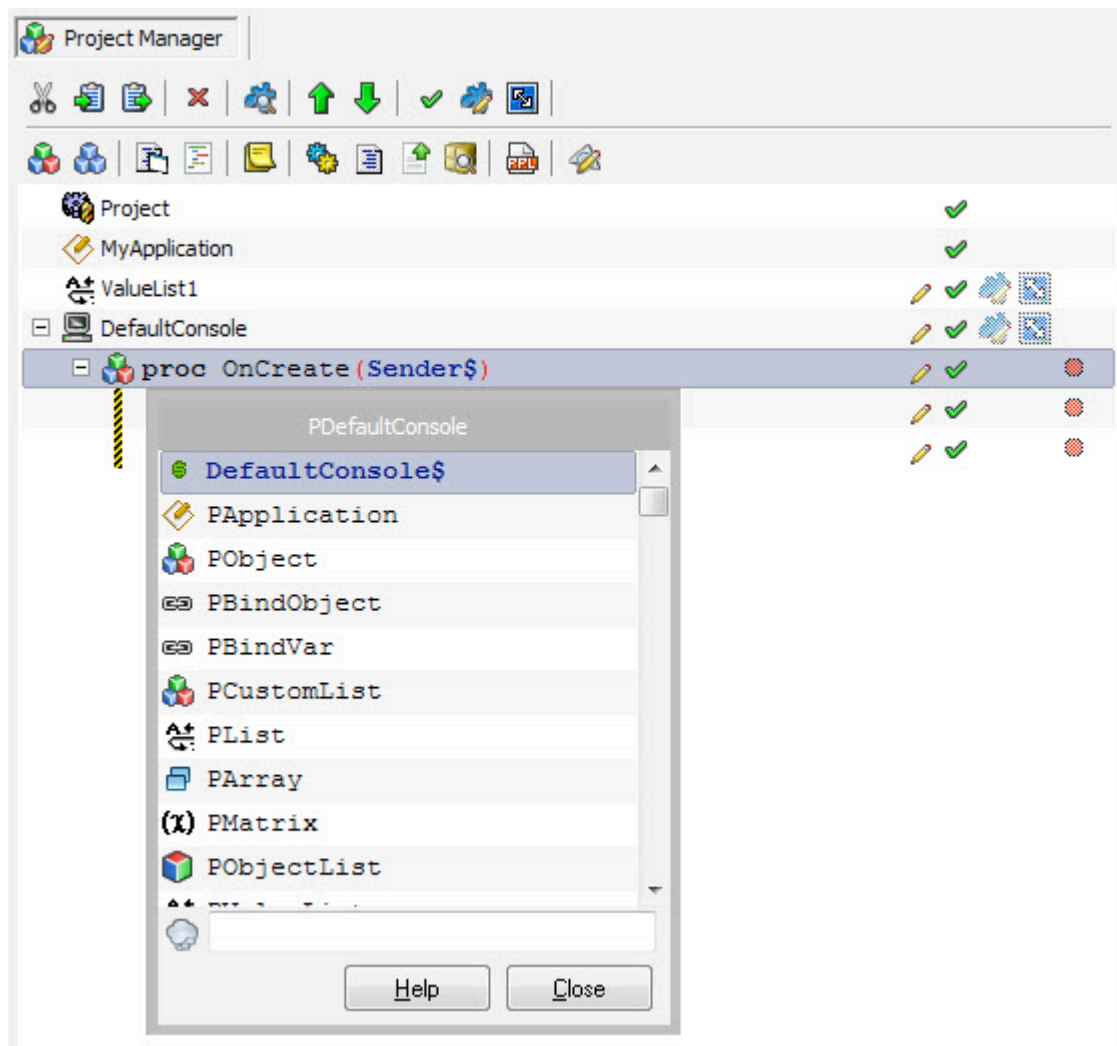


Figure 124: Code Complete window

- Select **PrintLine** object and drag the **PList** object to the **Value** property of **PrintLine**.

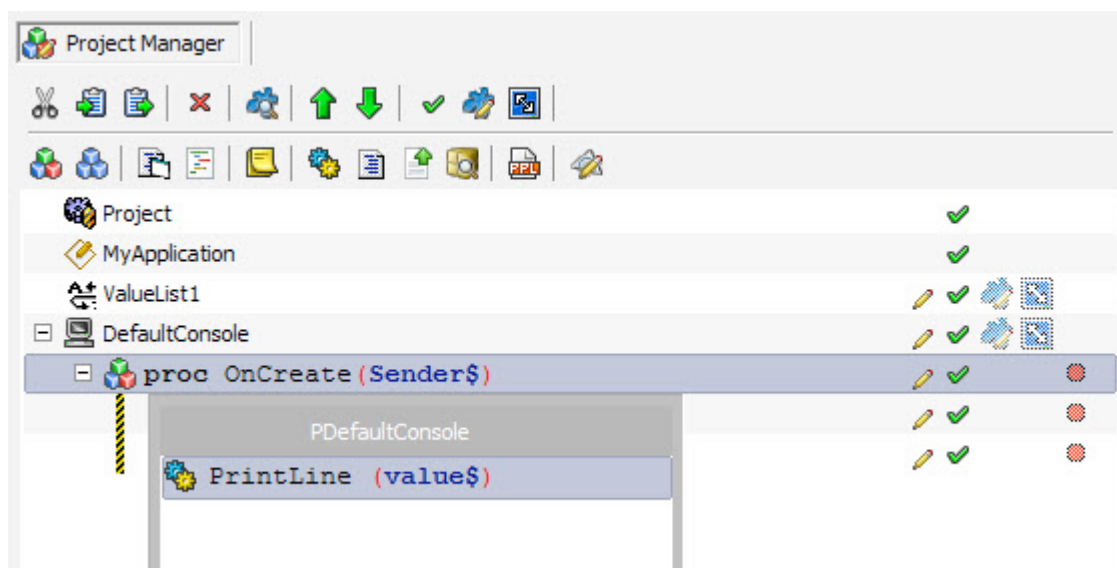


Figure 125: The value property

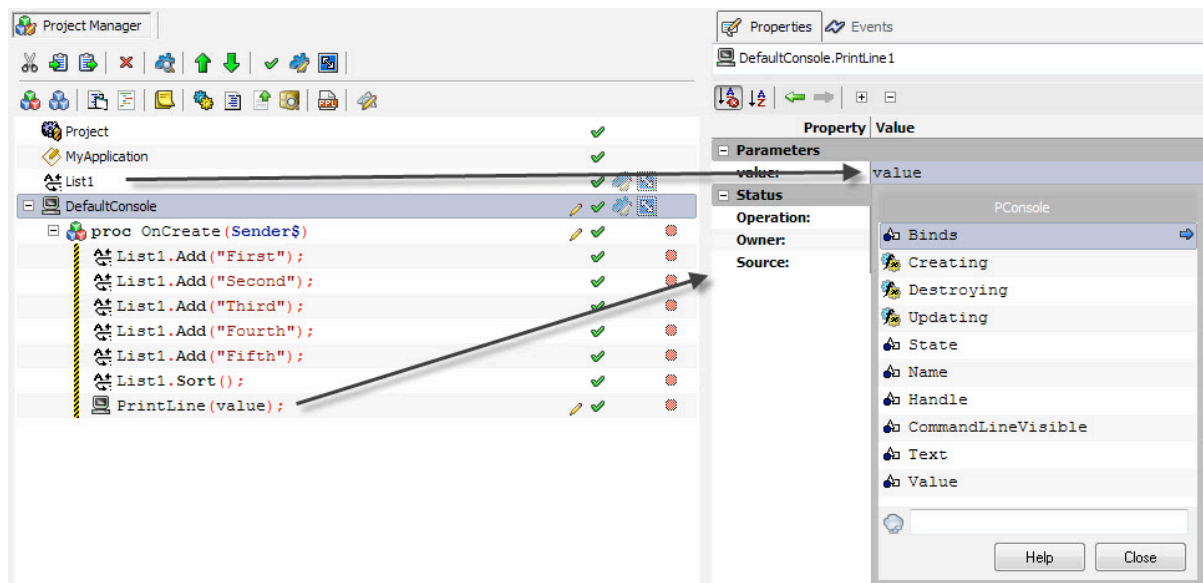


Figure 126: Change the value property

- The drag operation above will result in a **Code Completion** window; select the **Text** command in it. Doing this will pass the text of **PValueList** to the **PrintLine** method which will then print it on the screen.

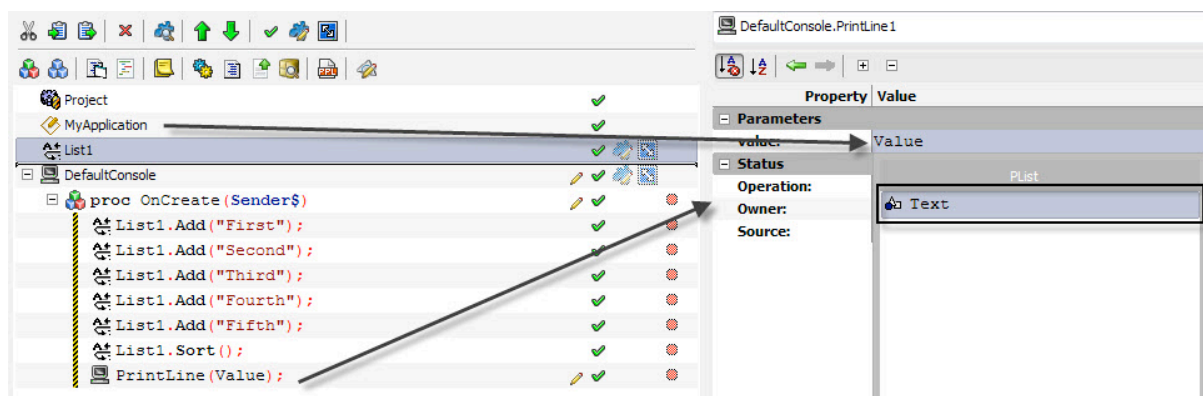


Figure 127: Add Value

- Save your project by pressing **Ctrl+S** or go to the **File Menu>Save**. Give an appropriate name in the **Save As..** window and press **Save** to save the file.

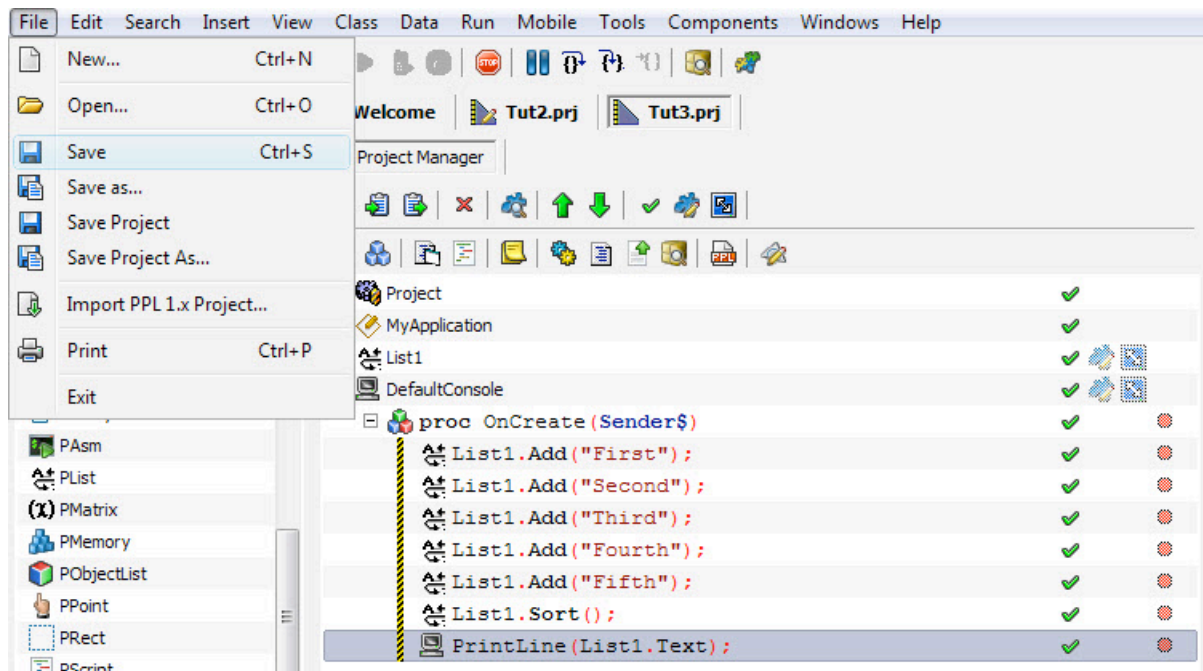


Figure 128 Save Project

- After saving the project, press **F5** or run the project to see the output.

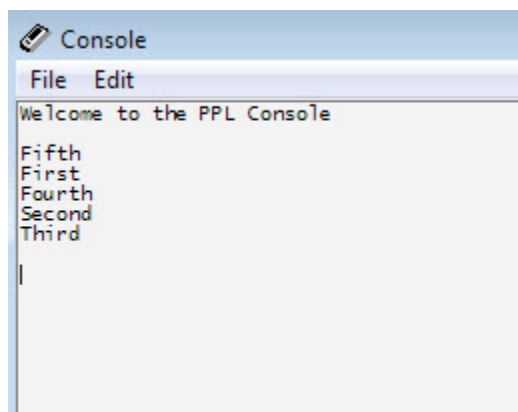


Figure 129: Output

Note: By placing the **PValueList** before **DefaultConsole**, we are assuring that **PValueList** is created before **DefaultConsole** when the project compilation happens.

## Using PValue

Just like a **PVariable**, **PValue** object can contain either a numerical value or a string value. Using **PValue** is a great way to use methods like **Uppercase**, **Random** etc to your application. Given below is an example that will show you how to use **PValue** as a container for various values and tweaking them with its help. In the example given below we will convert a string of alphabets in lowercase to uppercase with the help of **PValue**:

- Open PIDE2 and create a new **Console** project from the **Select New Project Type...** window. Using **Ctrl+N** is the keyboard shortcut for displaying **Select New Project Type...** window; alternatively you can also use **File Menu>New** also.

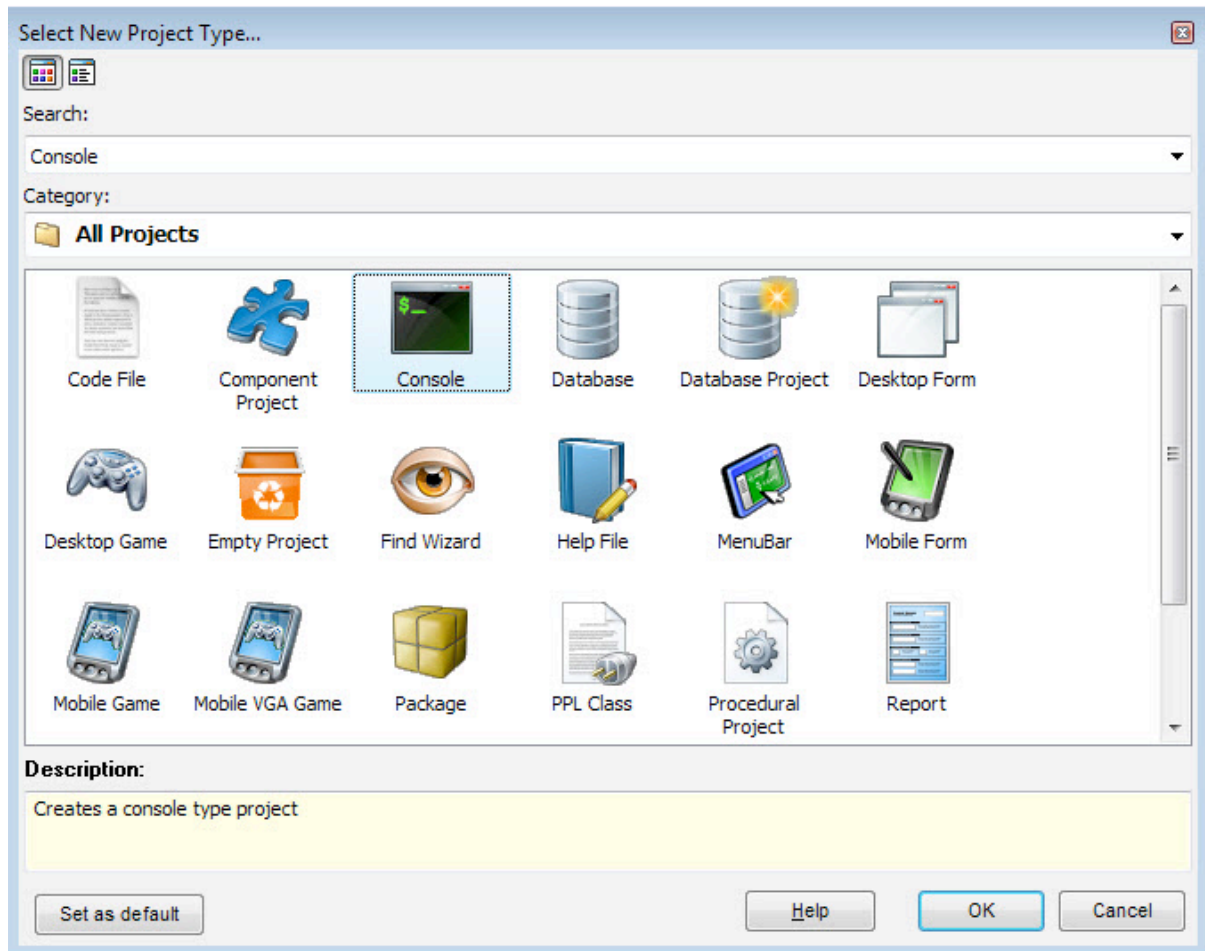


Figure 130: Create new project

- Create an **OnCreate** event on **DefaultConsole**. We need to create an event that will trigger an action after our objects have been created thus, we will use an **OnCreate** event. **OnCreate** event can be created by two ways; either by double clicking the **DefaultConsole** object or by selecting **OnCreate** in the Events section of the right click context menu of **DefaultConsole**.



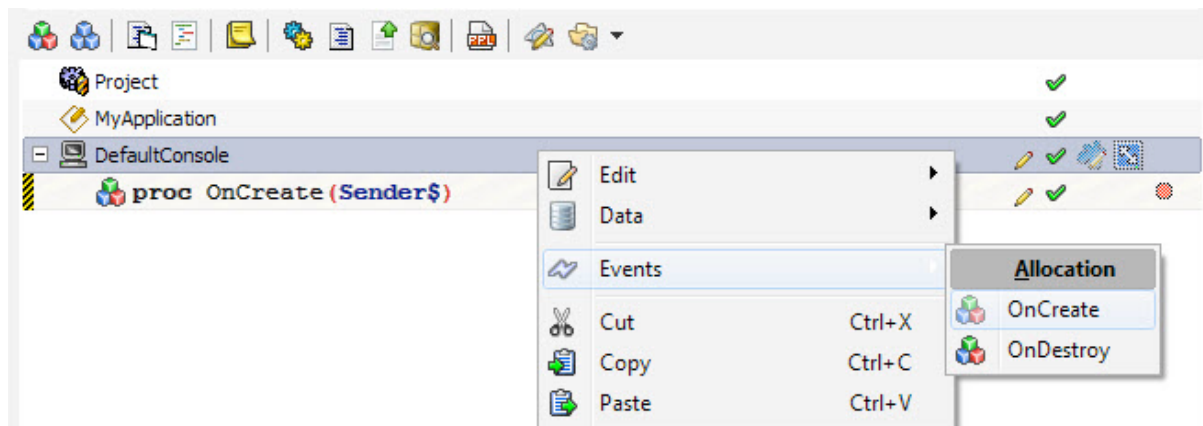


Figure 131: OnCreate event

- After an **OnCreate** event is created, drag a **PValue** object on it. This will create a **PValue** object declaration and also trigger a **Code Complete** menu.

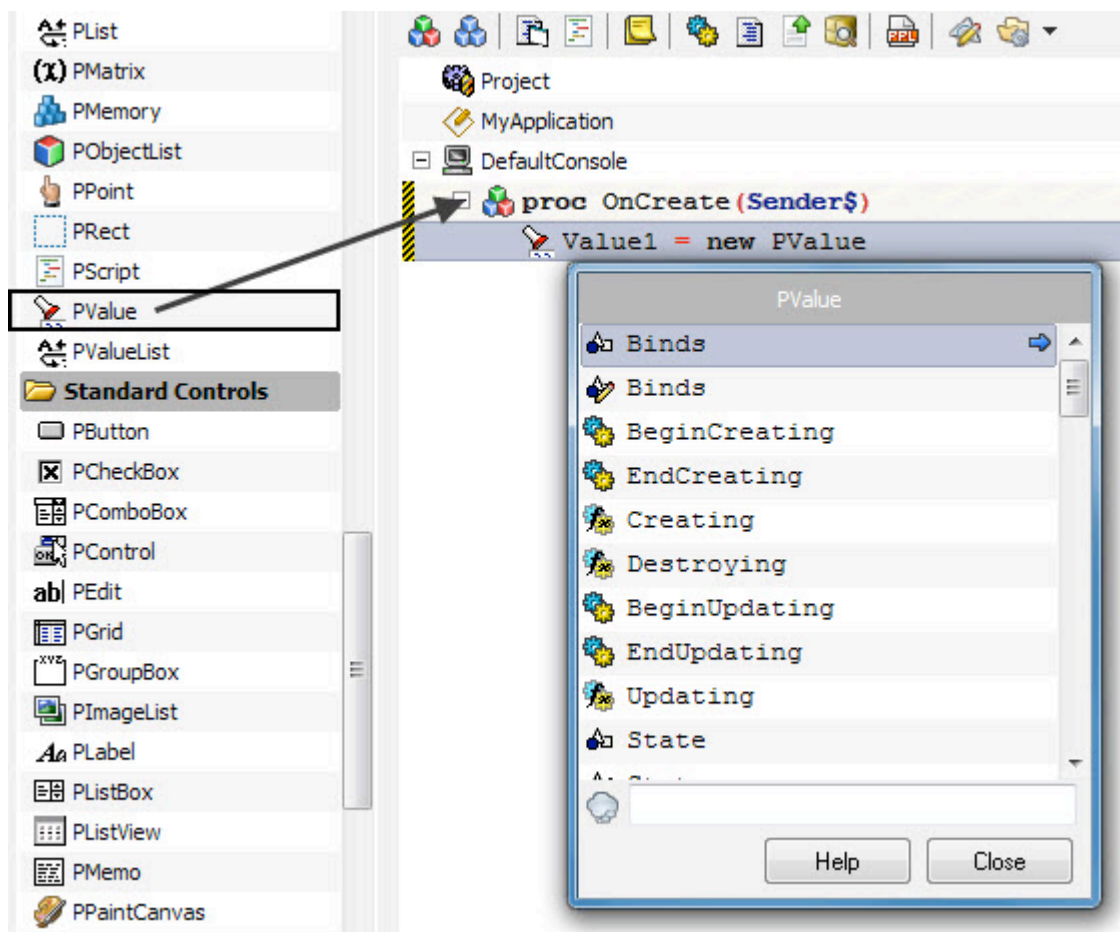


Figure 132: PValue

- In the **Code Complete** menu, select value. This would create a **value1.value=;** statement.



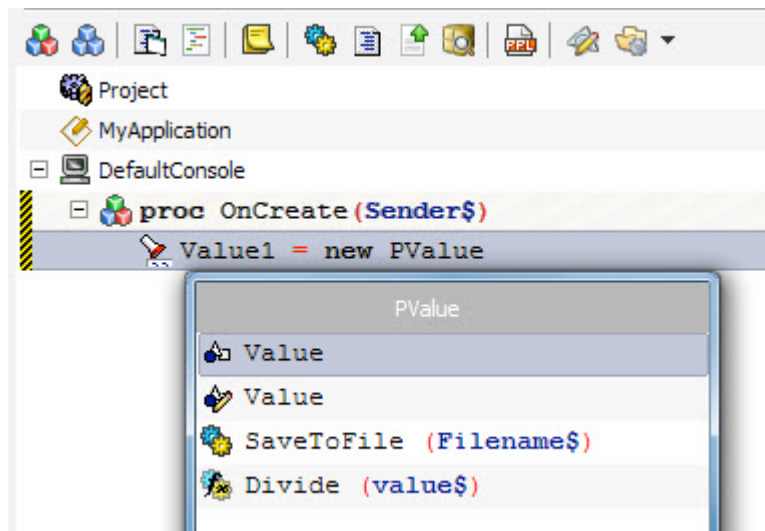


Figure 133: Value property

- Click on the newly created **Value1.value=;** statement and change its **Expr** property to "this all is in small but not for long".

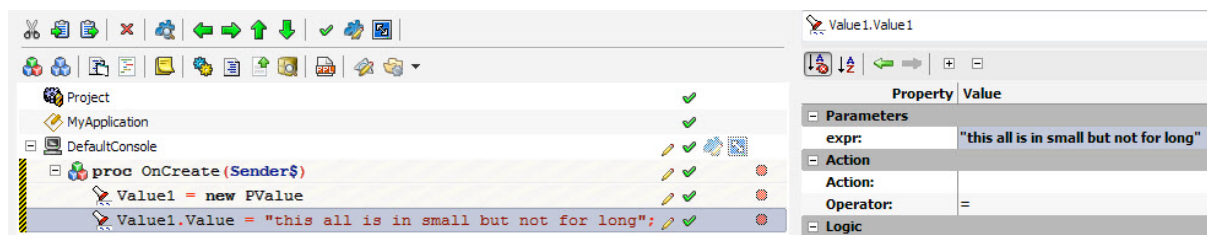


Figure 134: Change Expr property

- Now, click on the **OnCreate** event and press **Ctrl+Space** bar to trigger code complete menu. Select **ShowMessage()** from it. This will be used to display the string in **Uppercase** in a message box.

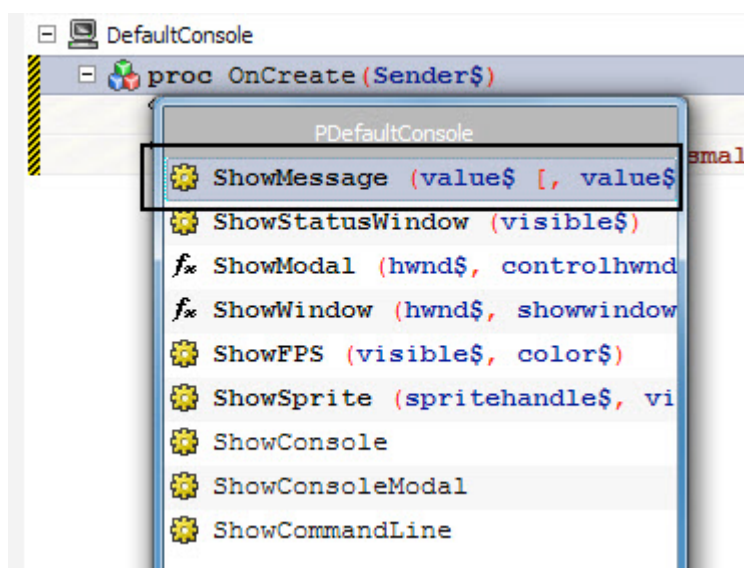


Figure 135: Select ShowMessage

- Drag the **Value1** object to the value property of **ShowMessage** and select **UpperCase** from the code complete menu that appears.

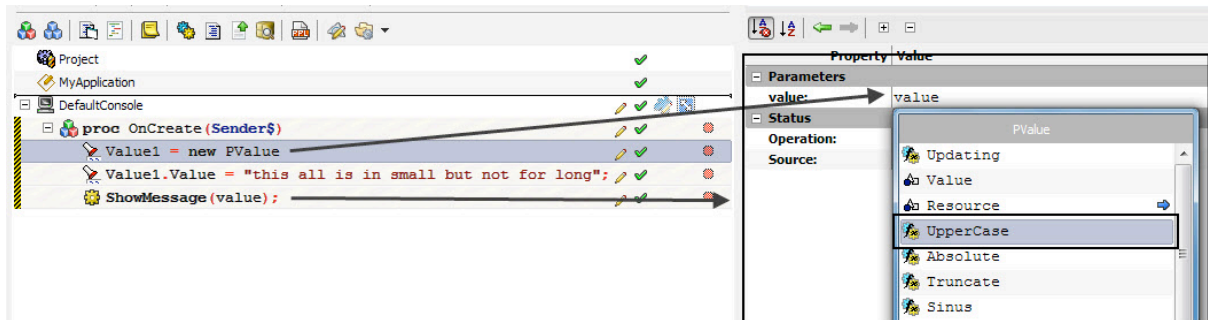


Figure 136: Change UpperCase from Code Completion

- That's it! You are ready with an application that converts lowercase letters to uppercase by using **PValue**. Save your project by pressing **Ctrl+S** or go to the **File Menu>Save**. Give an appropriate name in the **Save As..** window and press **Save** to save the file.

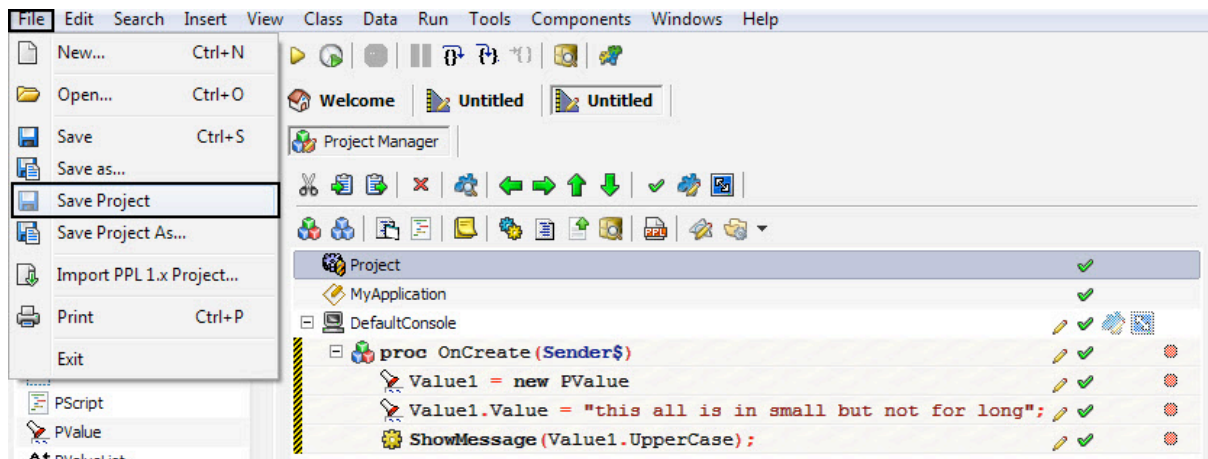


Figure 137: Save Project

- Press **F5** to run the project and see the letters in uppercase.

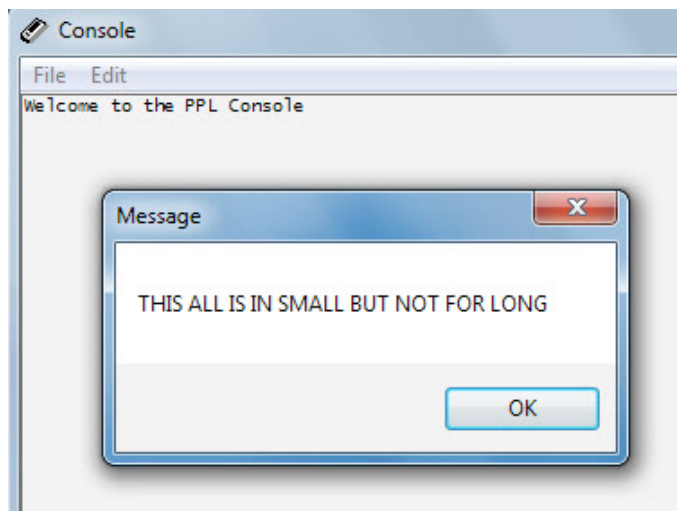


Figure 138: Output

## Using PValueList

Like the name suggests, a **PValueList** is a **PList** object that contains **PValue** objects in it. Given below is an example that will convert two **PValueList** values to uppercase.

- Open PIDE2 and create a new Console project from the **Select New Project Type...** window. Using **Ctrl+N** is the keyboard shortcut for displaying **Select New Project Type...** window; alternatively you can also use **File Menu>New** also.

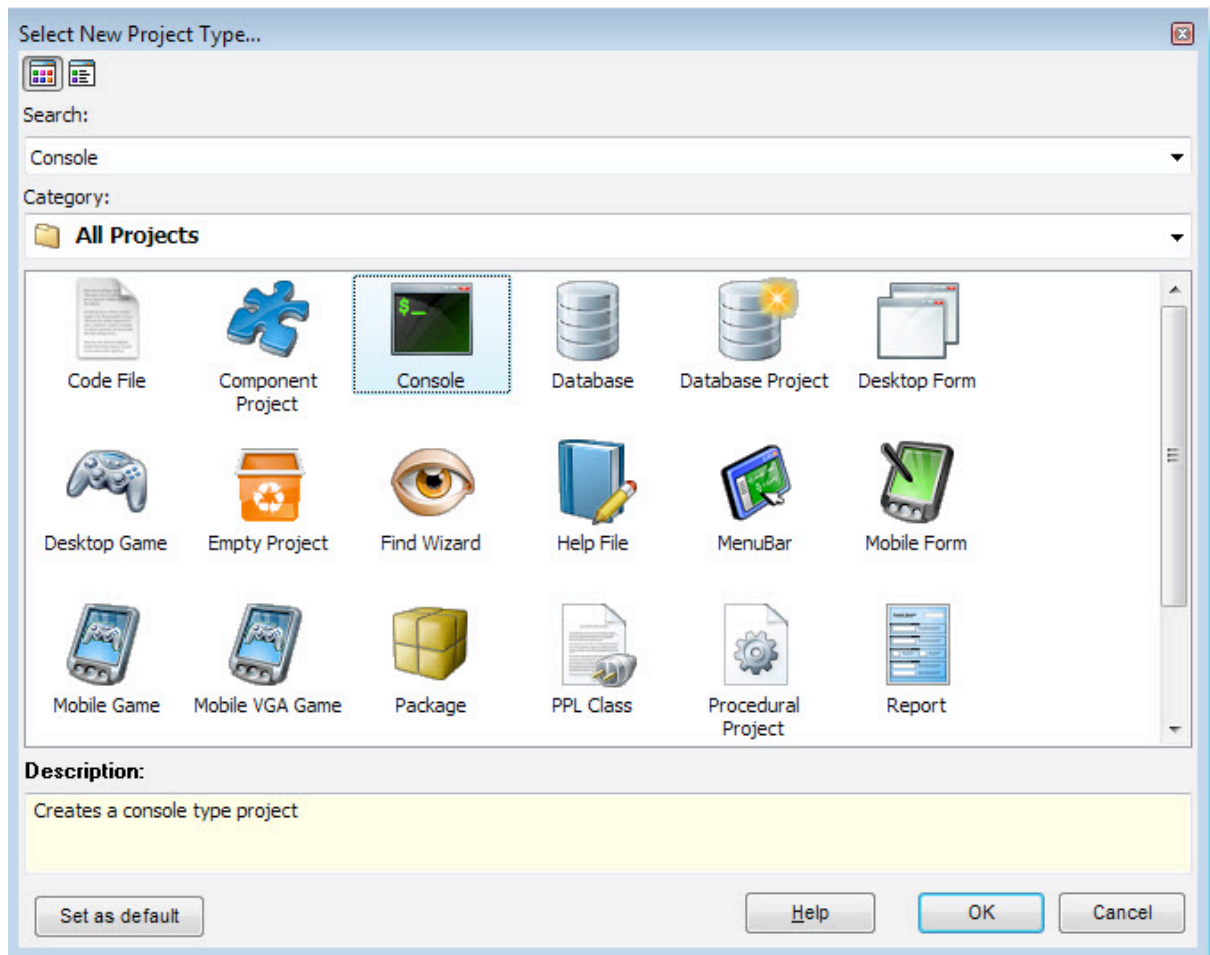


Figure 139: Create New Project

- In the **Project Manager**, drag a **PValueList** object from the **Components Panel** and place it above the project **DefaultConsole**.

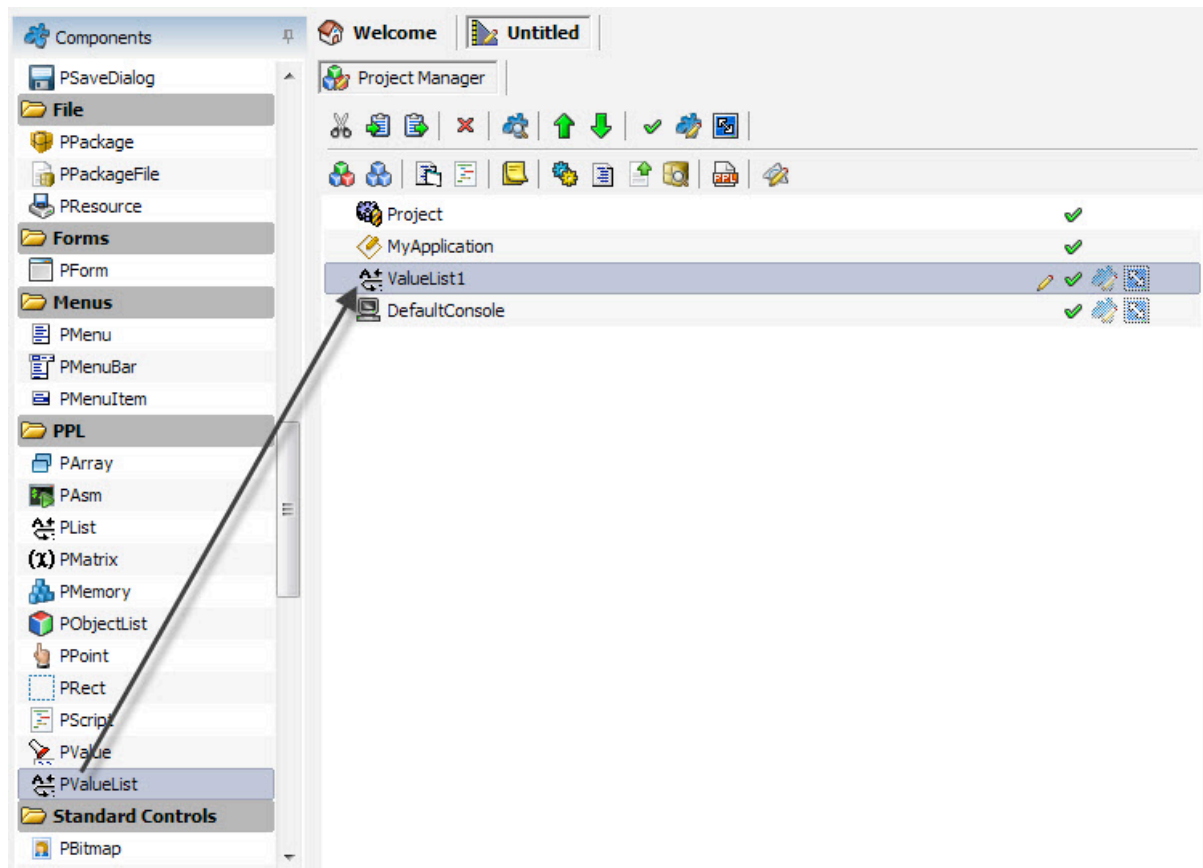


Figure 140: Drag ValueList

- Create an **OnCreate** event on **DefaultConsole**. We need to create an event that will trigger an action after our objects have been created thus, we will use an **OnCreate** event. **OnCreate** event can be created by two ways; either by double clicking the **DefaultConsole** object or by selecting **OnCreate** in the Events section of the right click context menu of **DefaultConsole**.

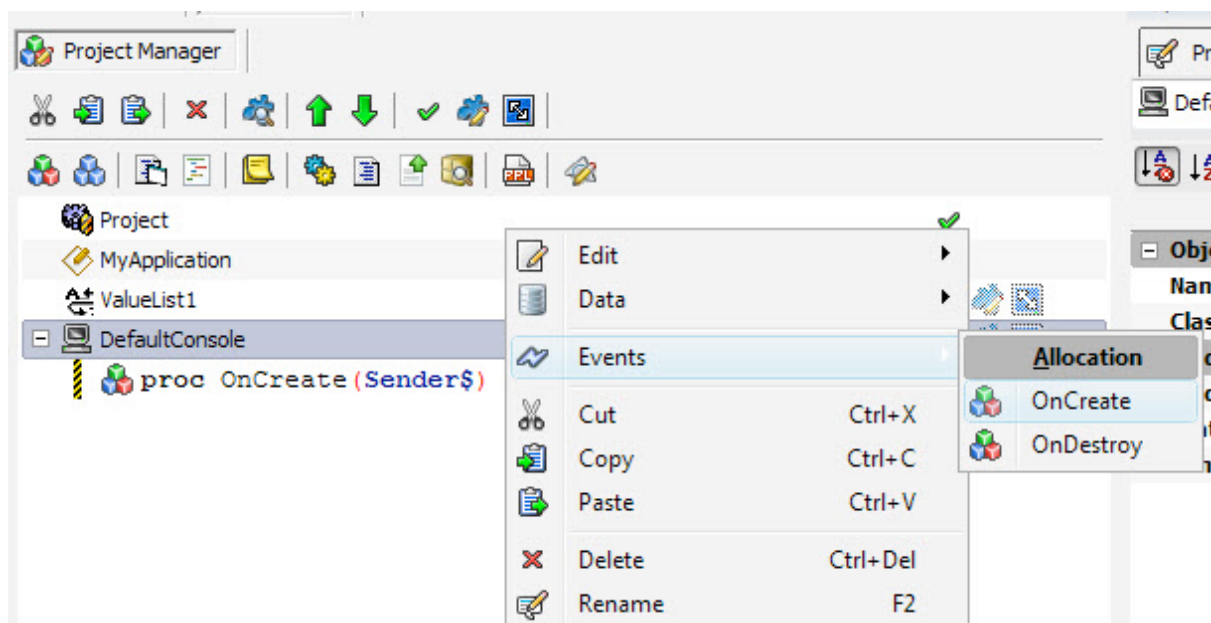


Figure 141: Create OnCreate Event

- Drop the **PValueList** object on the **OnCreate** Object. Doing this will allow us to specify actions on the creation of **DefaultConsole**.

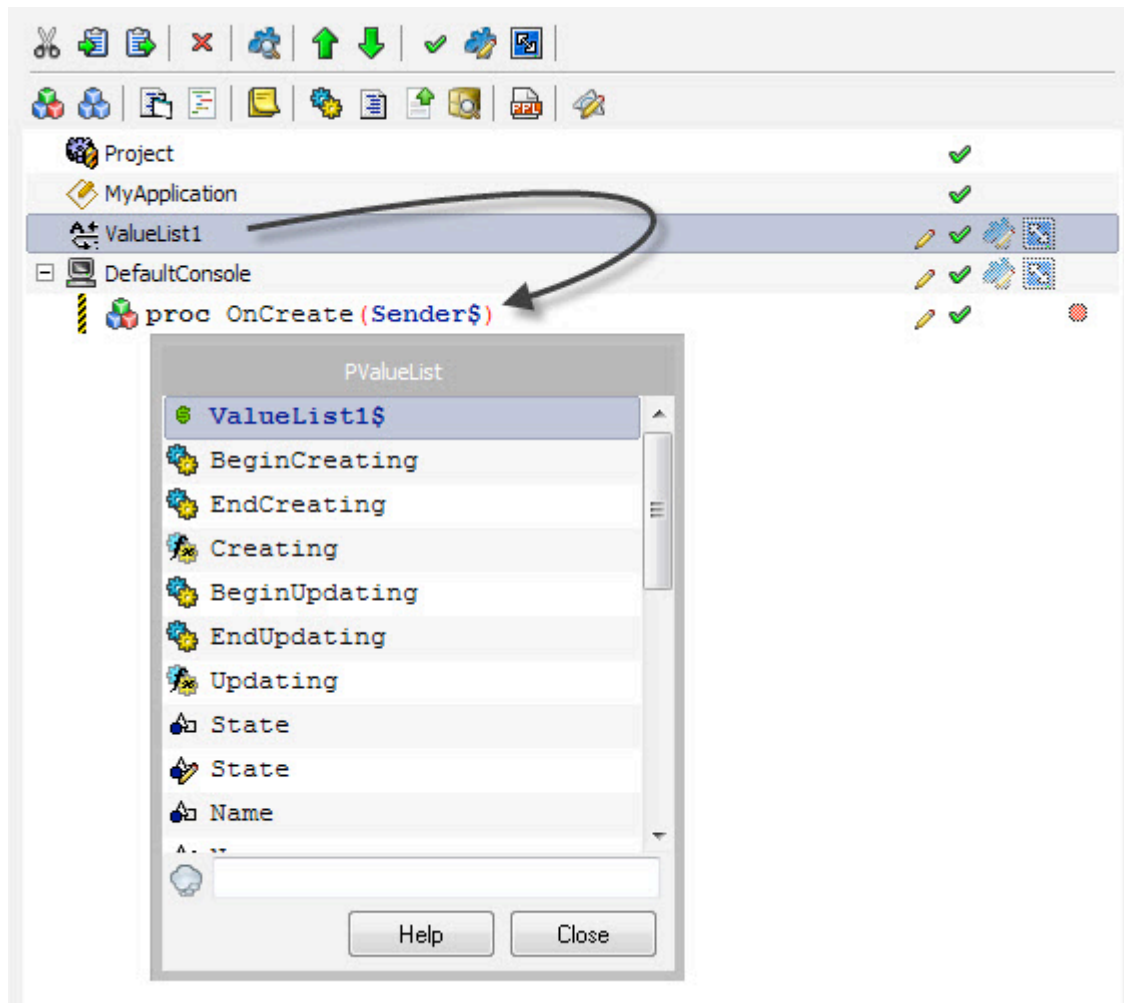


Figure 142: Code Completion window

- The Drop operation will result in auto **Code Completion** box. Select **Add** in it as we have to add values in it.

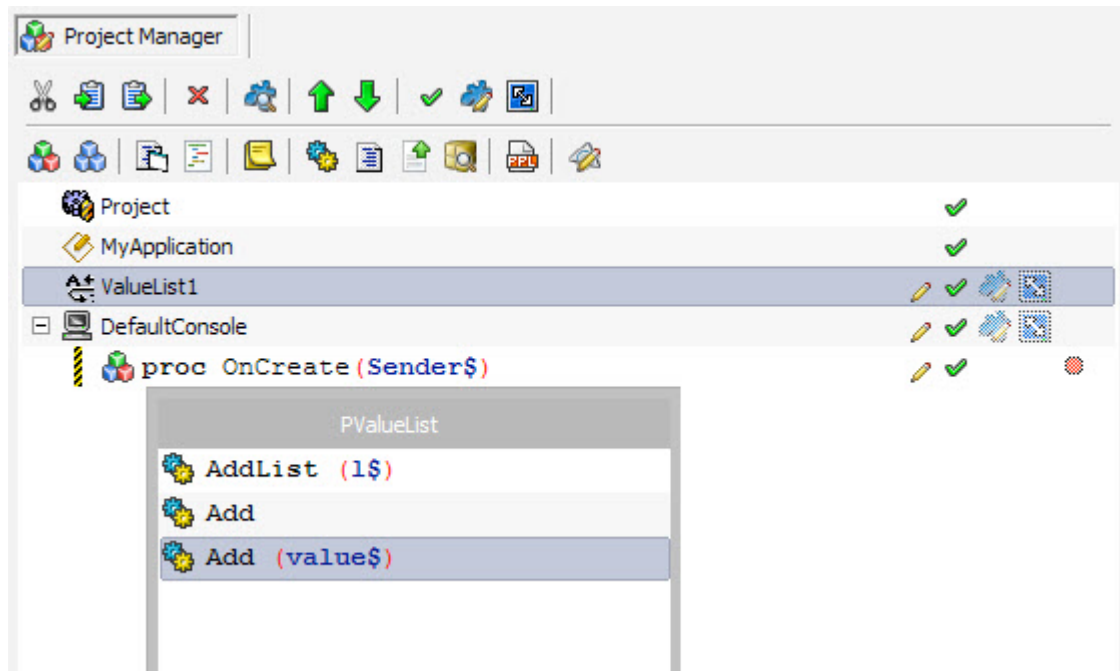


Figure 143: Select Add

- Enter the Value you want to give to your first value in the **PValueList**. To do this, select the newly created **PValueList.Add** object and change its **Value** property to the test you want it to show.

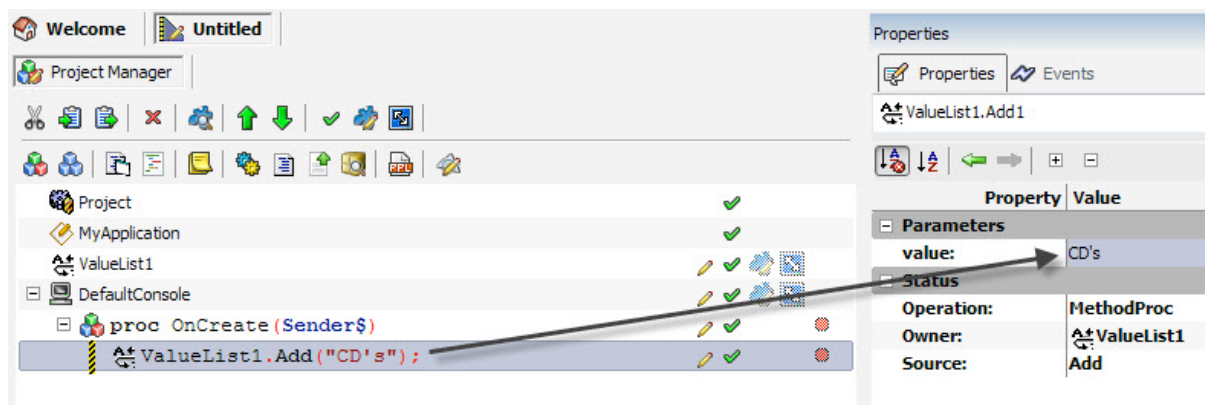


Figure 144: Change Value Property

- The above steps mark the creation of a list value in our **PValueList** object. Now we will learn how to add one more object in our list.
- To add another object in the **PValueList**, drag the **PValueList** object to the **OnCreate** event and select **Add** from the **Code Completion** window. Change the text of the **value** property to create yourself another list value.



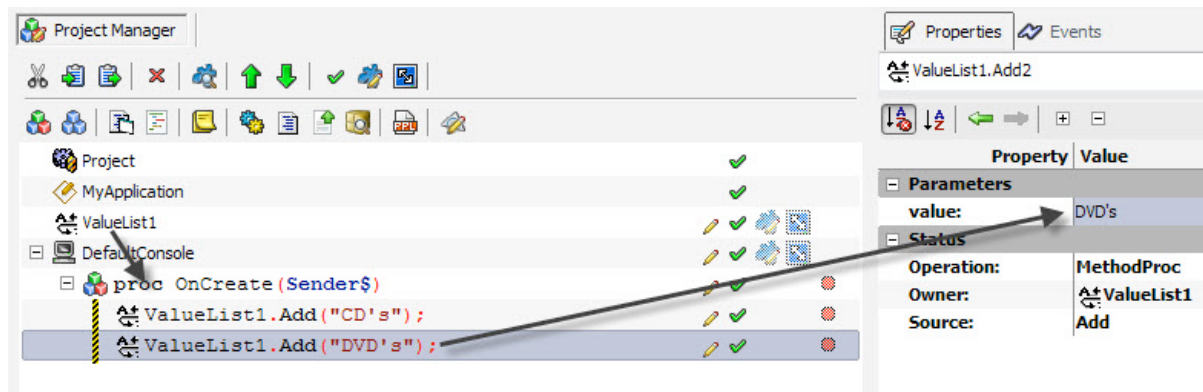


Figure 145: Change Value Property

- Again, follow the same procedure of dragging and dropping the **ValueList** object on the **OnCreate** procedure and selecting Add from the **Code Completion** window. One can create as many list values as he or she likes by following the steps given above. After all the other values have been created, we will have to make them in Uppercase with the use of **Uppercase** property.

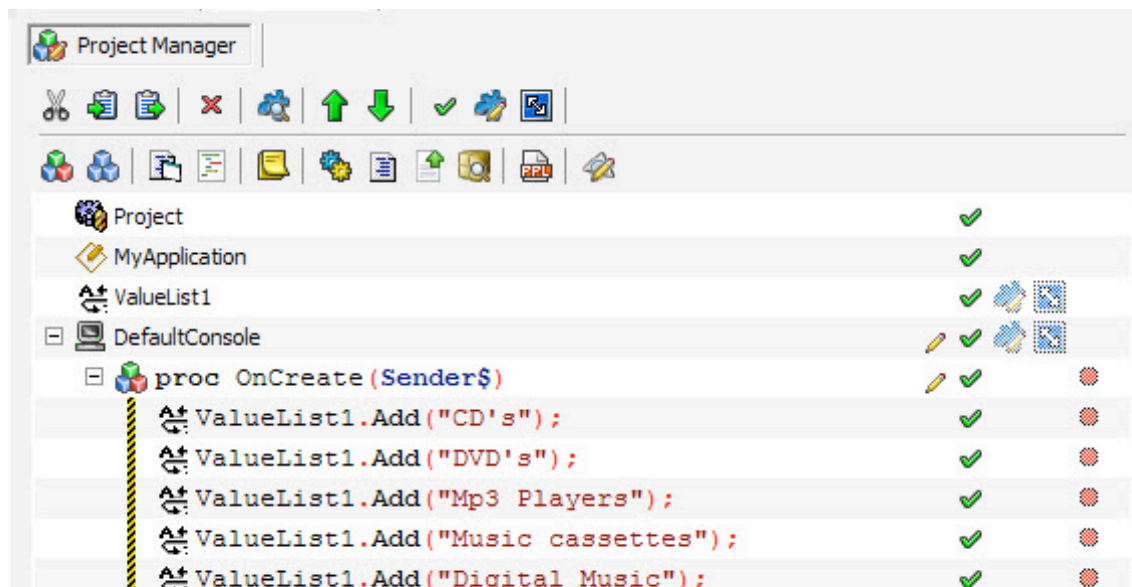


Figure 146: Project View

- A list contains various items and this for making each and every item in uppercase, we will have to convert each and every item to Uppercase. For doing this, we will have to use the **ForEach** loop. This loop will perform a specified action for each and every item in the **PValueList**. Drag the **ForEach** loop from **Components Pane** to the **Project Manager**.

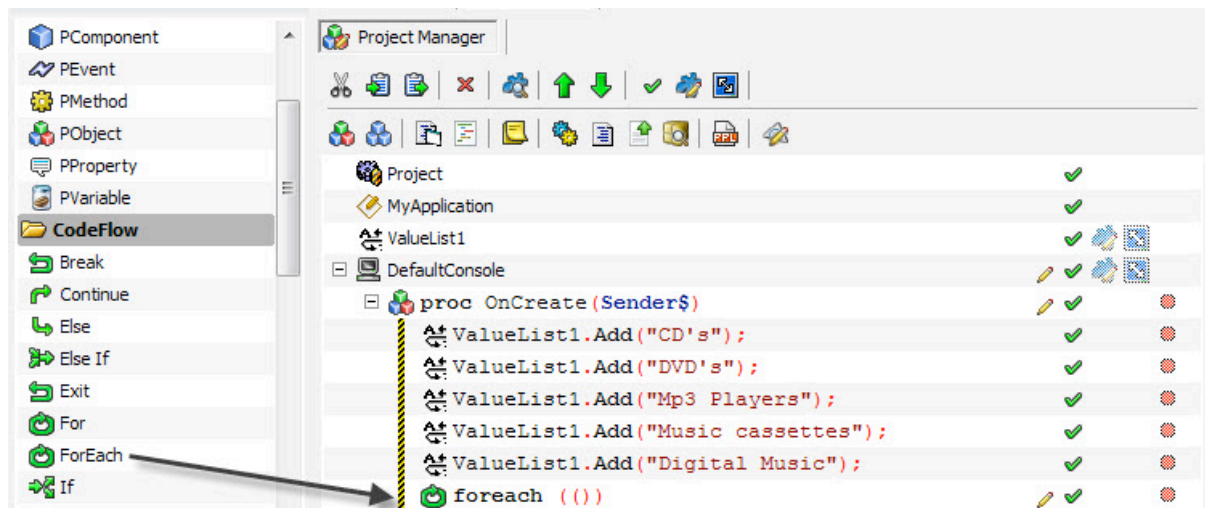


Figure 147: Drag ForEach

- We will specify the **PValueList** list to the **ForEach** loop through the **Expr** property. Drag the **PValueList** object to the **Expr** property.

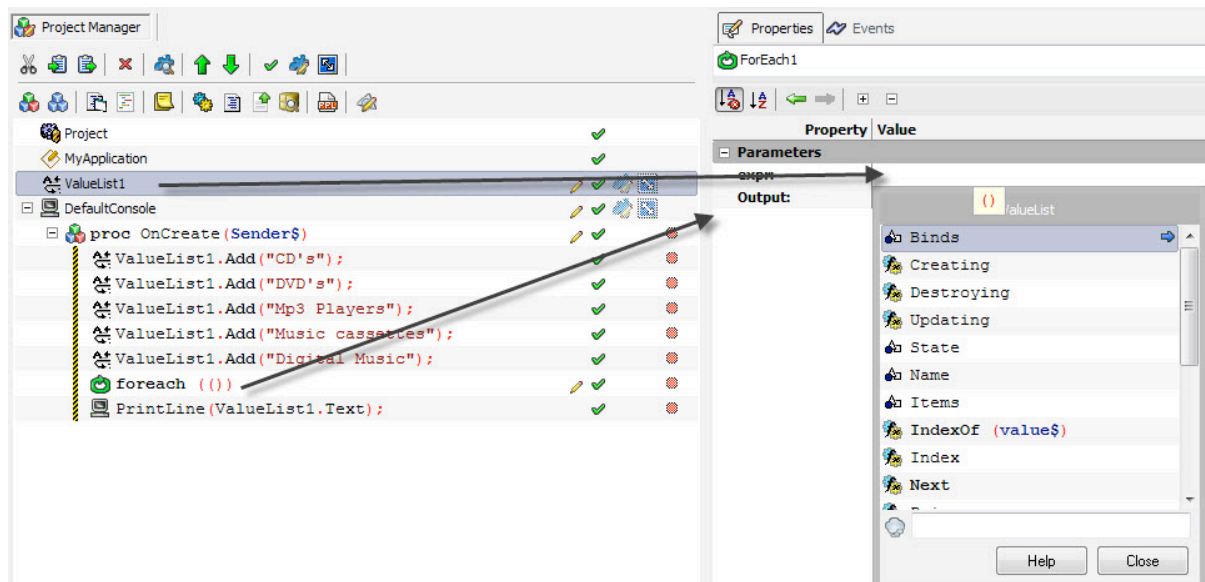


Figure 148: Drag PValueList

- In the **Code complete** window, select Items.

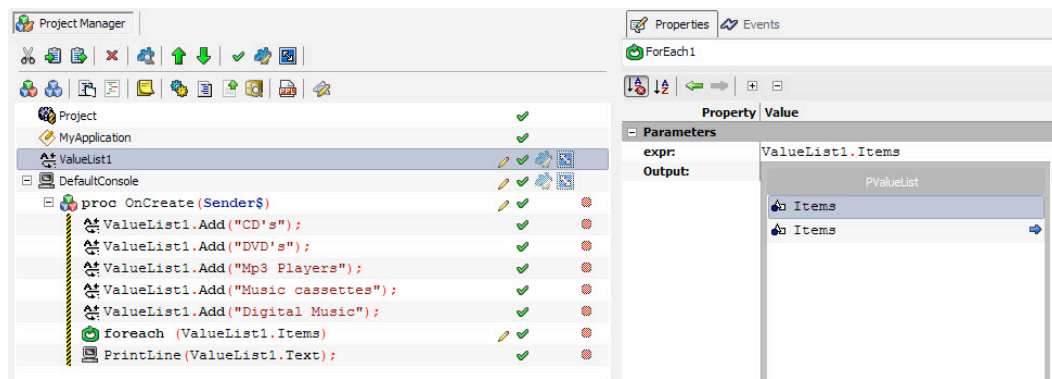


Figure 149: Select Items

- Drop the **PValueList** object to the **ForEach** loop. This will bring a **Code Completion** window.

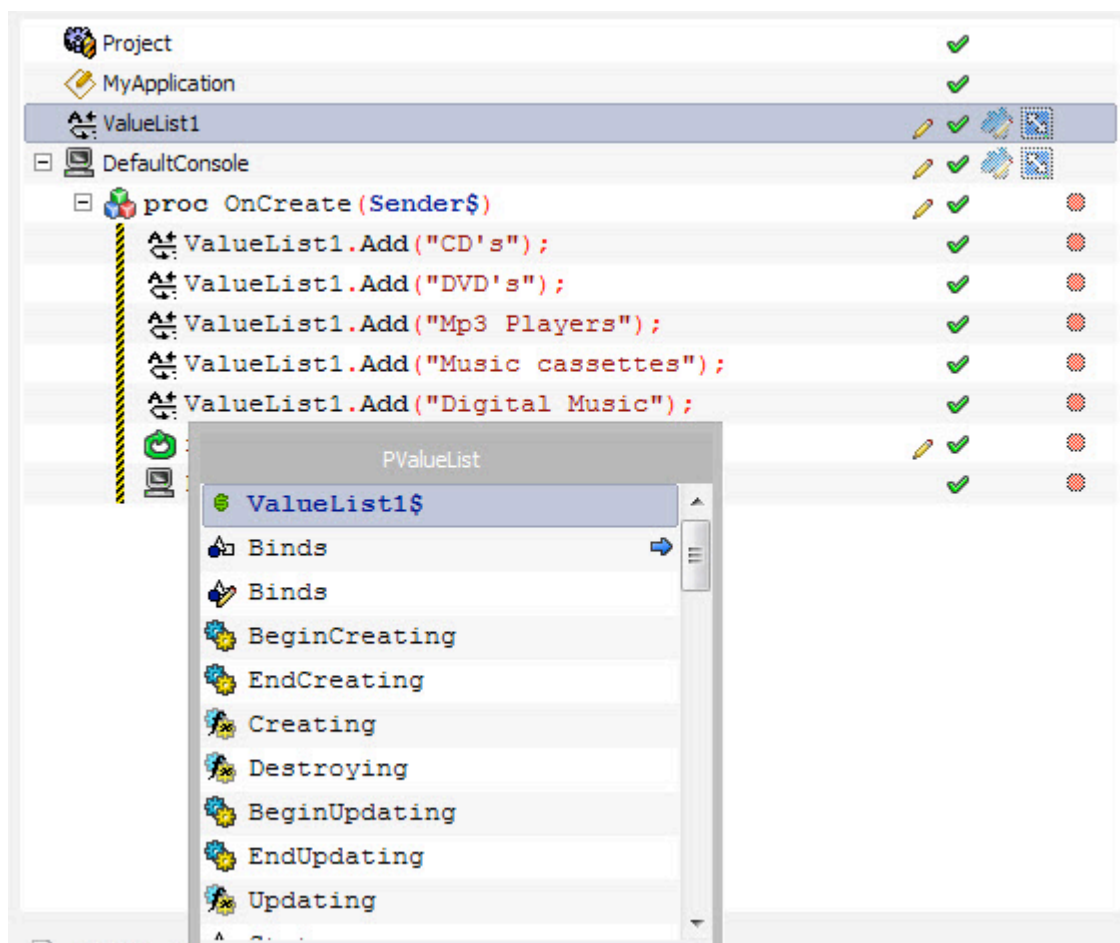


Figure 150: Code completion window

- Scroll to **Items** property from the list and click on the green arrow pointing towards right. This will again bring a **Code Completion** window.

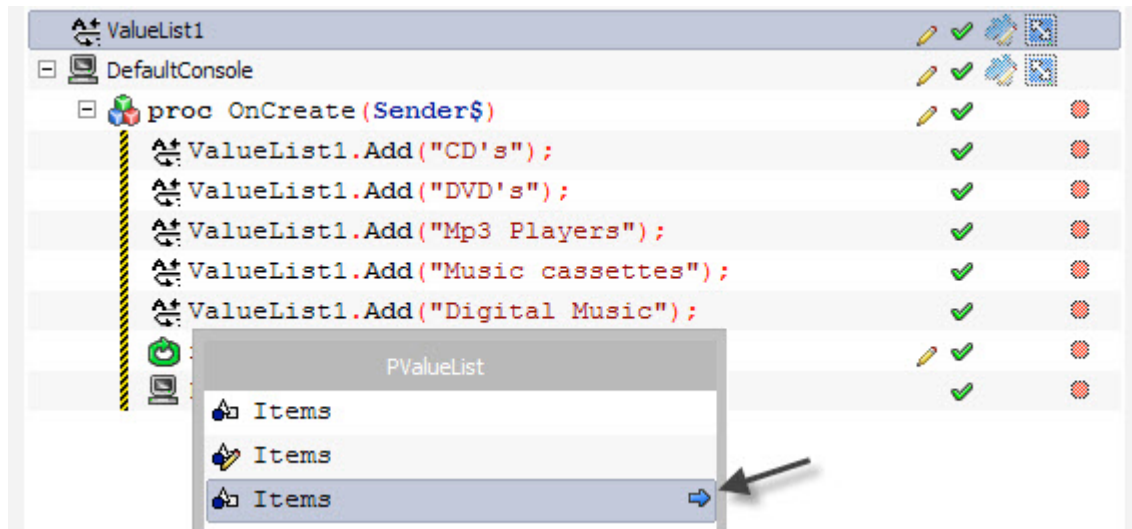


Figure 151: Select Items

- Select the **Uppercase** property in the new **Code Completion** window.

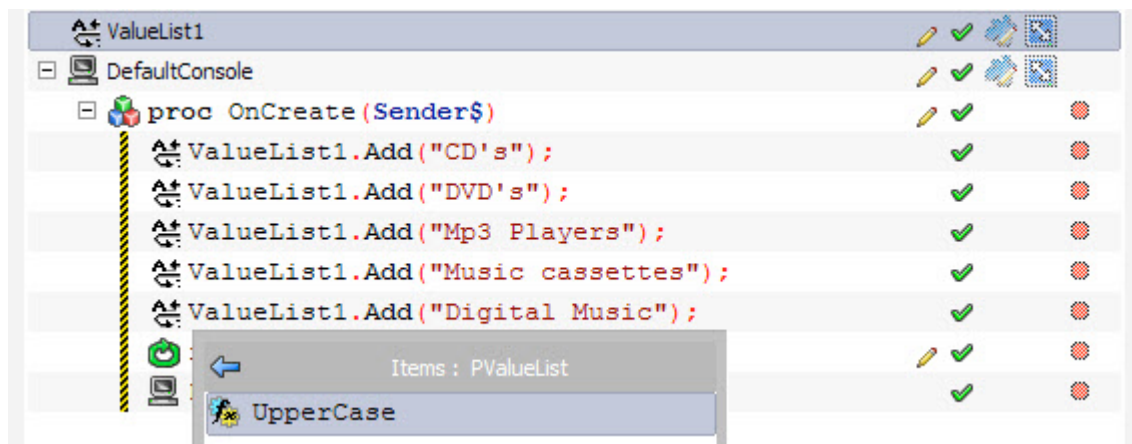


Figure 152: Select Uppercase

- Now we will have to display our **PValueList** values in the screen. For printing the values, we will use the help of **PrintLine** command.
- Click **OnCreate** event and press **Ctrl+Space** bar to bring **Code Completion** window.

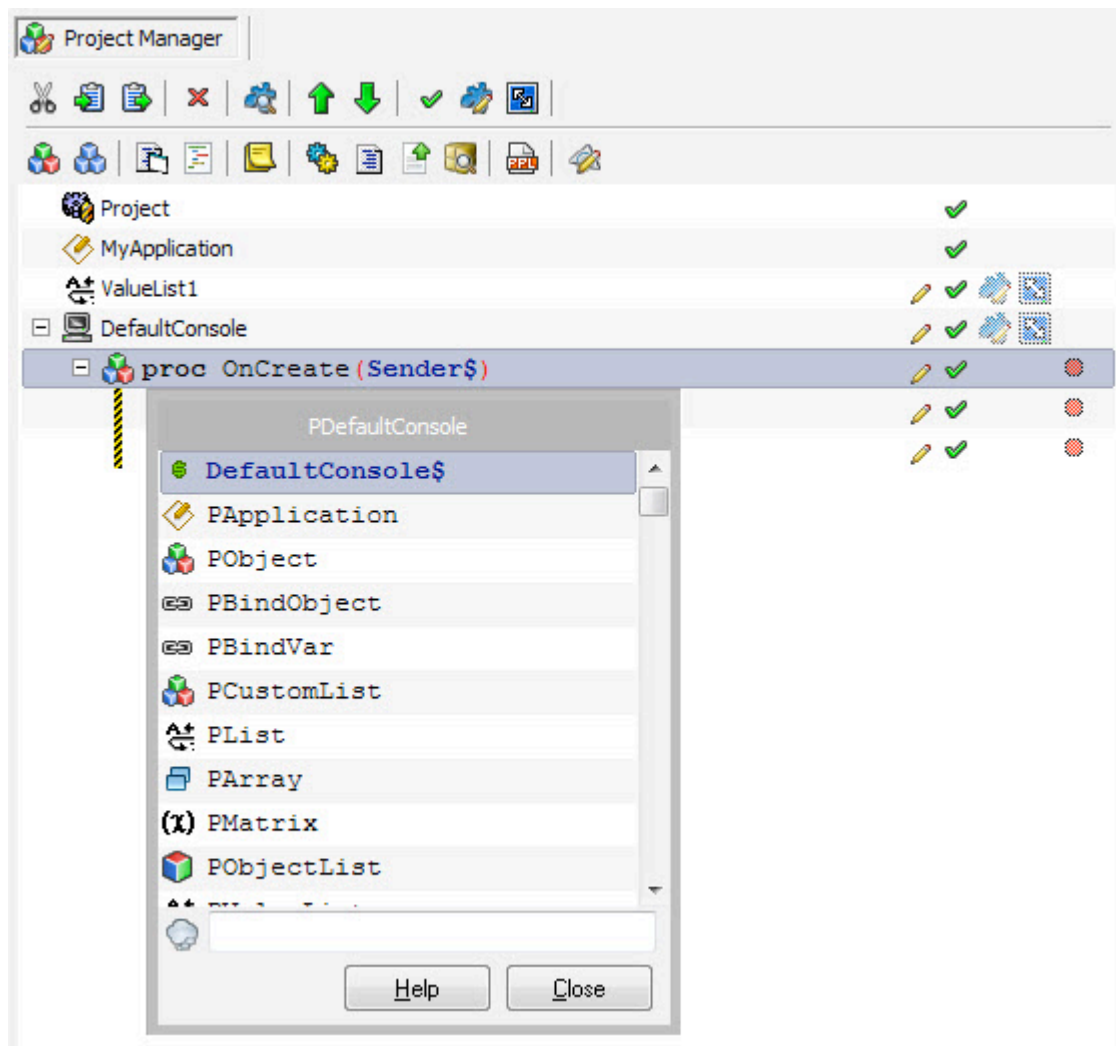


Figure 153: Code Complete Window

- Select **PrintLine** object and drag the **PValueList** object to the **Value** property of **PrintLine** object.



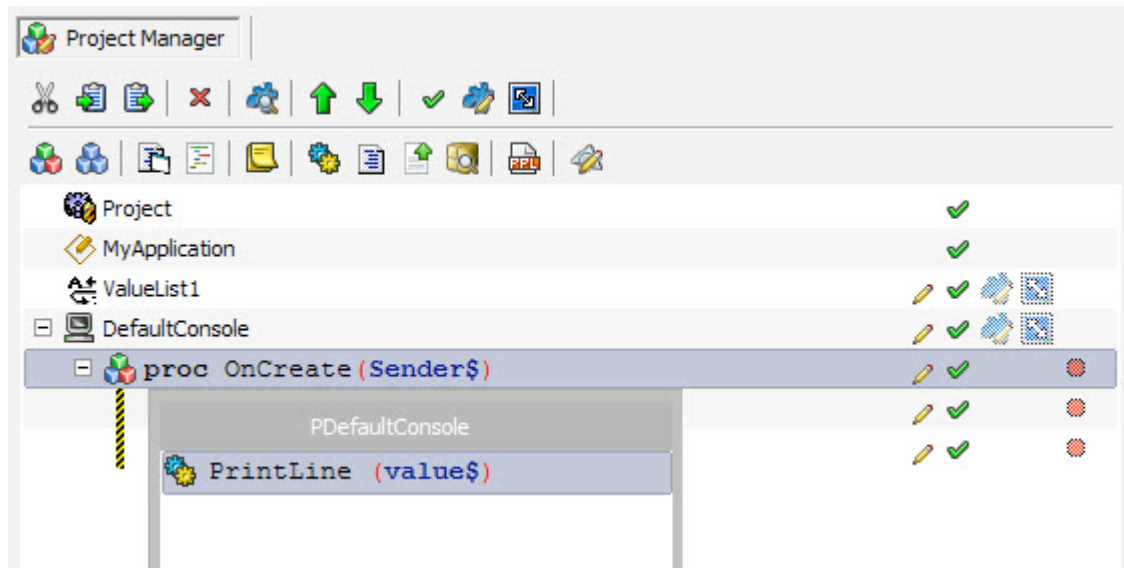


Figure 154: Select PrintLine

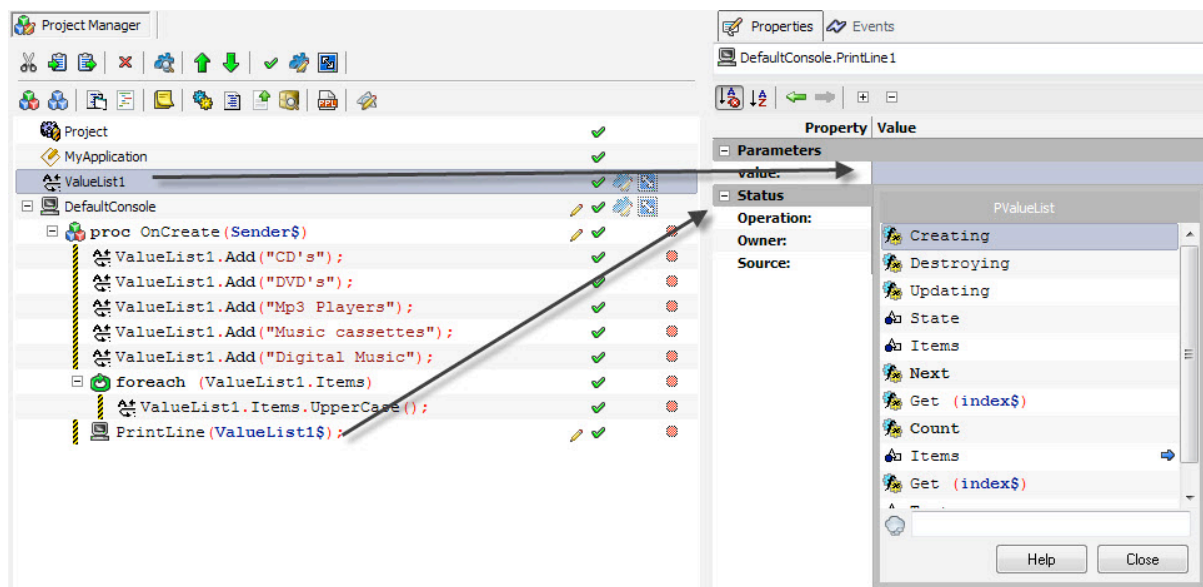


Figure 155: Change Value property

- The drag operation above will result in a **Code Completion** window; select the **Text** command in it. Doing this will pass the text of **PValueList** to the **PrintLine** object which will then print it on the screen.

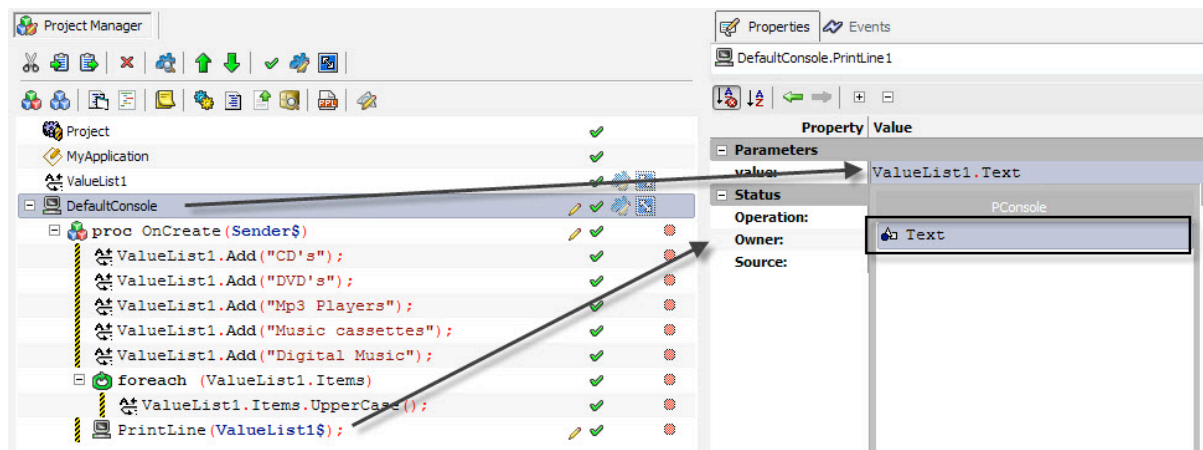


Figure 156 Change the Value property

- Save your project by pressing **Ctrl+S** or go to the **File Menu>Save**. Give an appropriate name in the **Save As..** window and press **Save** to save the file.

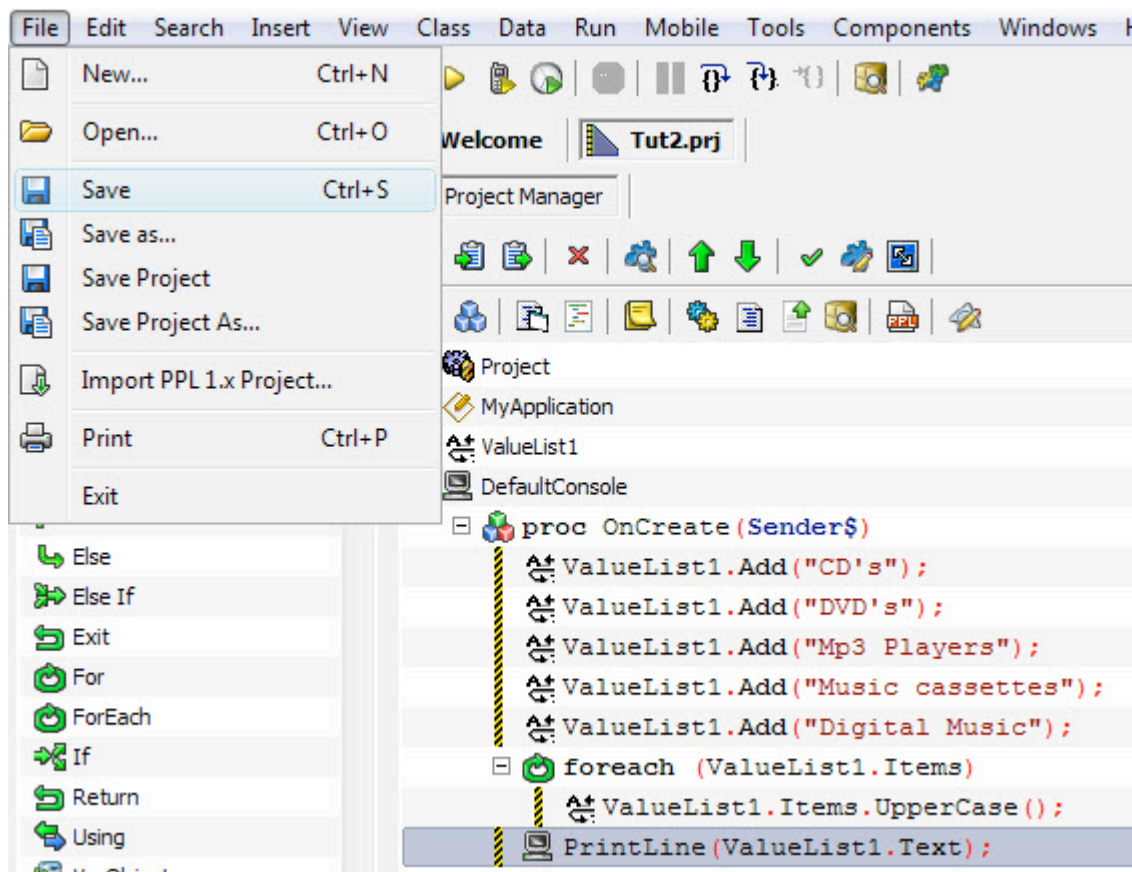
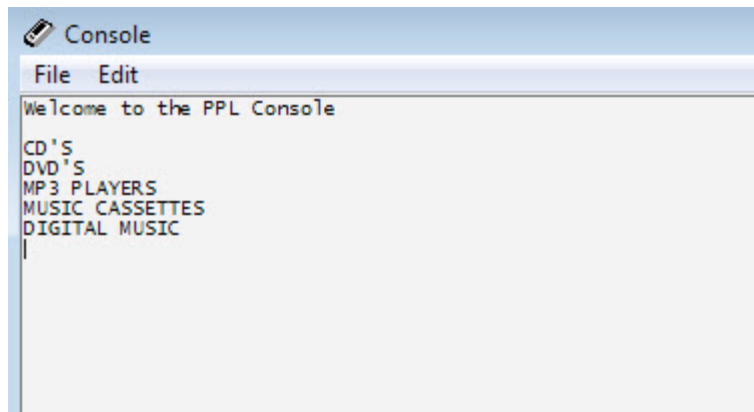


Figure 157: Save project

- After saving the project, press **F5** or run the project to see the output.





**Figure 158: Output**

**Note:** By placing the **PValueLise** before **DefaultConsole**, we are assuring that **PValueList** is created before **DefaultConsole** when the project compilation happens.

## Using PObject

Being a base class of all the objects in PCI (PPL Control Library), **PObject** is an important object but its use is more or less restricted to being a base class for other objects only.

**PObject** can also be used to create a custom class component **PObject** as a basis. Given below is an example that will allow you to create a custom class that will have Username and Password as properties and will also have VerifyPassword as a method.

- Create a new component project. For doing this, click on the **File>New** menu item or press **Ctrl+N** on the keyboard to bring the **Select New Project Type...** window. Now select **Component Project** in this window and press **Ok**.

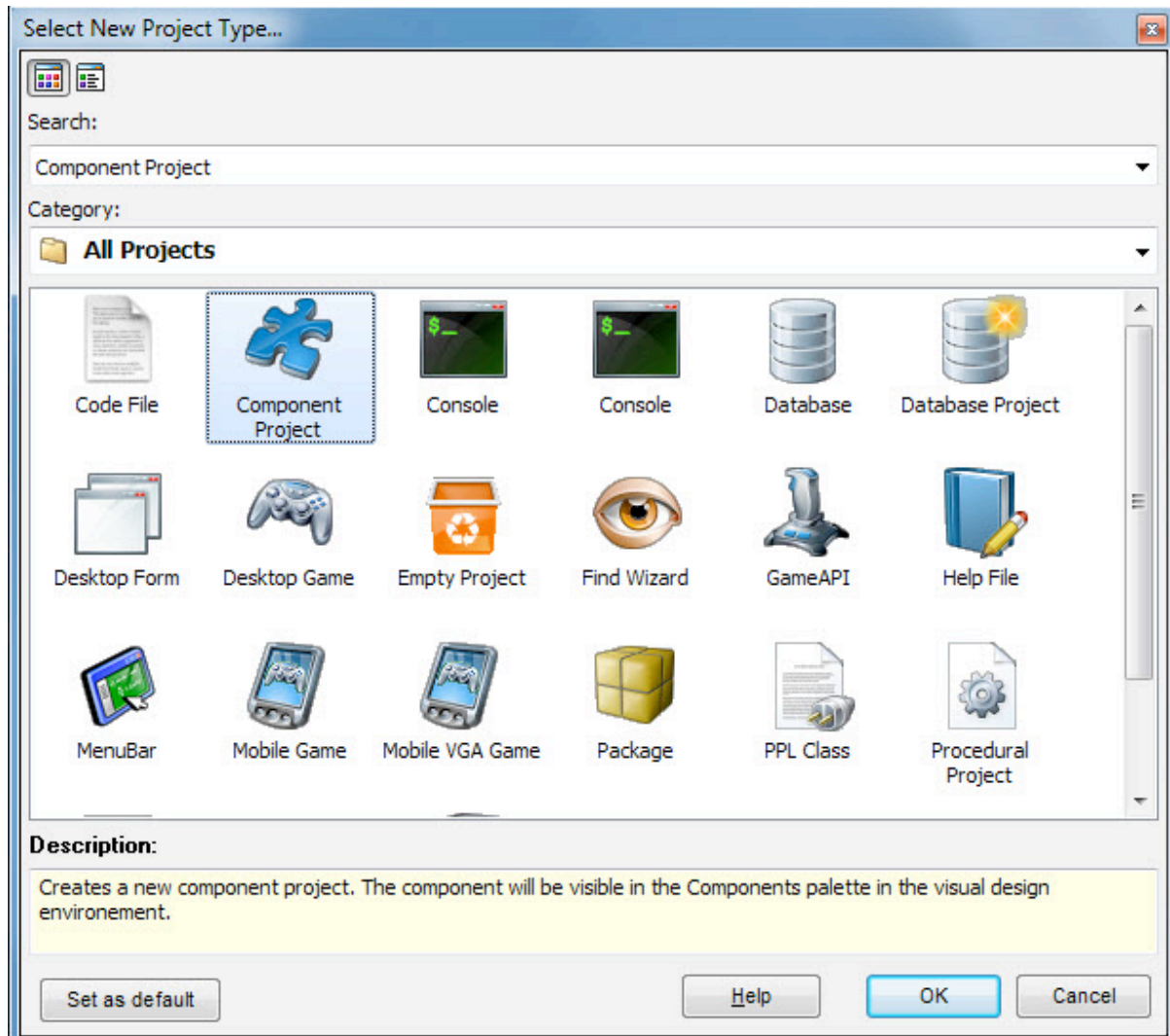


Figure 159: Create new project

- Delete the **MyComponent** object as we need to create our own class.

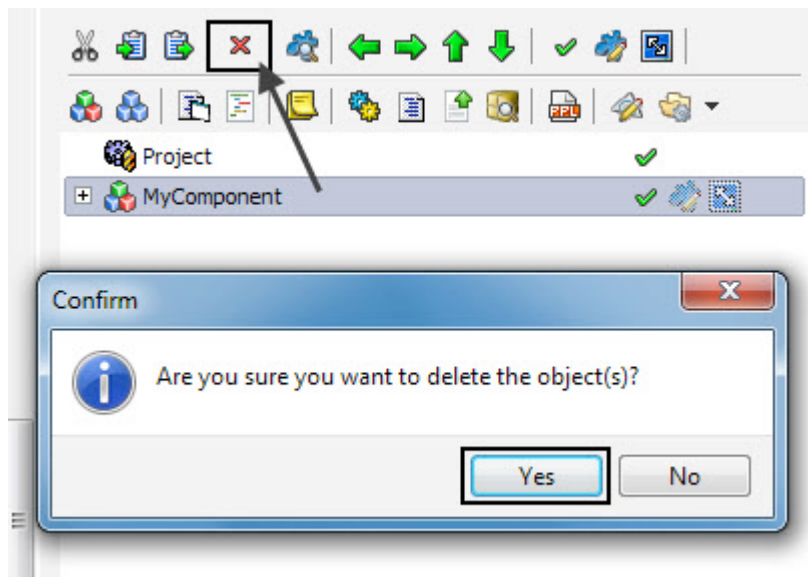


Figure 160: Delete Object

- Now drag a **PObject** object to the **Project Manager** and name it MyClass. For doing this press **F2** on the keyboard while the **PObject** is still selected and change the name of the object.

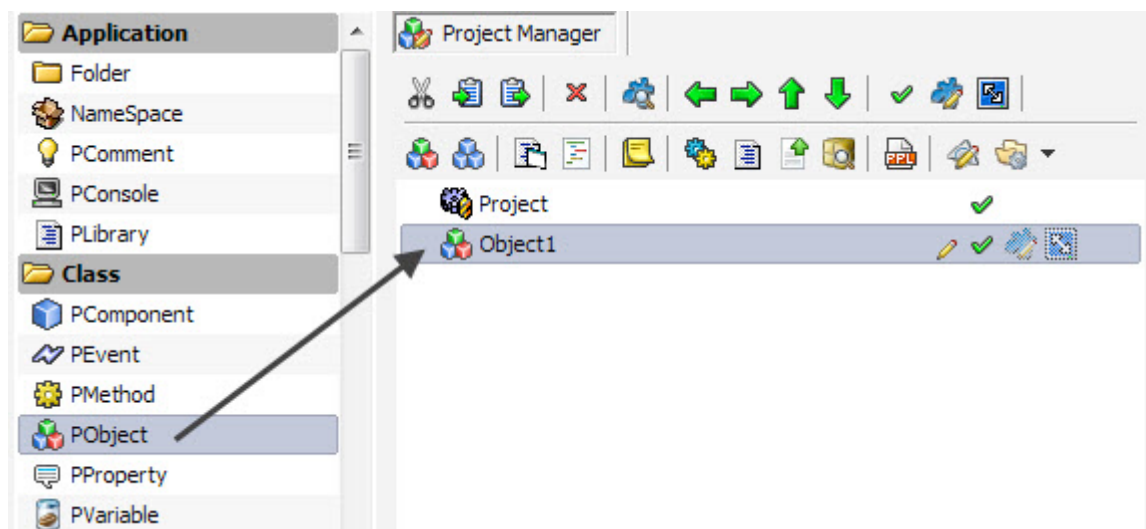


Figure 161: Drag PObject

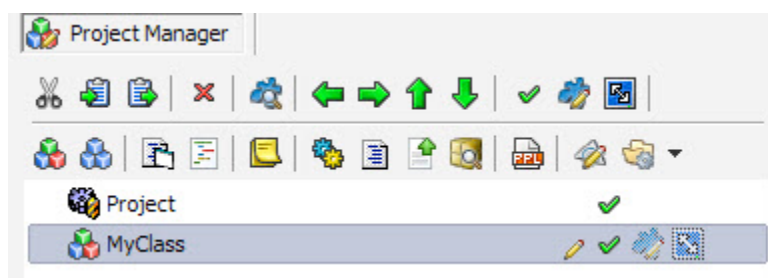


Figure 162: Rename

- Now drag two **PProperty** objects to the **MyClass** object so that they become its child objects.

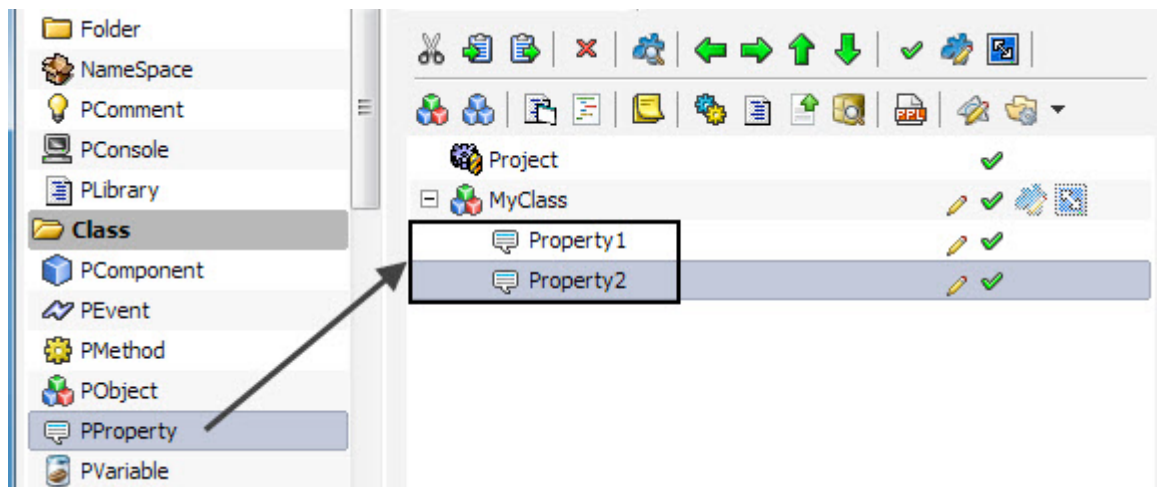


Figure 163: Drag PProperty to Project Manager

- Rename the first **PProperty** object to Username and the other to Password. Press **F2** on the keyboard to do so.

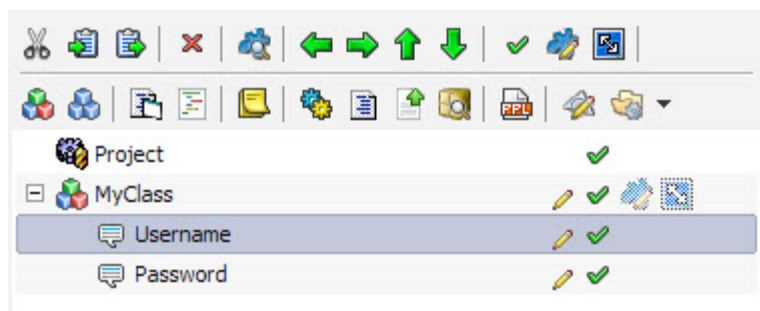


Figure 164: Change PProperty Name

- After renaming both the **PProperty** objects, drag a **PMethod** object to the **Project Manager** and rename it as VerifyPassword.

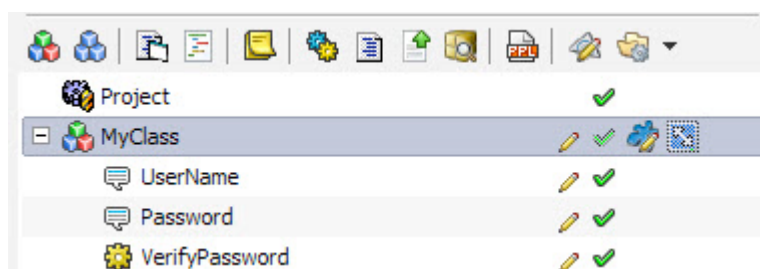


Figure 165: Drag PMethod and rename it

- After all the components of the class are declared, click on the **Class** menu item and select **Create Class**.

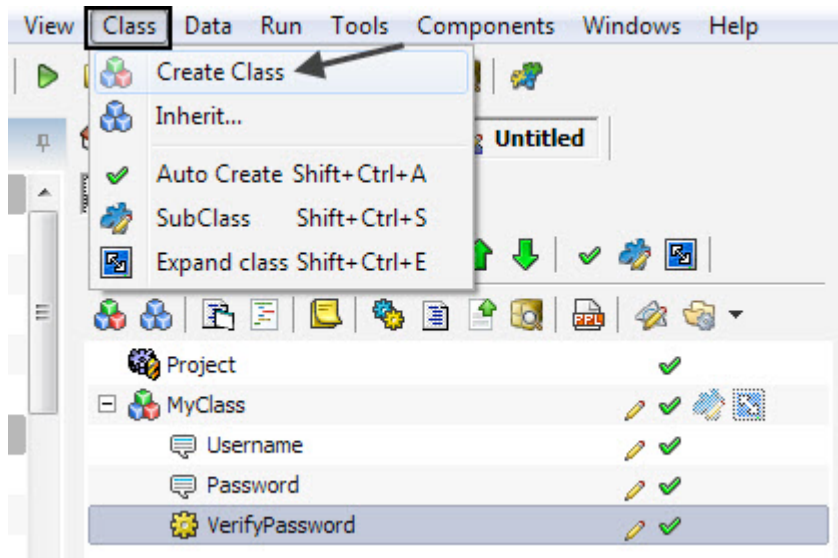


Figure 166: Create Class

- Clicking on the **Create Class** menu item will create a **MyClass** custom object in the **Components Panel** that can be used in the project. This component can be used in your code and will automatically have username as well as password as its property. The VerifyPassword method can be used to verify the password entered.

## Creating A Game Visually

PIDE can be used to create a game visually. Creating a game without using any coding means that you will be able to create as well as run the game just by using drag and drop features of PIDE. Given below are the steps used to create a game visually through PIDE:

- Start by creating a new project and select a **Desktop Game** project in the **Select New Project Type..**

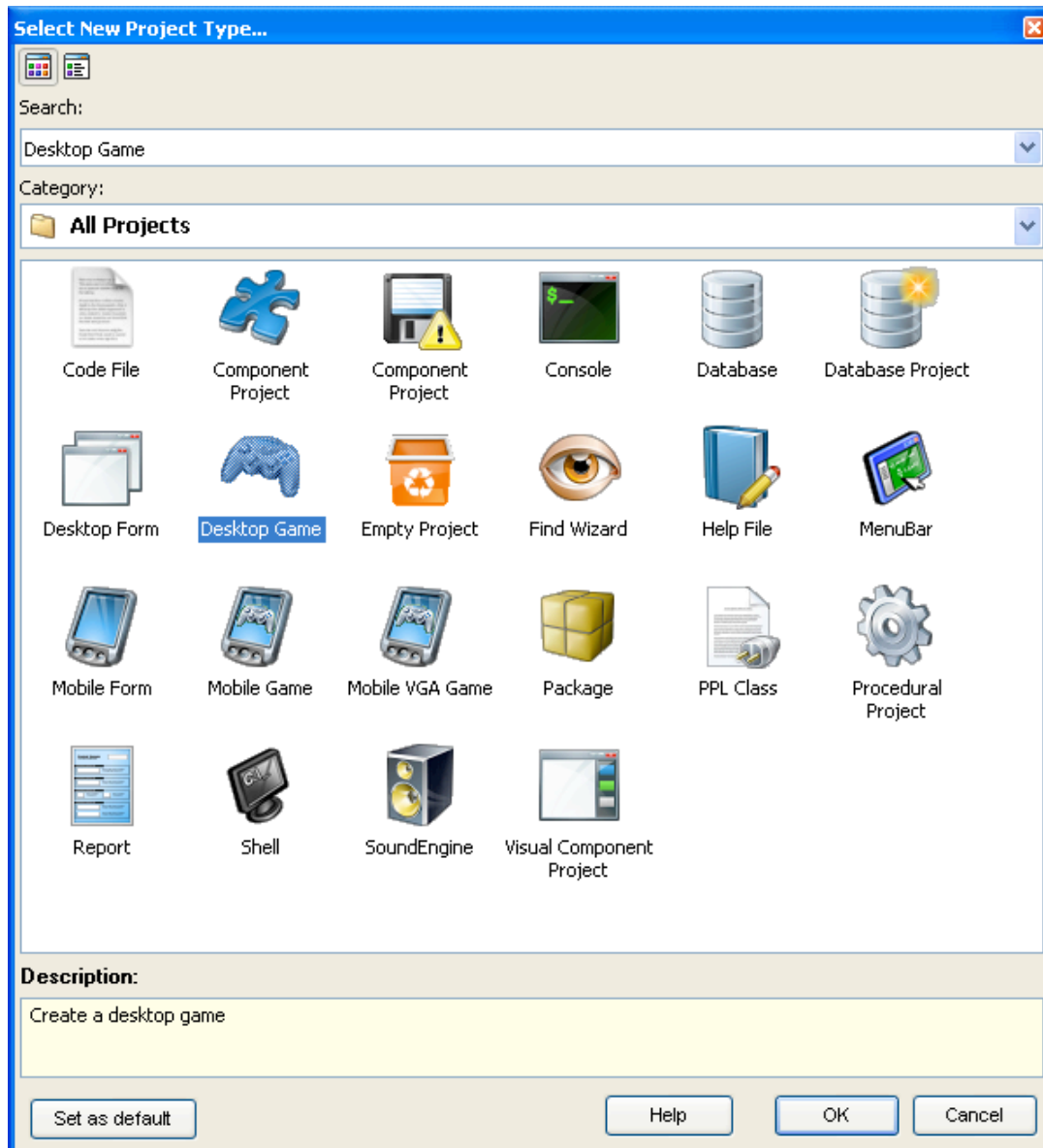


Figure 167: Create New project

- Once a new **Desktop Game** project is created, add a **PSurface** object to it by dragging and dropping the **PSurface** object to the surfaces folder. A **PSurface** is an object that contains a bitmap that is attached to a sprite.

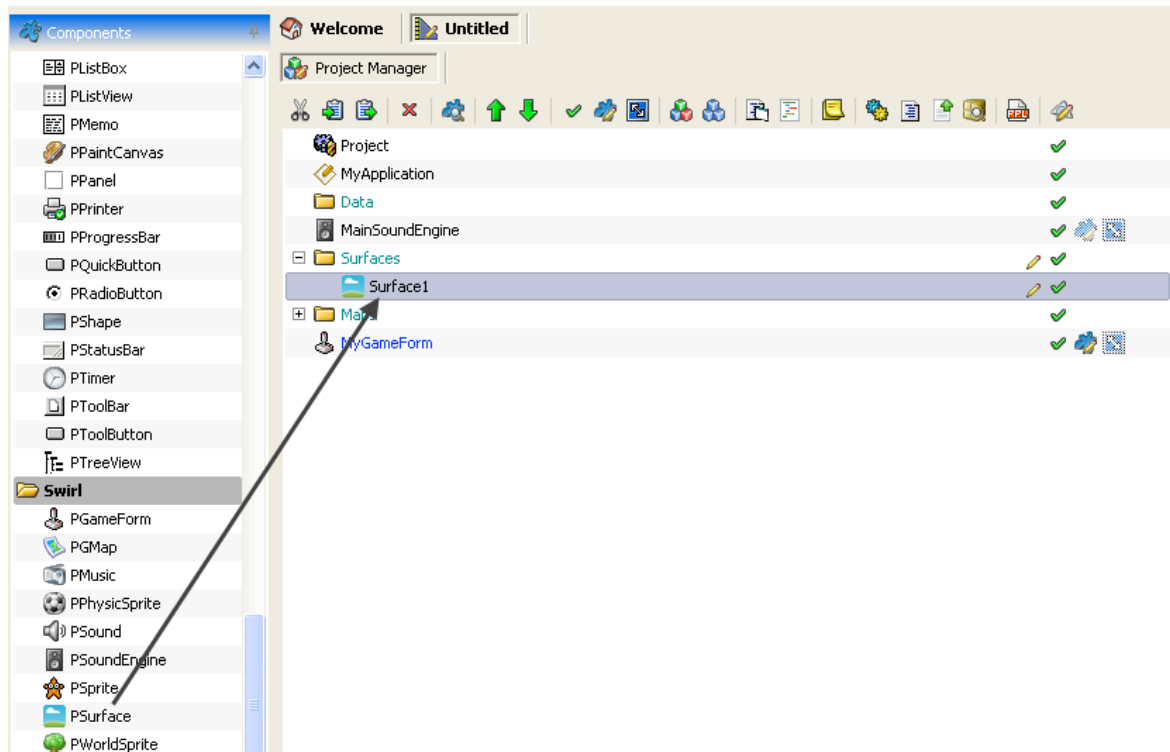


Figure 168: Drag Psurface

- Double click the **PSurface** object and choose an image from your computer. Once you have chosen image is included in the bitmap, close the **surface editor**.

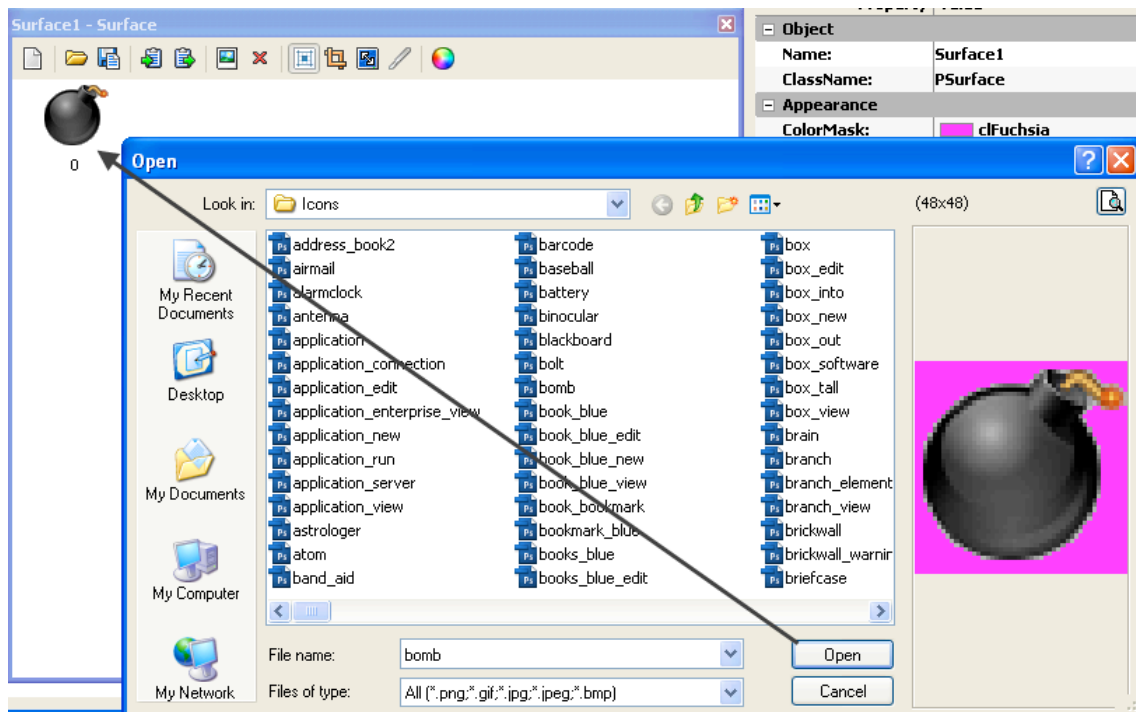


Figure 169: Surface Editor

- Expand the maps folder and click the **PGameMap** Object to open the **Game Map**.



- Create a **Sprite** object by double clicking the **PSprite** object on the **Game Map**. A **Sprite** is used to display the surface has a lot of attributes that can be used to manipulate how objects in game behave.

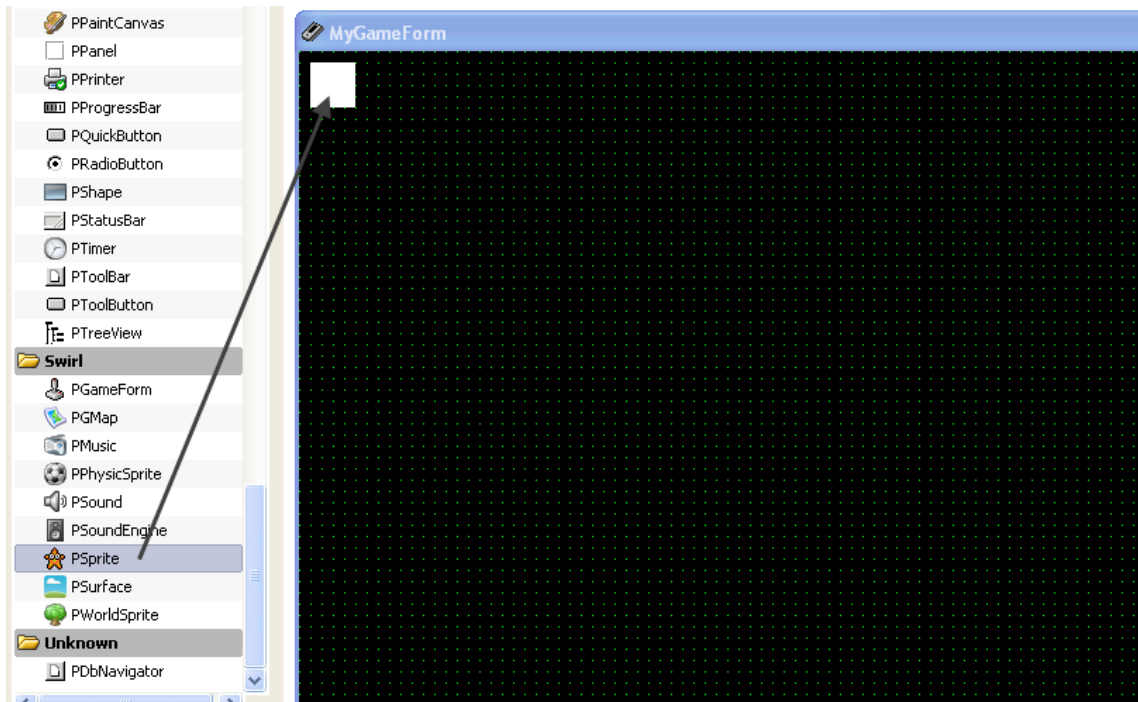


Figure 170: Drag PSprite

- While the **Sprite** object is still selected, go to its properties and specify the surface you created earlier in the **Surface** property.

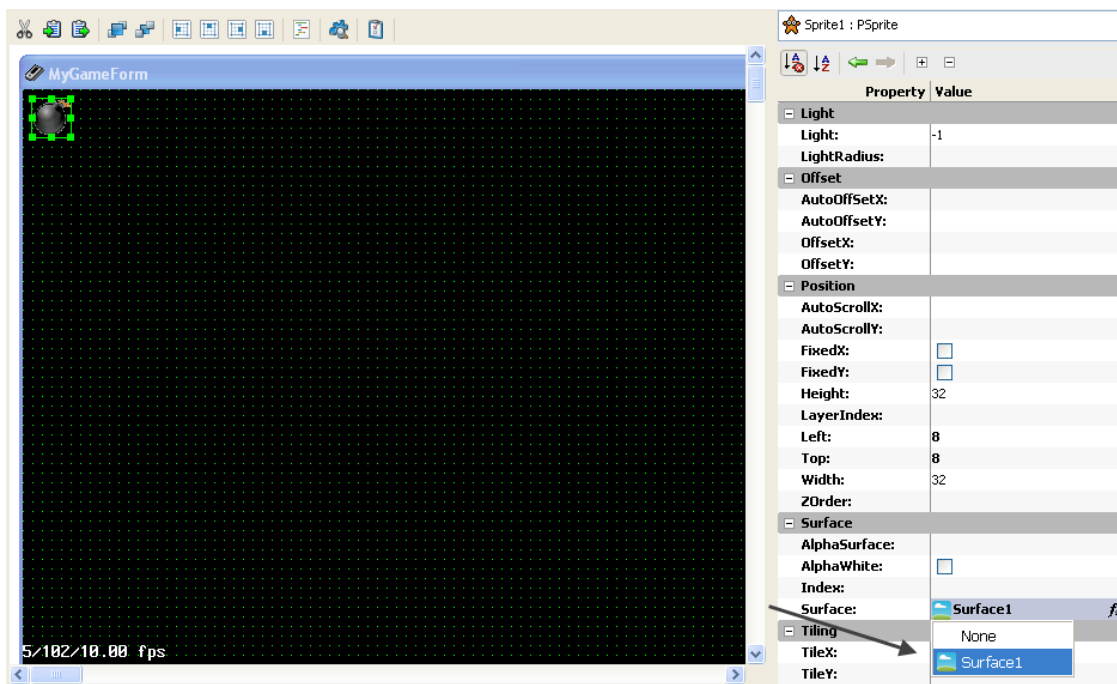


Figure 171: Select Surface Property

- Now we will create an event that would allow us to perform actions according to mouse movements. For implementing a move movement event, right click on the **GameForm** object and select **OnMouseMove** event from the **Events** tab.

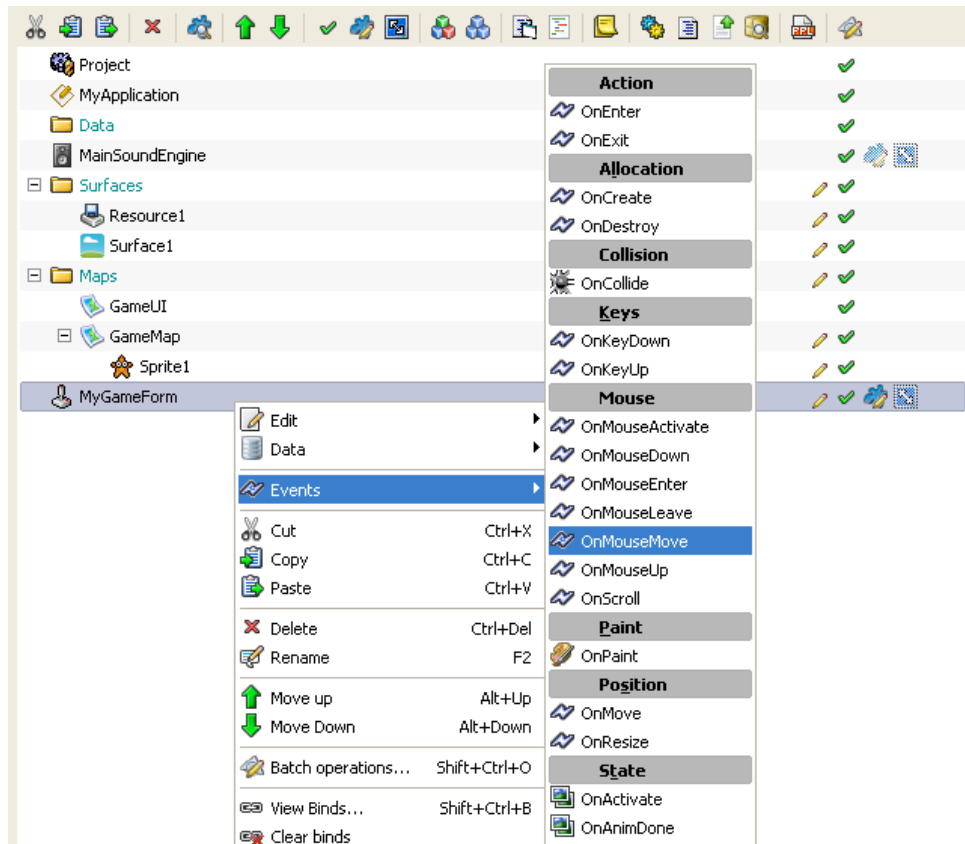


Figure 172: The events tab

- Drag the **PSprite** object present in your project and drop it on the newly created event. This will open the **auto insert** box that would allow you to choose a method from the list. Here you can write “**Move**” to choose the **Move** method.

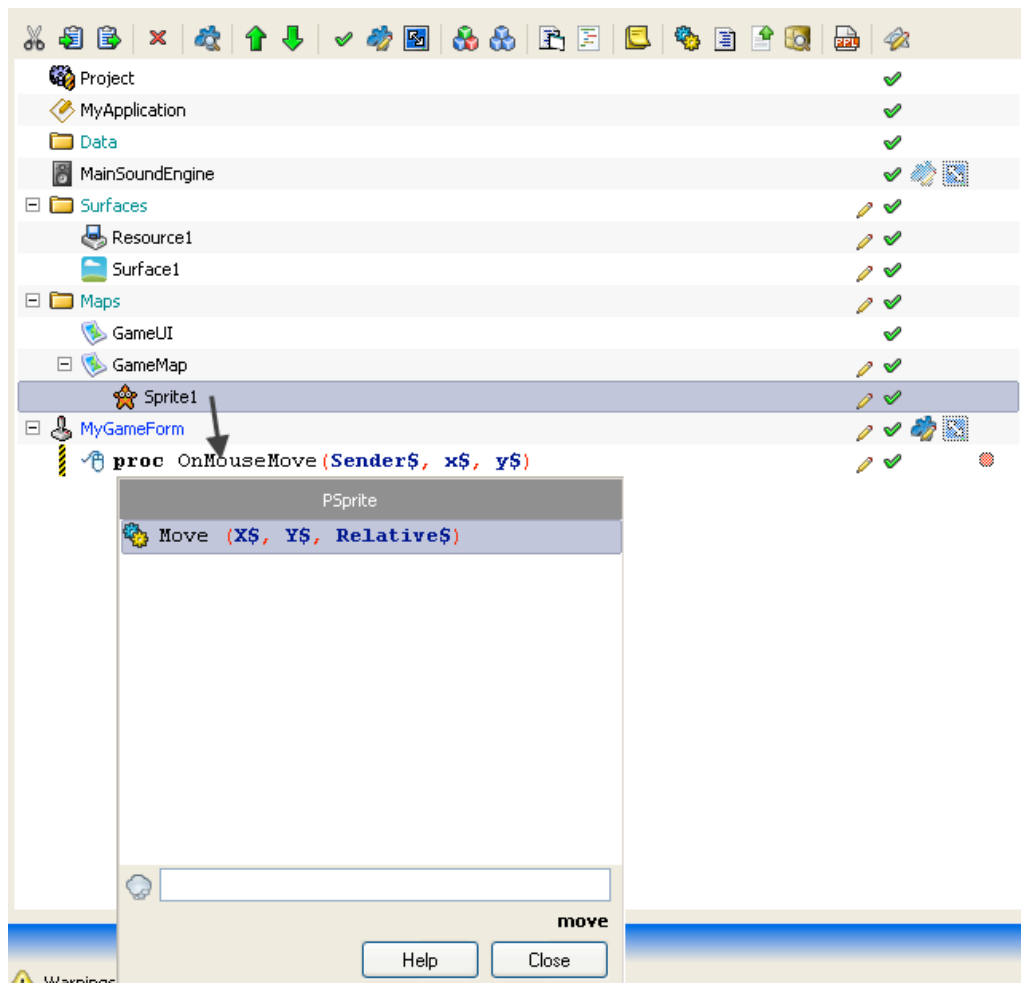


Figure 173: Code complete

- After the method is created, click on it and change its X as well as Y property to point to \$x and \$y.

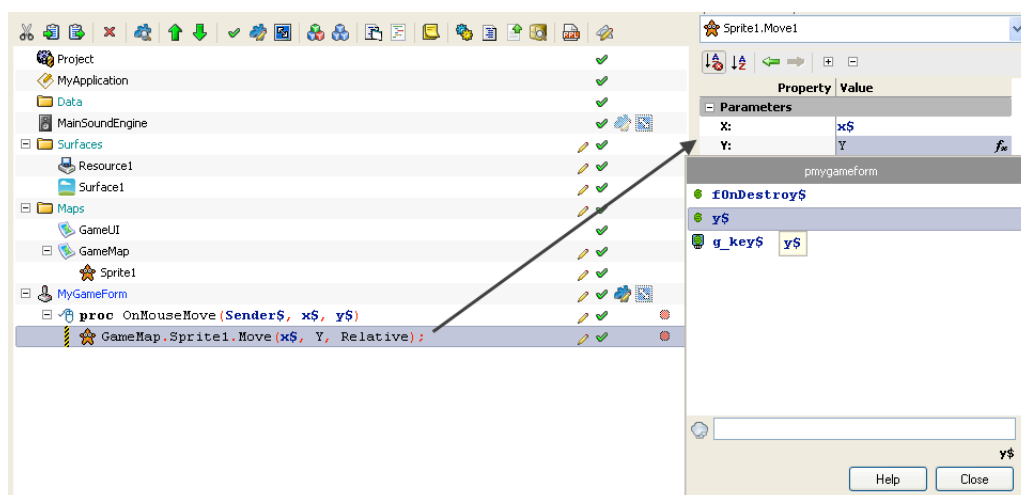


Figure 174: Change Property

- Once you have set the \$x and \$y variables, run the project to see your game in action.

## The Subtleties Of The Visual Programming.

### Loops and Conditions: For Loop

Having loops or iterations is very important of any programming language. Loops not only allow a programmer to perform a complex recursive task in an efficient manner, it also allows him/her to use other parts of programming in an efficient manner to result in a highly complex application.

Given below is an example that will allow you to process a set of instructions to a pre-set number of times if a condition is deemed true. In the given example, we will to print the factorial of a number entered by the user.

- Start by creating a new **Desktop Form** project. To do so, start PIDE and press **Ctrl+N** and select **Desktop Form Project** from the “**New Project Type..**” window. Alternatively, you can also go to **File> New** and select **Desktop Form** from the “**New Project Type..**” window.

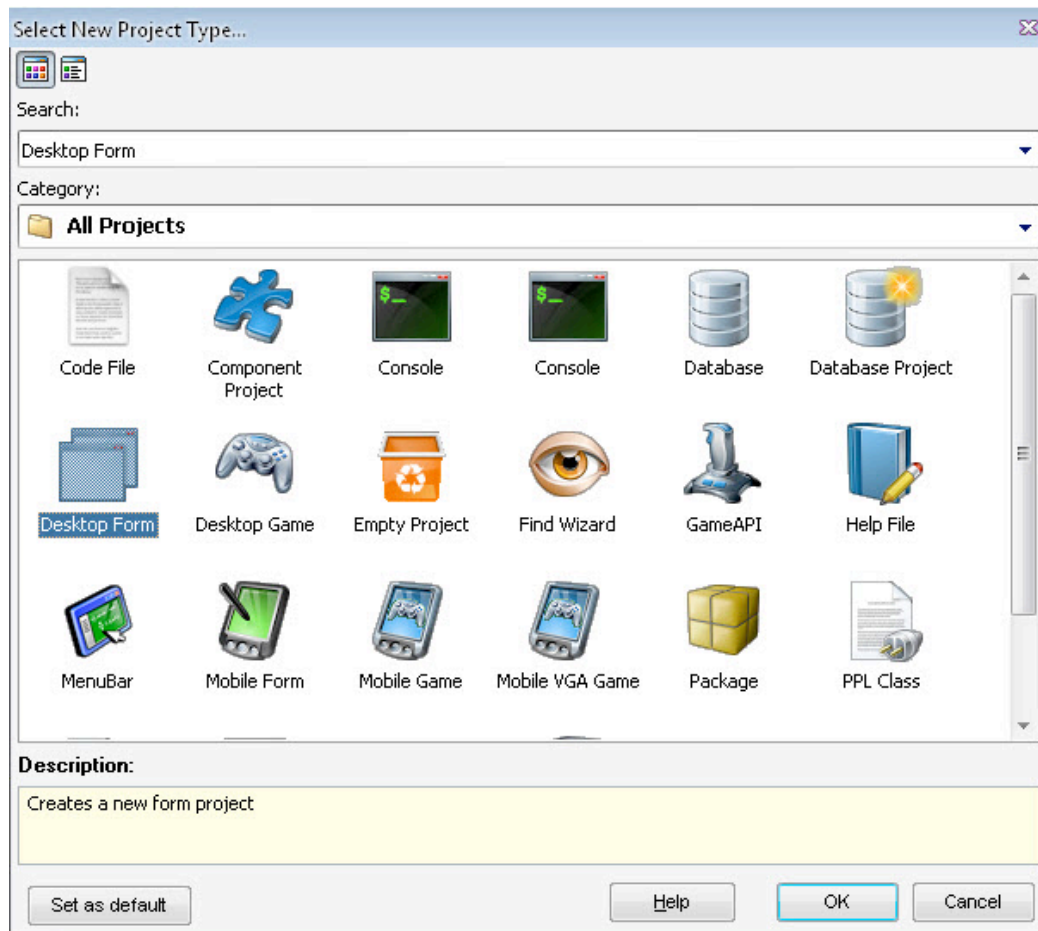


Figure 175: Create New Project

- In the project, double click the **Default Form** object to enter the **Form editor**.

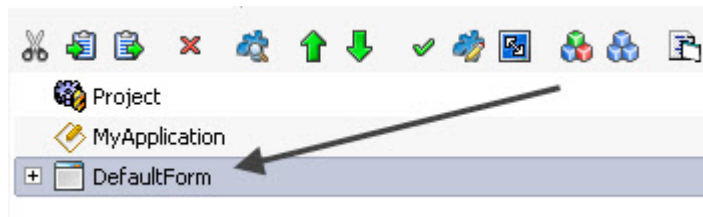


Figure 176: Double click

- In the **Default Form** Editor, we will place one **PEdit** box along with one **PButton**. While the **PEdit** boxes will contain the number we want to use for finding the factorial, **PButton** will be used for the initiation for all the actions we will have on the number. On the **Components** Pane, look for **PEdit** object and click on it; then, click on the form to place the **PEdit** object on the form. After placing a **PButton** just like other **PEdit** object, you are ready to start visual programming.

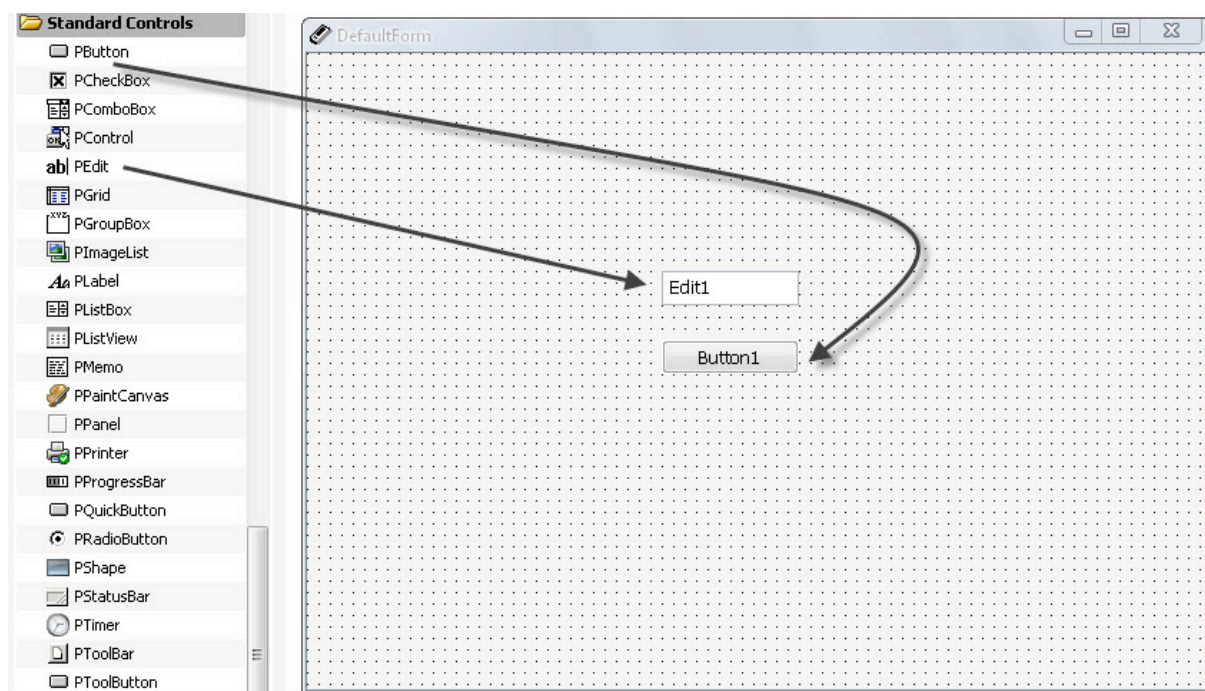


Figure 177: Place components in the form

- Once you have all the objects in place, click the **Project Manager** on the top left corner of the screen and return to it. In the **Project Manager**, click on the **PButton** so as to create an **OnClick** event. This event will be used to activate all the actions that we will be using once our **PButton** is clicked.

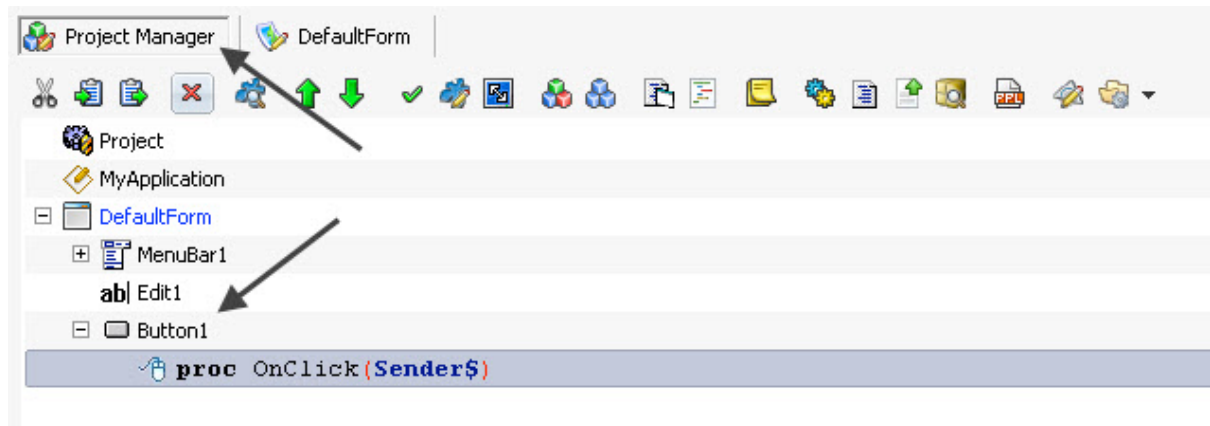


Figure 178: Project manager

- Like all the programming languages that require us to create variables so as to hold values and enable us to manipulate them, PIDE also does the same by providing the capability to create variables by first defining them and later declaring them. For doing this, drag a **PVariable** to the **OnClick** event. This will create a variable definition.

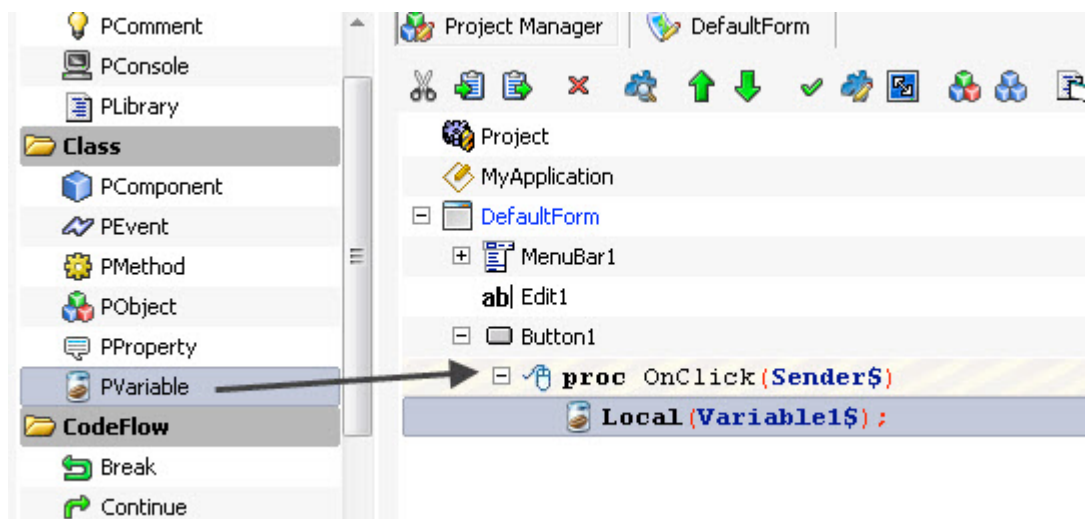


Figure 179: Drag PVariable

- Once the variable is declared, drag our newly created variable back to the **OnClick** event while keeping the **ALT** key pressed on the keyboard. This will create a variable that is ready to hold a value.



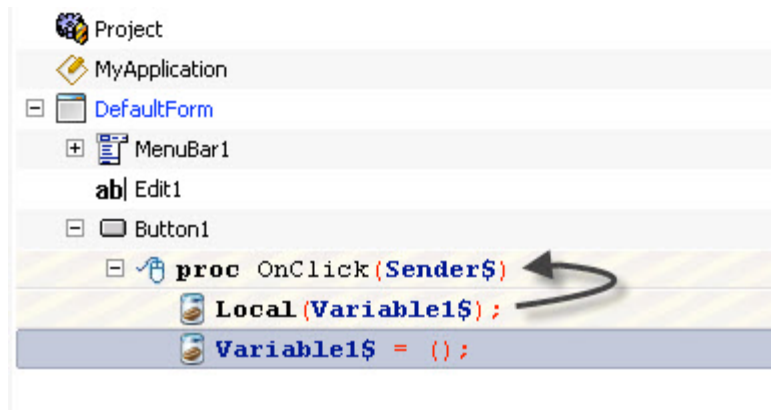
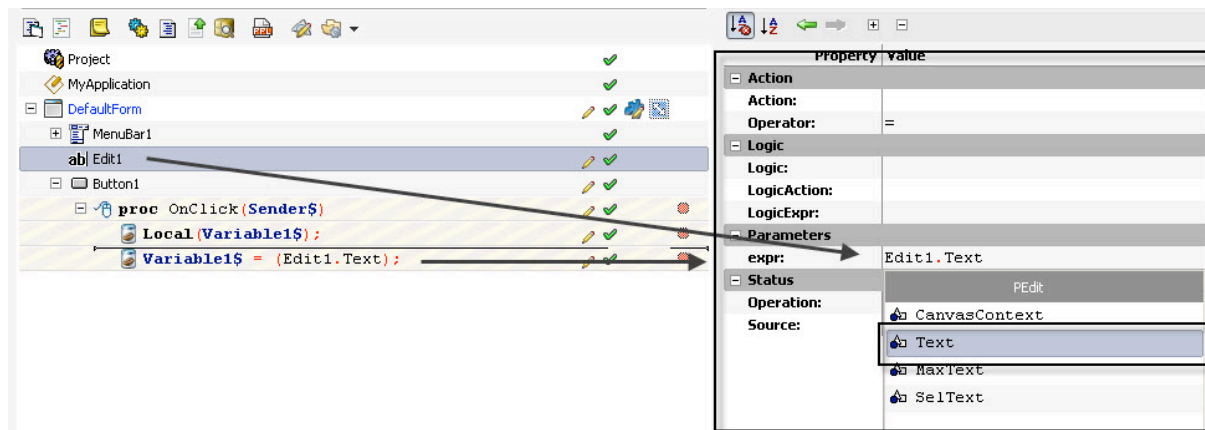


Figure 180: Drag with ALT key pressed

- Drag the **PEdit1** object to the **Expr** property of our newly created variable. This will initiate a code complete window.
- Select the Text property in the code complete window. This would give the value of the text entered by a user in our **PEdit** box to this variable.



- Drag a **For** loop on the **OnCreate** object. This would create a **For** loop that can be used to run a set of programming instructions specific number of times to get the factorial.

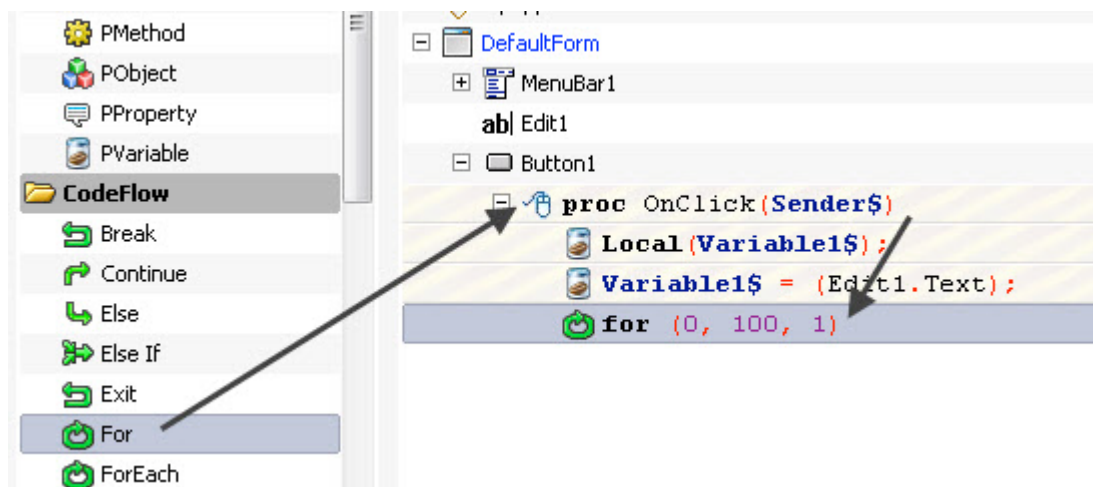


Figure 181: For loop

- Our **For** loop will need us to run the loop based on some conditions. Drag a new **PVariable** on the **OnCreate** event.

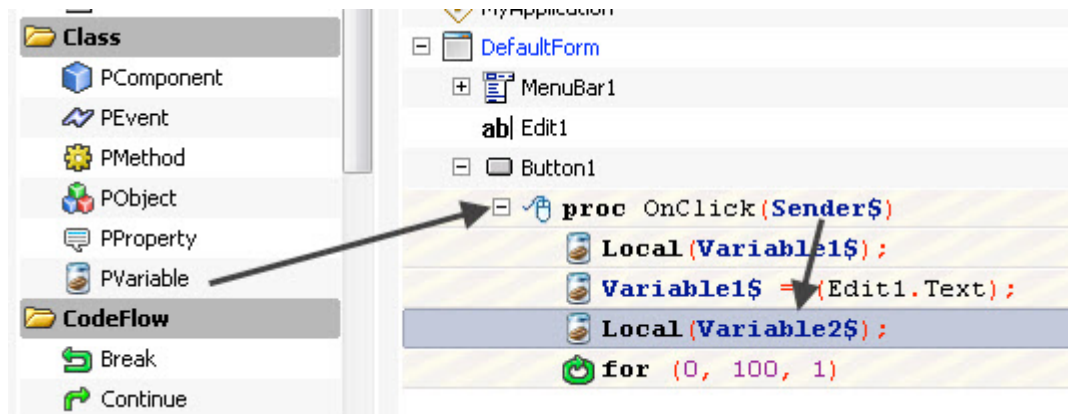


Figure 182: Drag PVariable on OnCreate

- Drag the newly created variable declaration back to the **OnClick** object while holding the **ALT** key. Once the variable is created change its **Expr** property to 1 so that our variable has a value 1.

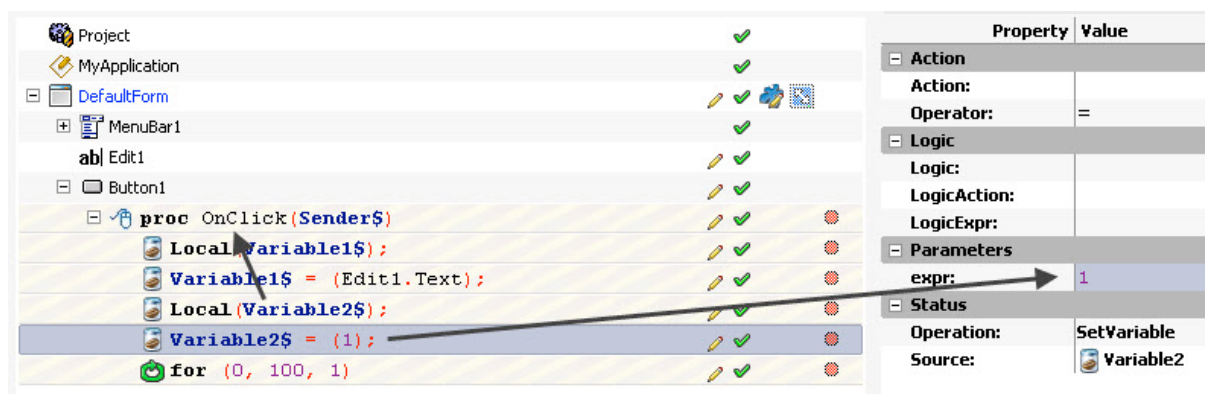


Figure 183: Drag to OnClick

- Now, in the **For** loop, drag the newly created variable to the start property. This will ensure that our loop starts at the value contained in this variable i.e 1.

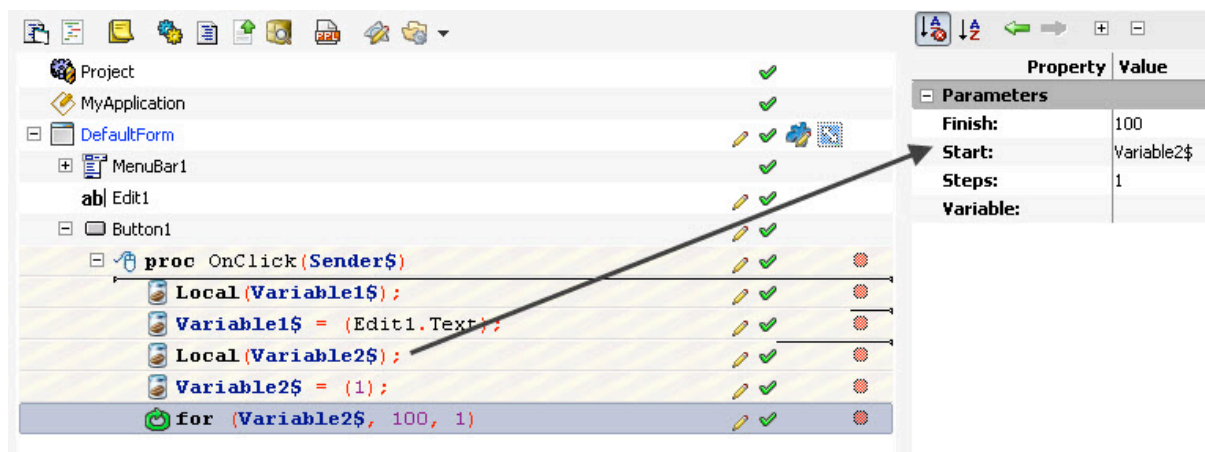


Figure 184: Drag variable to start property

- Drag Variable1\$ to the Finish property of our **For** loop. This will ensure that our loop plays the required number of times we want i.e equal to the number that is input in the **PEdit** box.

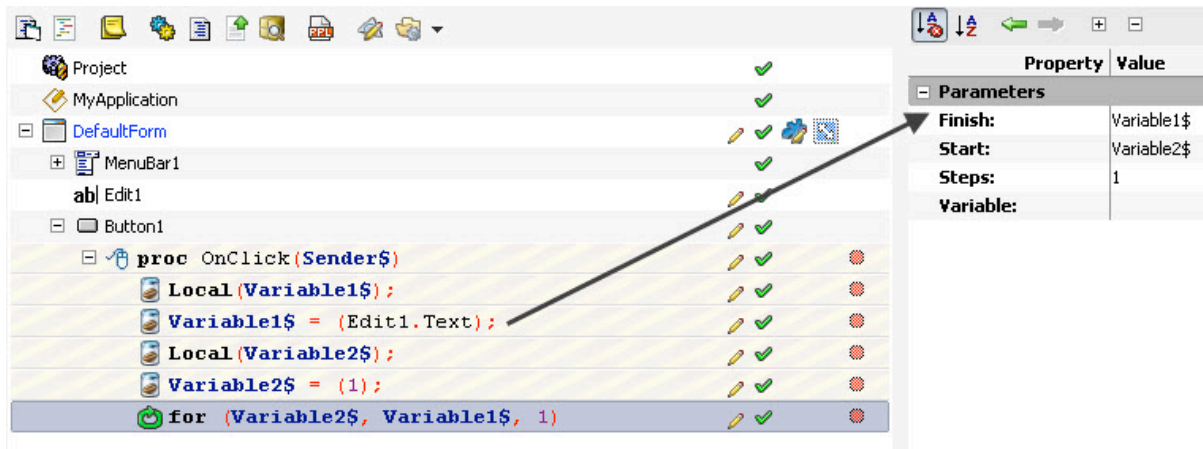


Figure 185: Drag Variable 1\$ to Finish

- Our **For** loop will run 1 step at a time so we will keep the **Step** property to 1.
- Drag a **PVariable** to **OnClick** event and drag the variable definition back to the **OnClick** object while keeping the **ALT** key pressed. This will create a new variable.

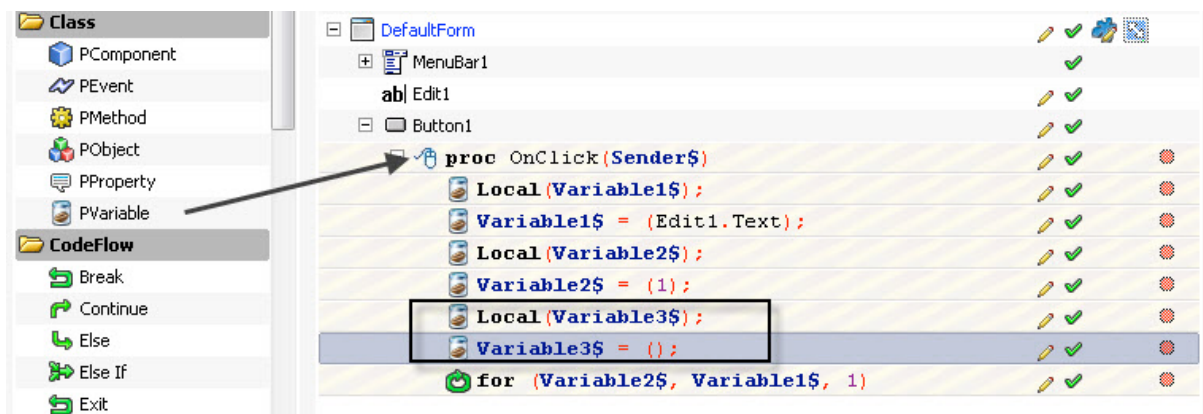


Figure 186: Create New Variable

- Once this variable is created, drag its definition one more time to the **OnClick** event while holding the **ALT** key. This will create one more instance of this same variable. Change the **Expr** property of one of this variable to 1.



Figure 187: Change Expr property

- Drag the other variable on the **For** loop.



Figure 188: Drag variable on For Loop

- Change the **Expr** property of our new variable (**Variable3\$\*Variable2\$**). Doing this would ensure that variable3\$ will contain multiples of all the numbers from 1 (**variable2\$**) to the number that is entered in the **PEdit** box(**variable1\$**).

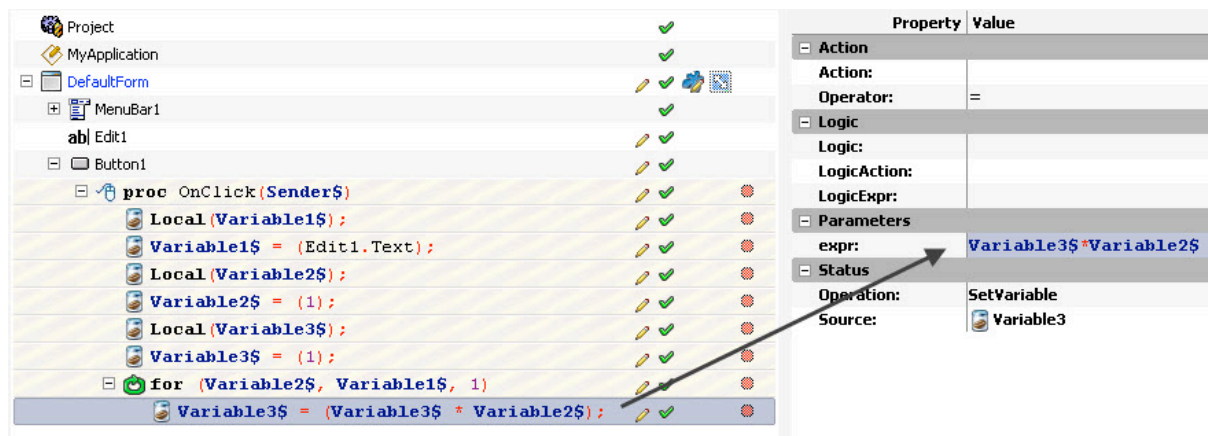


Figure 189: Change Expr property

- Click on the **OnClick** event and press **Ctrl+Space** bar to initialize a **code complete** window. Select **ShowMessage** property from the window.



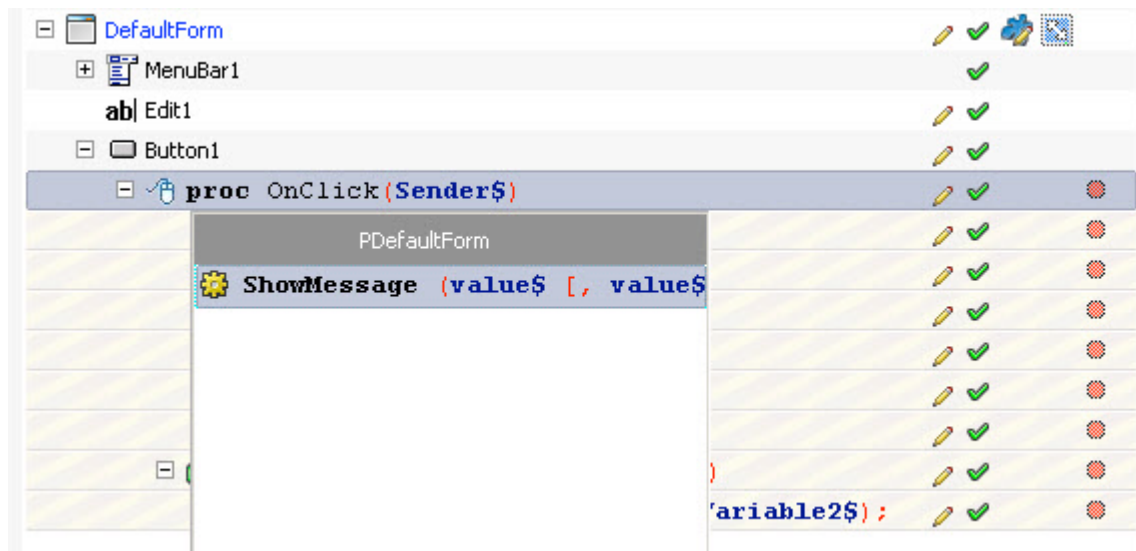


Figure 190: Code complete window

- Change the **Value** property of **ShowMessage** object to **Variable3\$**.

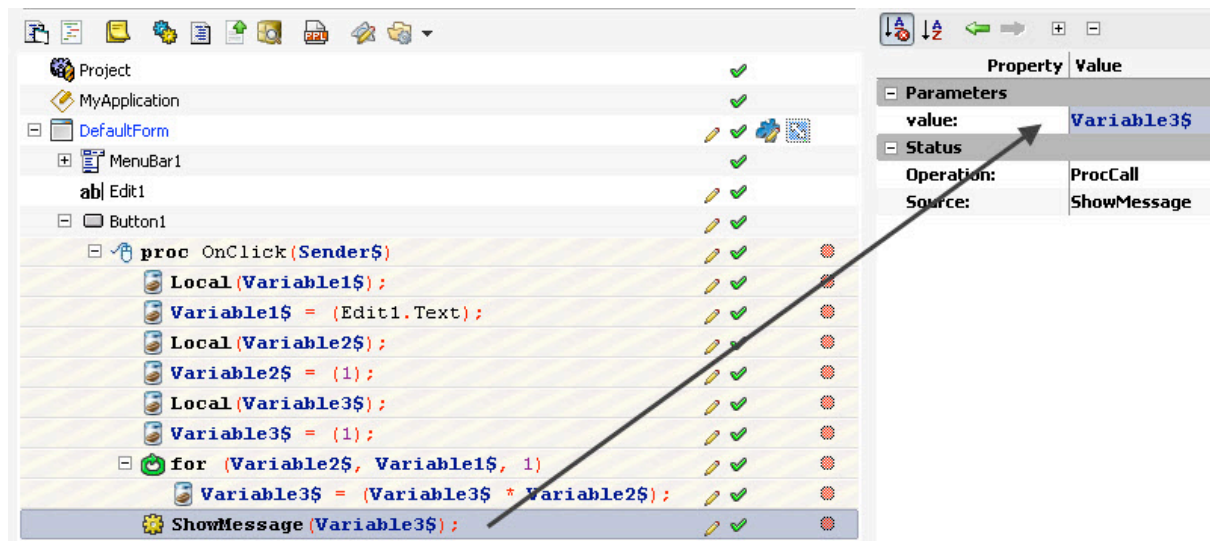


Figure 191: Change Value Property

- Press **Ctrl+S** to save the program. Alternatively, you can also go to **File>Save As** to save your project to wherever you want.

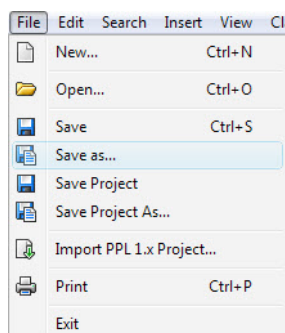


Figure 192: Save Project

- Now, press **F5** and check your program for results!

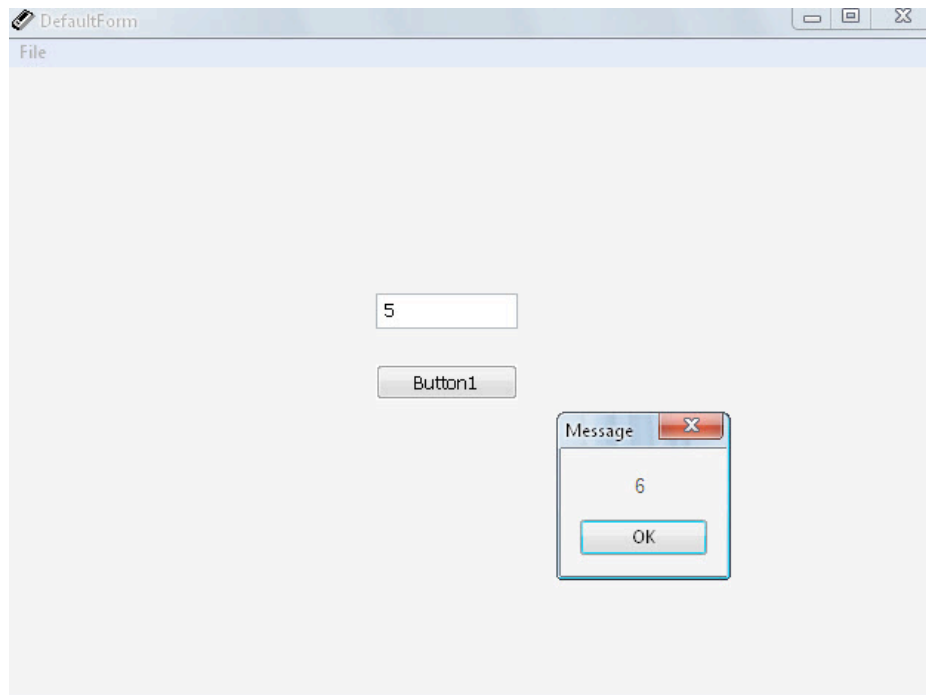


Figure 193: Output



## Loops and Conditions: While Loop

Quite similar to **For** object, the **While** object is used to perform a series of repetitive actions on a statement based on a condition but unlike the **For** statement, the **While** statement does not need as many parameters. A **While** condition will continue the iteration process while a certain condition is true. For example, whilst the **For** statement decides to process a set of instructions or not unless a condition is met and the corresponding start and end values are set, a while condition on the other hand will process the instructions enclosed within the while statement irrespective of the starting or the ending point. **While** condition is especially useful where a user is not familiar with the amount of loops certain set of instructions have to make.

Given below is a **While** condition that will allow you to process a condition without the starting or the ending position set. The example given below will display a set of lines that are equal to the number input by a user. The important thing to note here is that these lines are generated dynamically.

- Start by creating a new **Desktop Form** project. To do so, start PIDE and press Ctrl+N and select **Desktop Form Project** from the “**New Project Type..**” window. Alternatively, you can also go to **File> New** and select **Desktop Form** from the “**New Project Type..**” window.

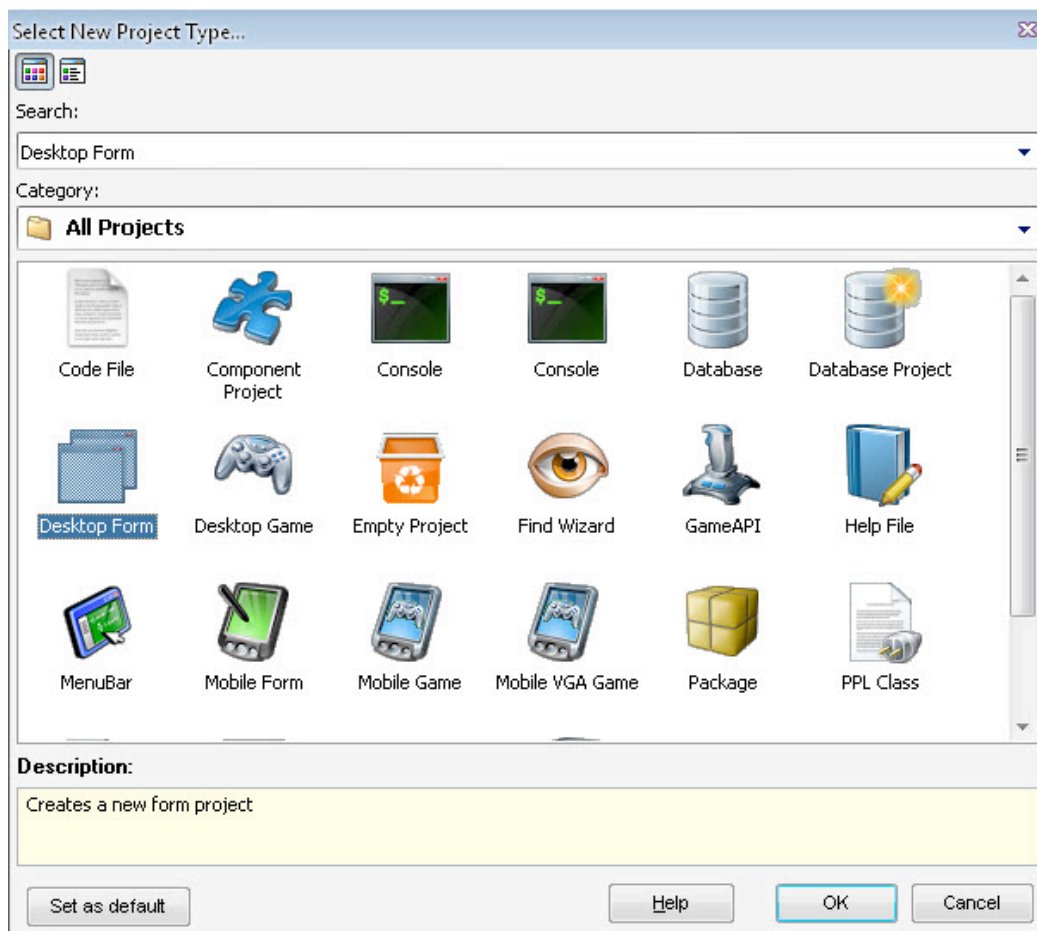


Figure 194: Create new project

- In the project, double click the **Default Form** object to enter the Form editor.

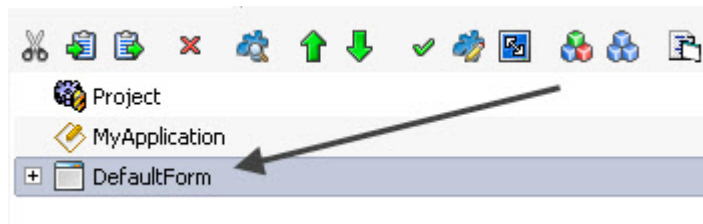


Figure 195: Double click on DefaultForm

- In the **Default Form** Editor, we will place one **PEdit** box along with one **PButton** and a **PLabel** on the form editor. While the **PEdit** box will contain the number we want to use for printing the text, **PButton** will be used for the initiation for all the actions we will have on the number and **PLabel** will be used to print the content. On the **Components Pane**, look for **PEdit** object and click on it; now, click on the form to place the **PEdit** object on the form. After placing a **PButton** and **PLabel** just like the other **PEdit** object, you are ready to start visual programming. Use the figure given below to arrange the components.

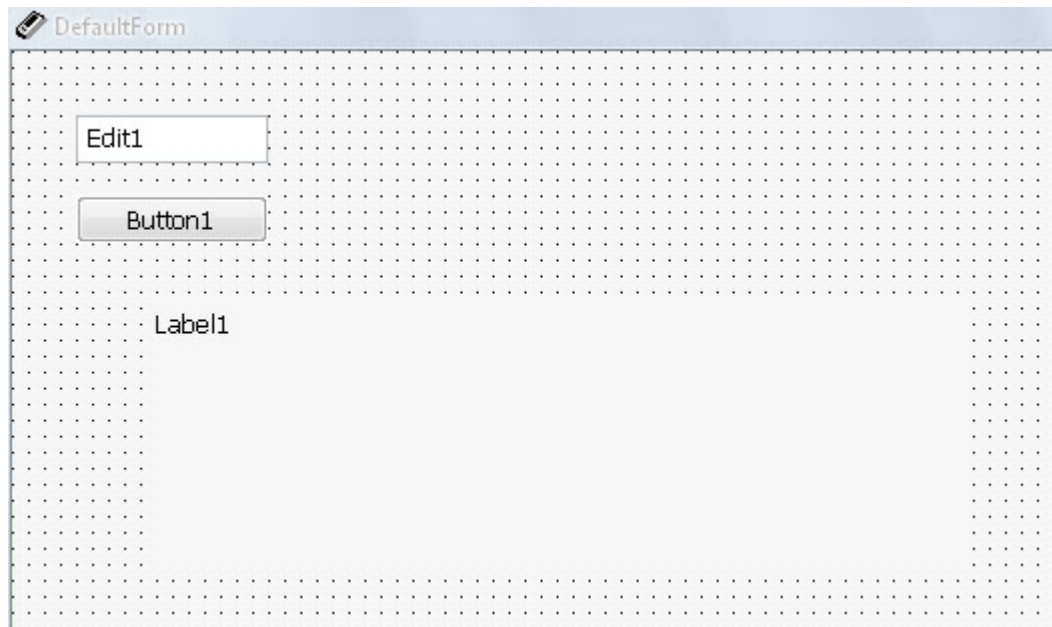


Figure 196: Component layout

- Click on the **Project Manager** button available on the top left of the screen and double click the **PButton** to create an **OnClick** event. This **OnClick** event will be used to initialize the various actions we will have on the data entered in the **PEdit** box.

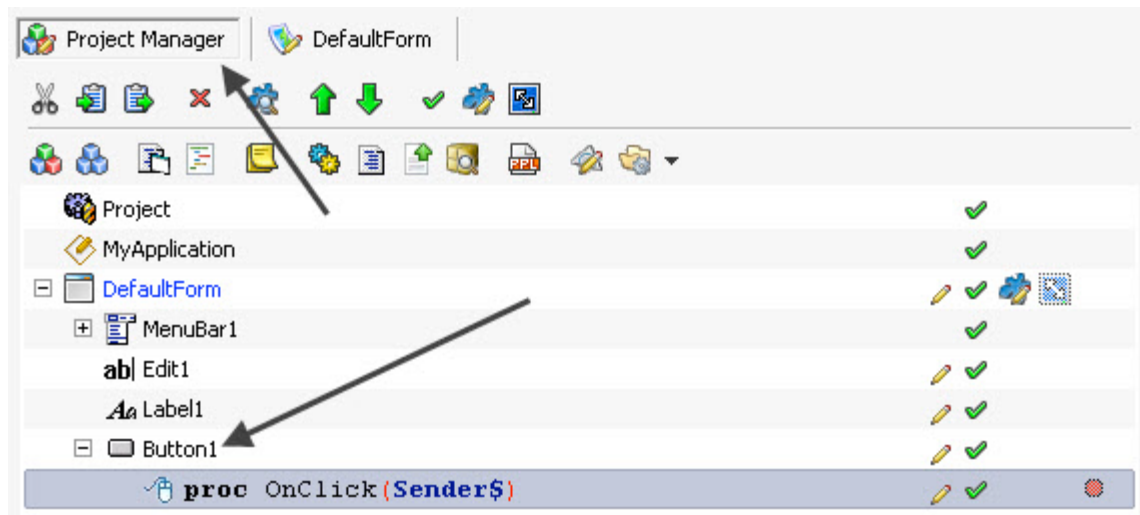


Figure 197: Project manager

- Drag a **PVariable** from the component panel on the **OnClick** event. This will create a variable declaration for our variable.

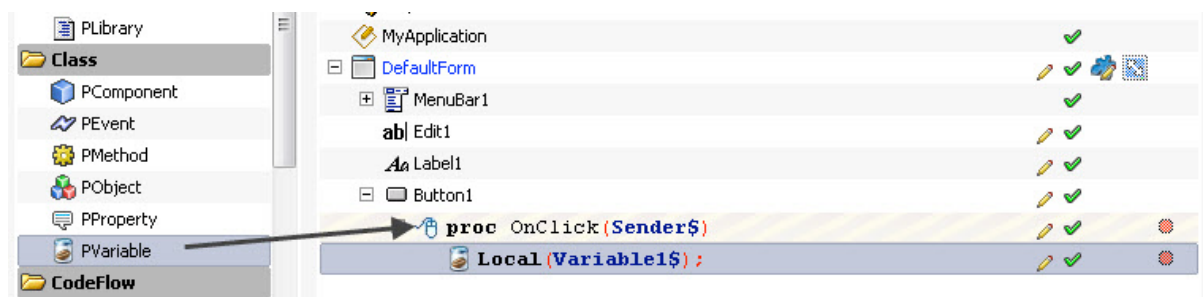


Figure 198: Drag PVariable to OnClick

- Drag the newly created variable declaration on the **OnClick** event while holding the **ALT** key the keyboard. This would create a new variable that we can use to hold values.

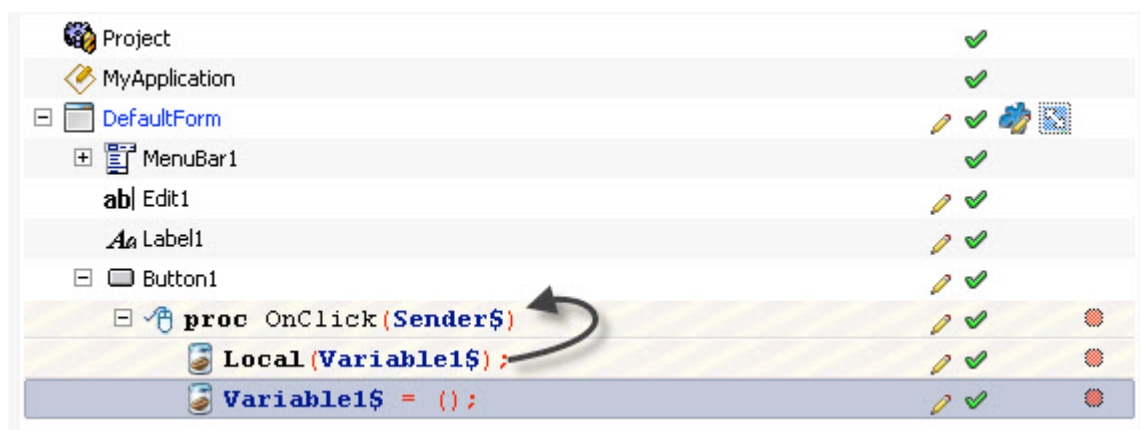


Figure 199: Drop Variable1\$ to OnClick

- Drag the **PEdit** object to the **Expr** property of our newly created variable and select **Text** property in the code complete window. This would assign the value entered in the **PEdit** box to this variable.

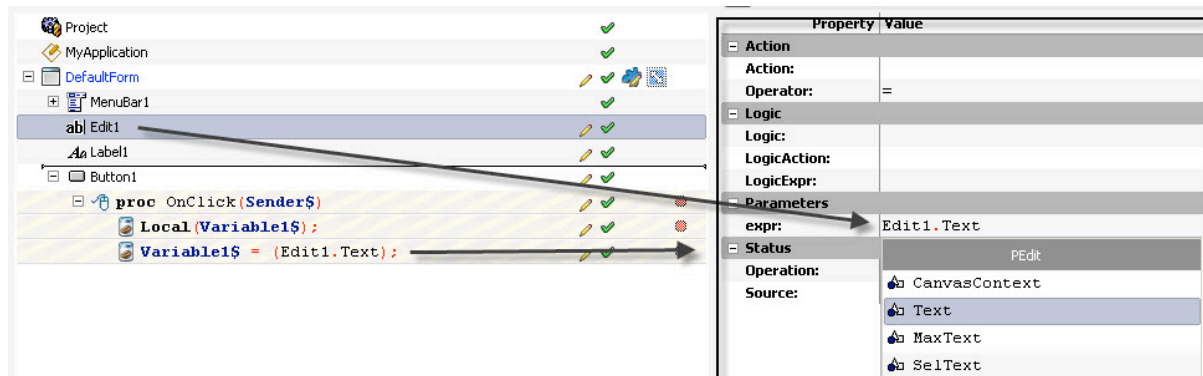


Figure 200: Drag PEdit1 object to Expr property of Variable 1\$

- Drag a While object from the **Components Panel** to the **OnClick** event.

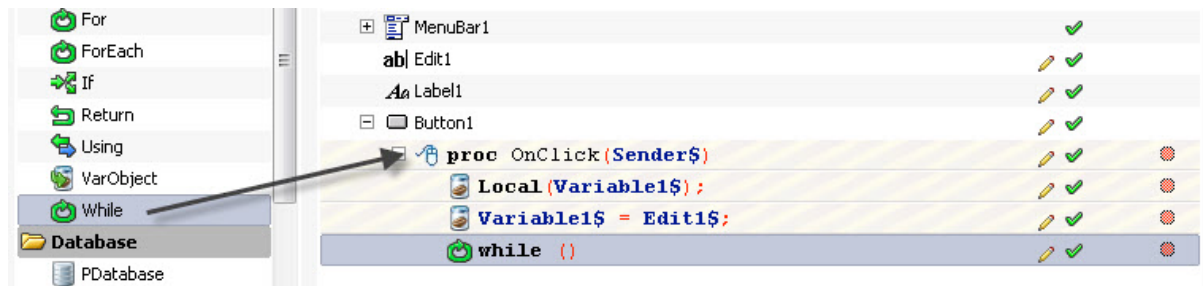


Figure 201: Drag While to Project Manager

- Go to the While statement and change its **Expr** property to **(Variable1\$>=1)**

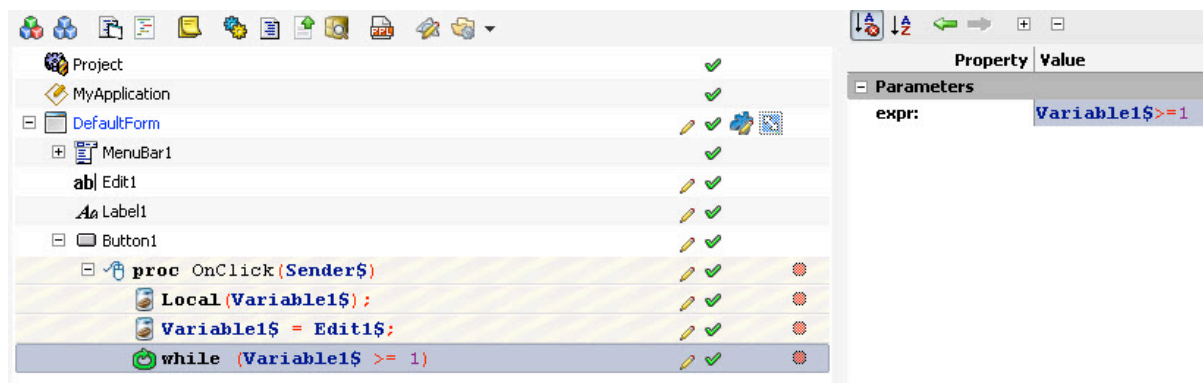


Figure 202: Change the expr property

- Drag a **PVariable** to the **OnCreate** event so as to create a declaration for a new variable and drop this newly created variable declaration onto the **OnClick** event while holding the **ALT** key on the keyboard. This will create a new variable.

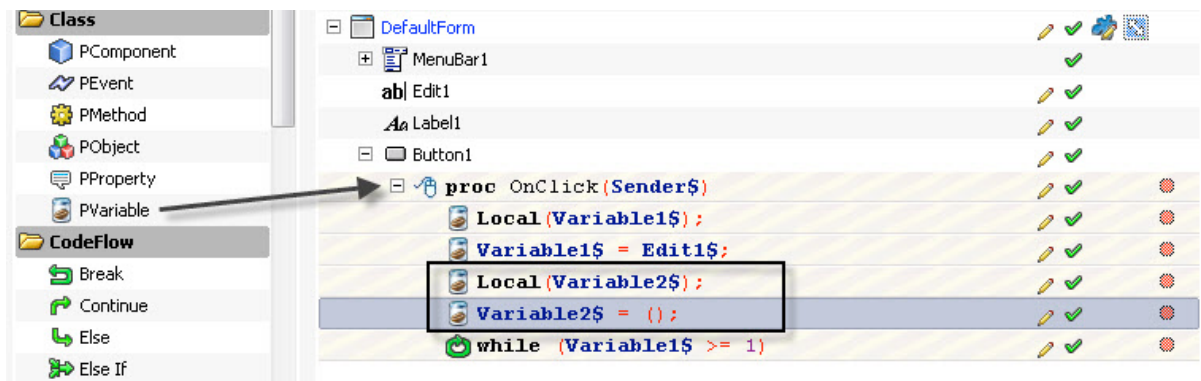


Figure 203: Drag PVariable to Project Manager

- Drag this variable to **while** object.

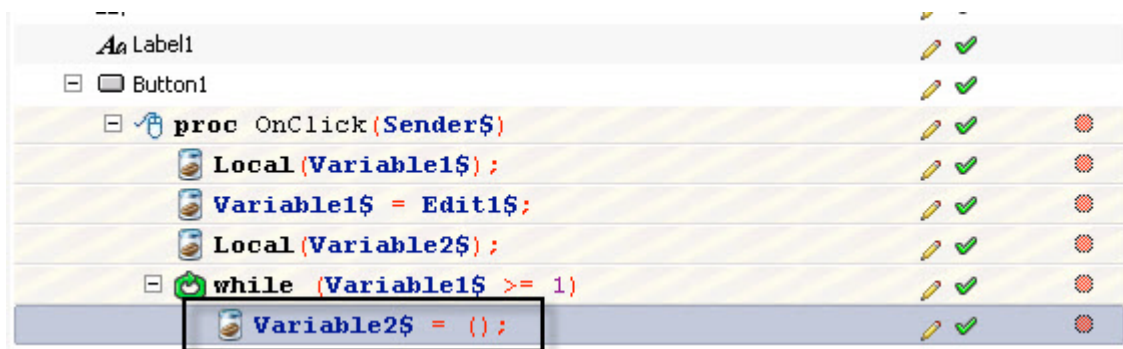


Figure 204: Drag Variable

- While the variable is still selected, change its **Expr** property to **Variable2\$+" Number is repeated "+Variable1\$+" times. "** This statement will assign a preformatted text to the variable.

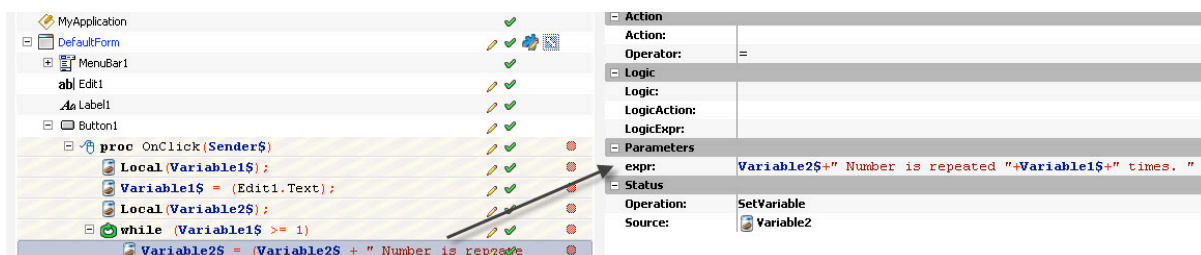


Figure 205: Change Expr

- Now we will place a decrement that would decrease the **Variable1\$** one at a time. For doing this, drag the declaration of **variable1\$** on the **OnCreate** event while holding the **ALT** key on the keyboard.



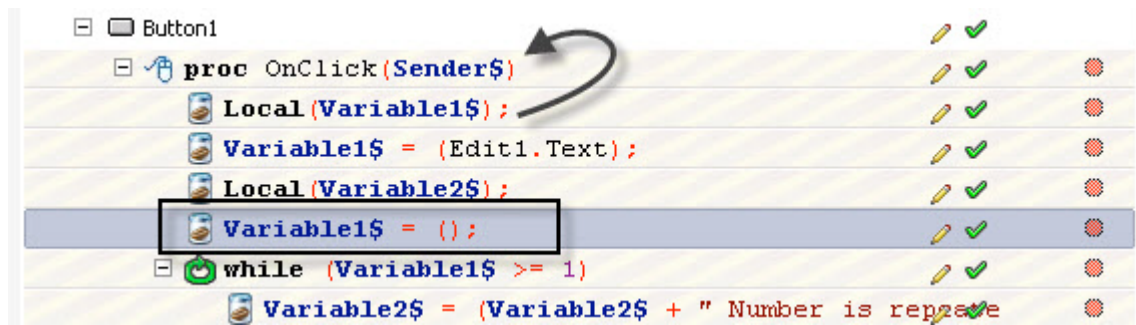


Figure 206: Drag Variable1\$ with ALT key

- Drag this variable to the **While** loop and change its **Expr** property to **(Variable1\$-1)**. This statement will decrease our variable by 1 every time the loop plays. Alternatively, you can also just select **-=** in the Operator property and put 1 in the **Expr** property.

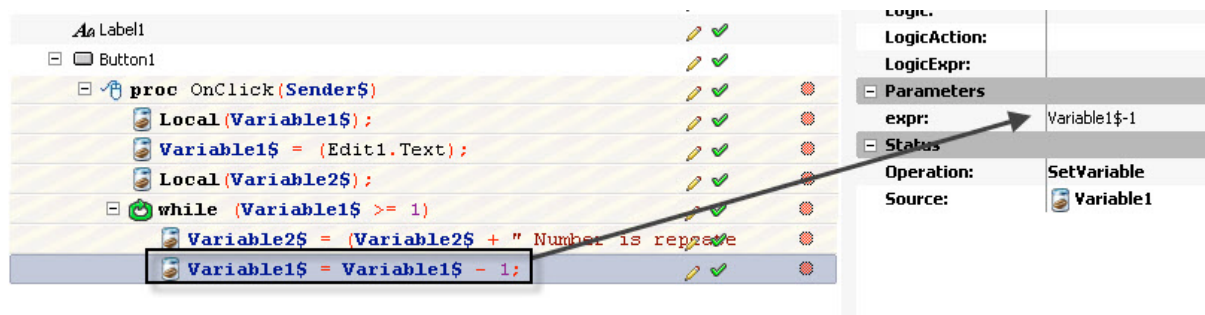


Figure 207: Change Expr property

- Now we have to print the preformatted text to the screen. For this we will use our **PLabel**. Drag the **PLabel** object to **OnClick** object and select the caption property in the **Code Completion** box that appears.

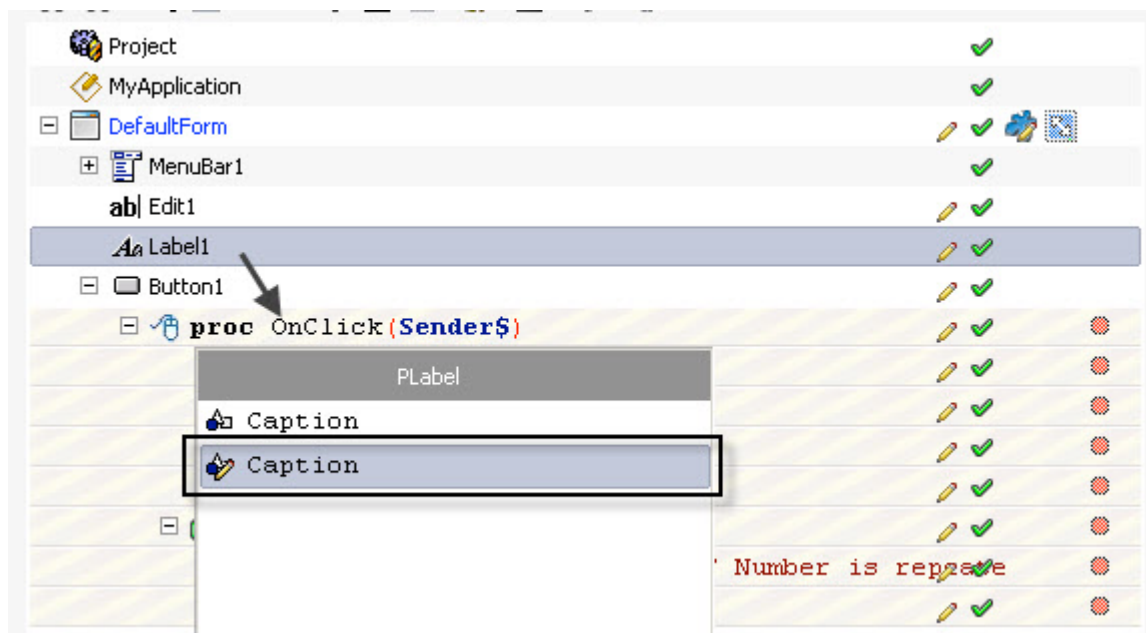


Figure 208: Drag PLabel



- Assign **variable2\$** to the caption property of **PLabel** we have just created. For doing this, drag **Variable2\$** to **Expr** property of the **PLabel** statement.

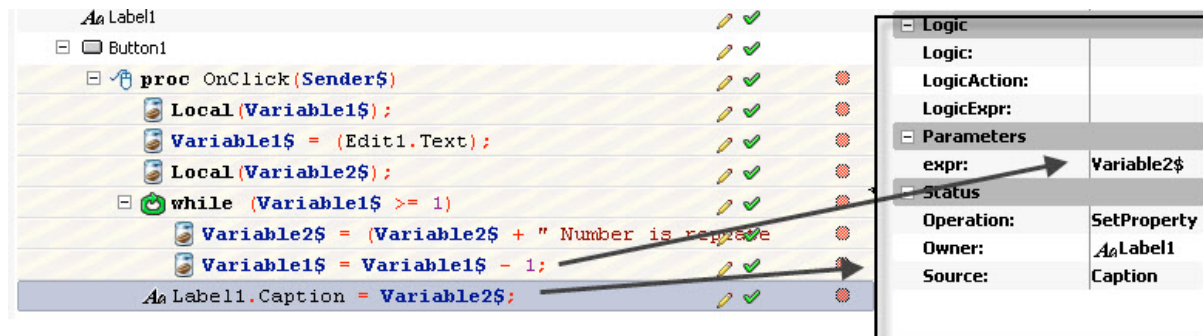


Figure 209: Drag Variable2\$ to expr property

- Press **Ctrl+S** to save the program. Alternatively, you can also go to **File>Save As** to save your project to wherever you want.

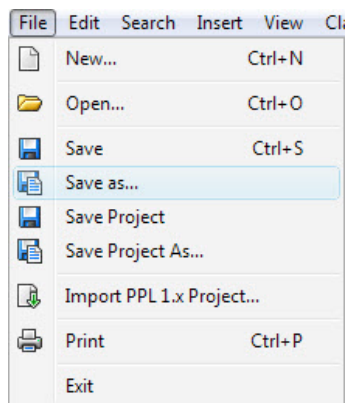
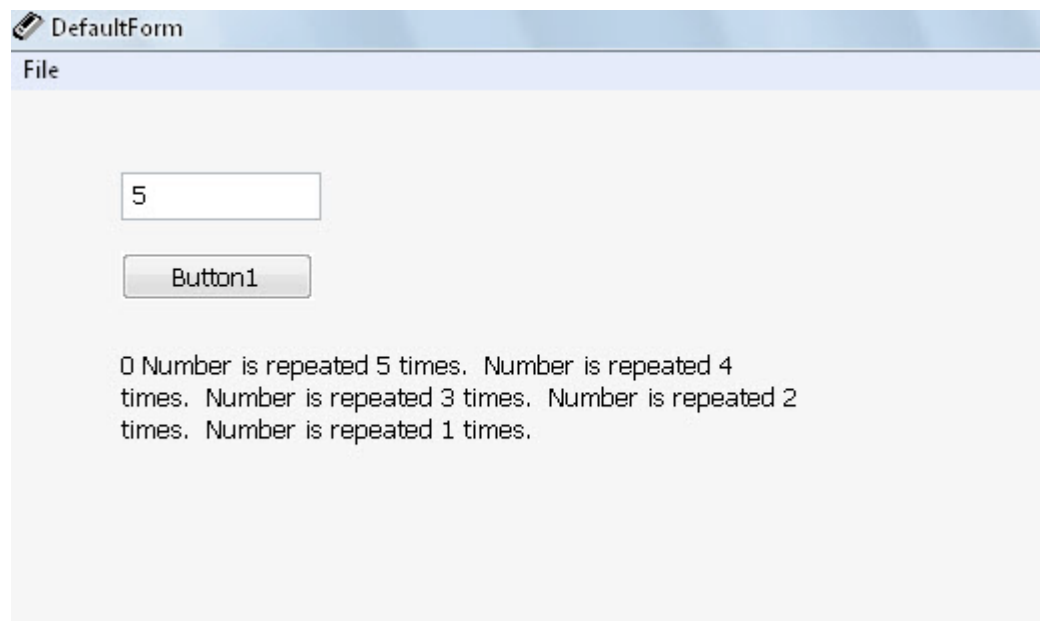


Figure 210: Save project

- Now, press **F5** and check your program for results!



**Figure 211: Output**

## Loops And Conditions: ForEach Loop

A very useful object used for performing iterations, **ForEach** object is like a shorthand for specifying the For object but for each item in another object with individual items. It is specially used where a list of items has to be iterated against an action.

Given below is an example that will allow you to see the usage of **ForEach** in PIDE for setting iterations. The given example scans each item available in a list (**PValueList**) and shows it in a message box individually.

- Create a new **Desktop Form** project by pressing the **Ctrl+N** key on the keyboard or going for the **File>New** option. In the **Select New Project Type..** window, select **Desktop Form** project.

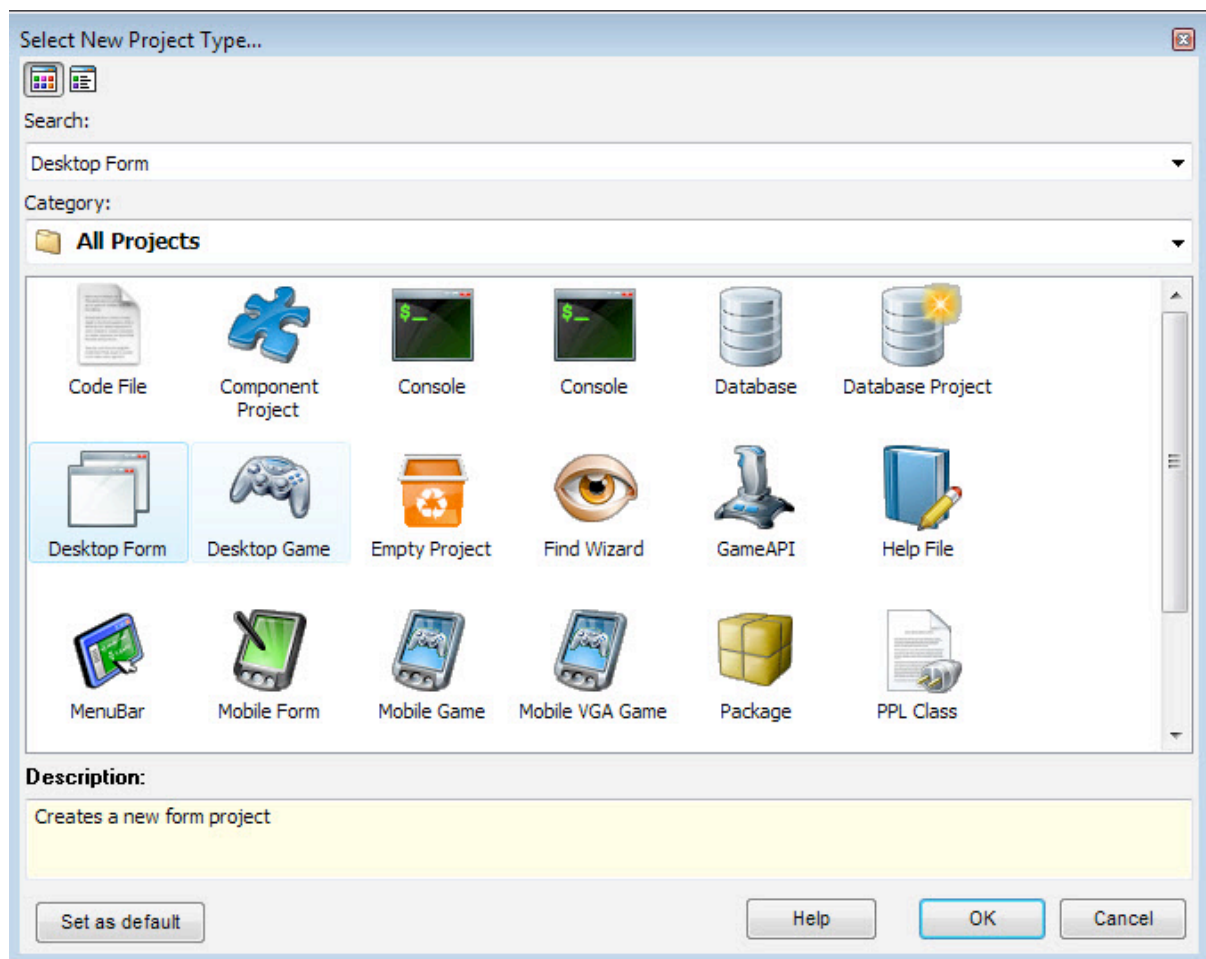


Figure 212: Create new project

- Right click on the **DefaultForm** and select **OnCreate** from the Events menu. This event would allow you to trigger actions when this form is created.

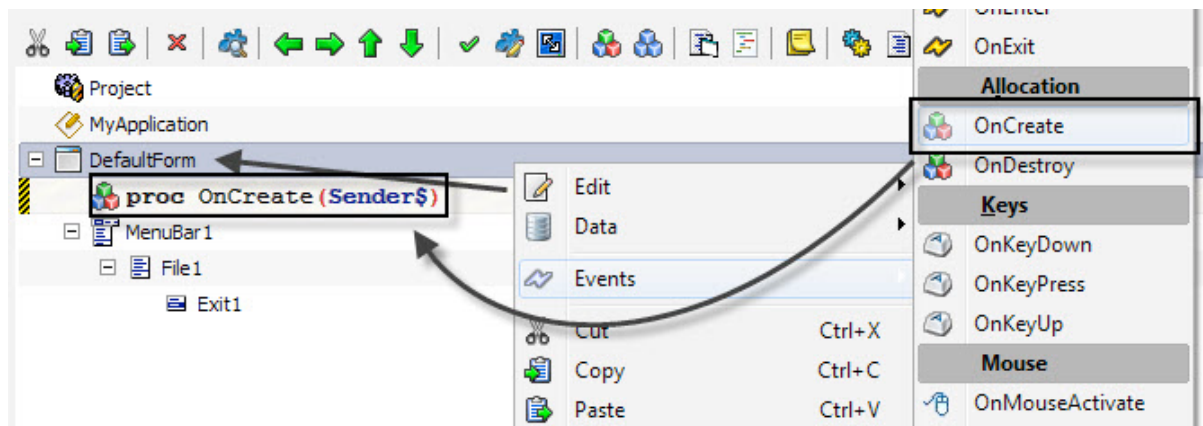


Figure 213: Create OnCreate object

- Add a **PValueList** to the project. For doing so, drag a **PValueList** on the **OnCreate** event.

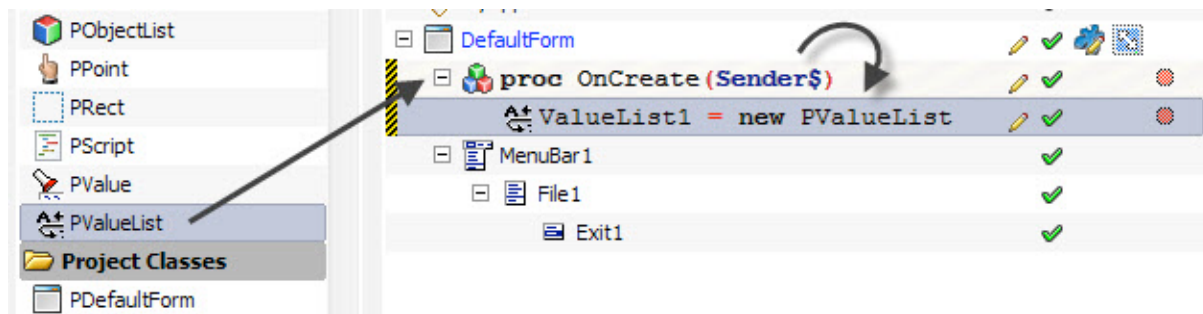


Figure 214: Drag PValueList on OnCreate event

- Now we will fill the items in the **PValueList**. To do this, double click the Value property of **PValueList** object and write down the list like in the figure below.

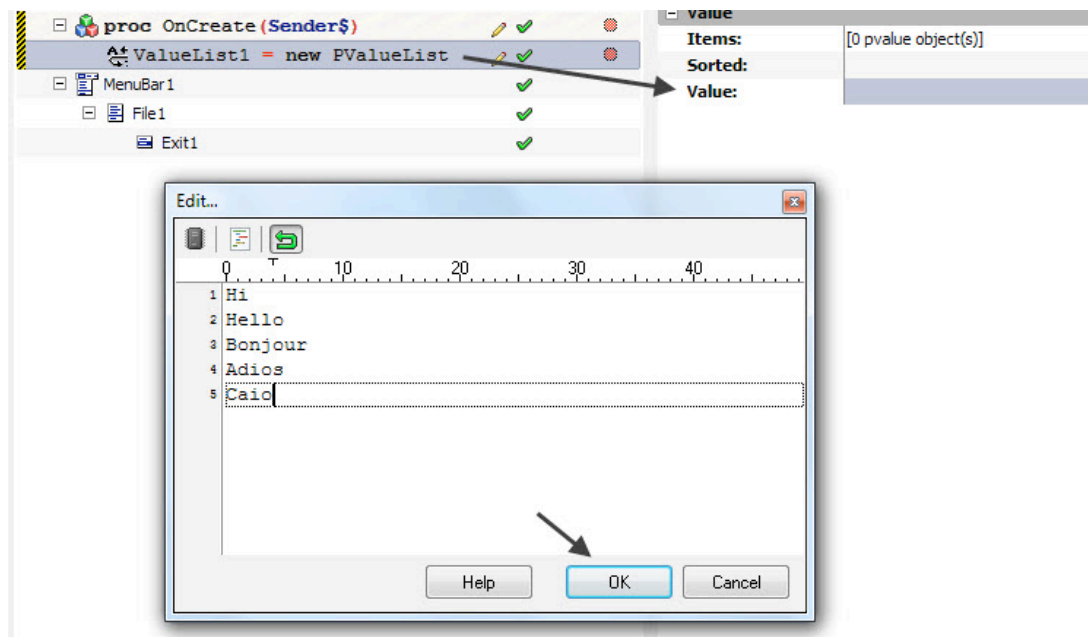


Figure 215: Fill items

- Now that the **PValueList** has been created, we need to join it with a **ForEach** object that will run a condition for each item of the list. Drag a **ForEach** object below the **PValueList** object.

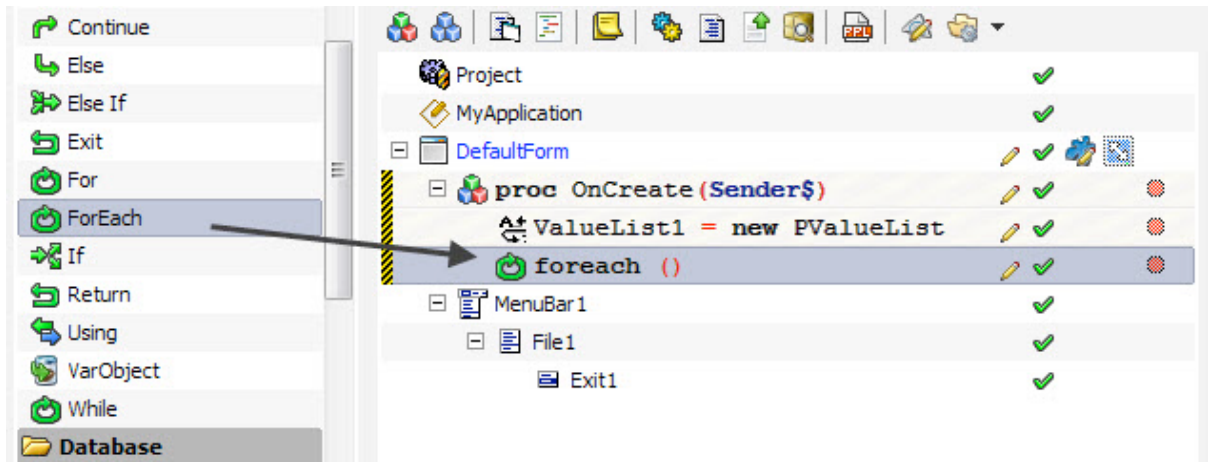


Figure 216: Drag ForEach object

- Drag the **PValueList** object we created earlier to the **ForEach** object's **Expr** property and select **Items** from the **Code Complete** window.

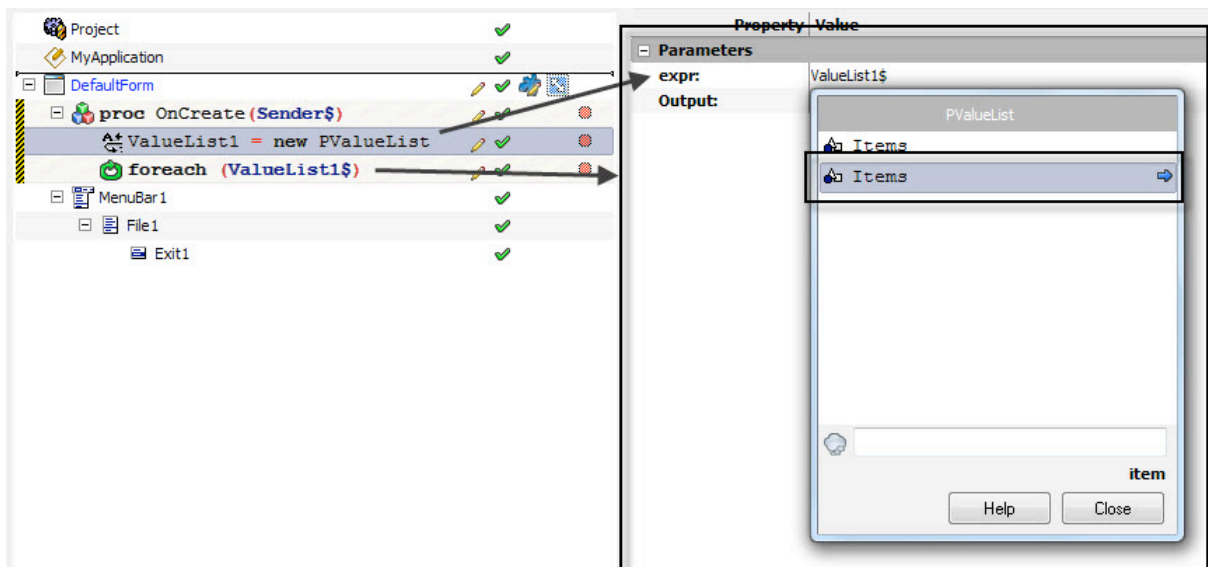


Figure 217: Select Items

- The **Output** property of the **ForEach** is the output variable where you want to store the current item. Since PPL uses a very relax type-checking, you can assign any object type to a variable as type-checking is done at runtime, not at compile time. For adding a variable to the output property, drag a **VarObject** component from **Component Pane** under **PValueList** object so that it is created before **ForEach**. The **VarObject** is used to tell the compiler which variable class it will hold.

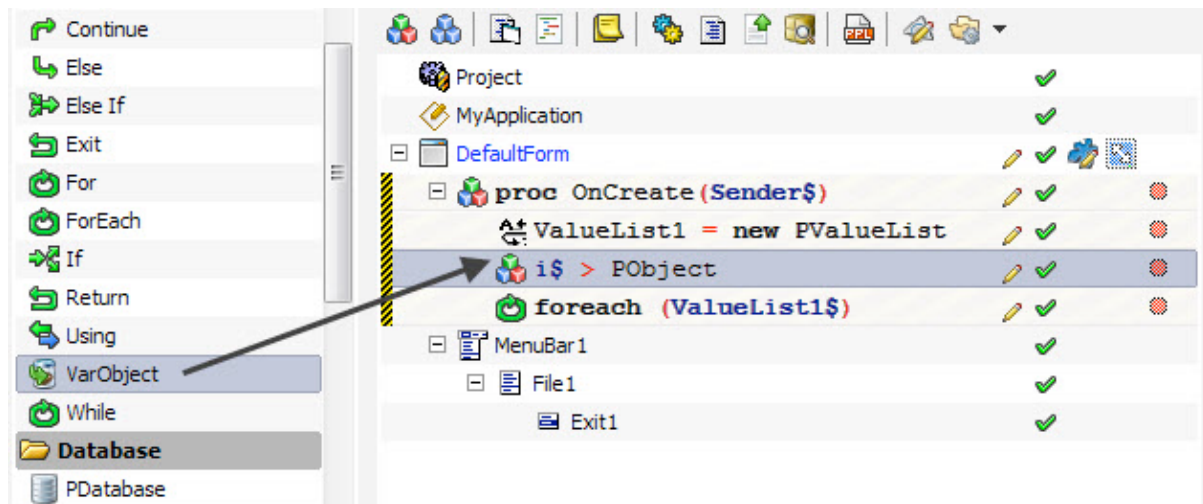


Figure 218: VarObject

- We now need to tell the visual code generator what class type object the output variable is going to contain. Select **PValue** from the **ClassName** property of the newly created **VarObject**.

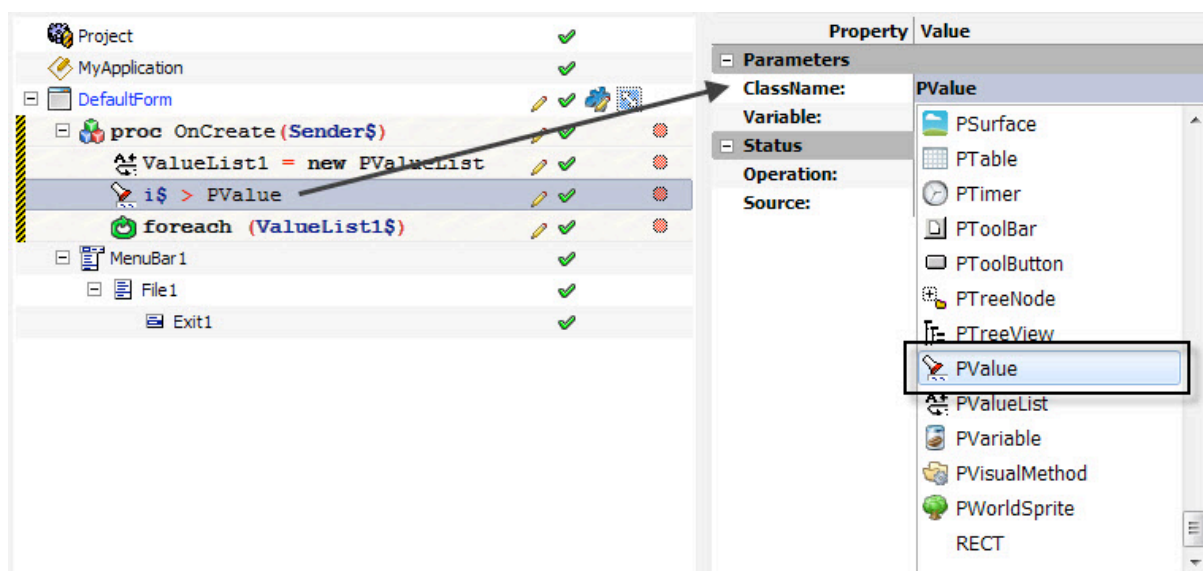


Figure 219: Select PValue

- Now drag the **PValue** statement to the Output property of the **ForEach** object.

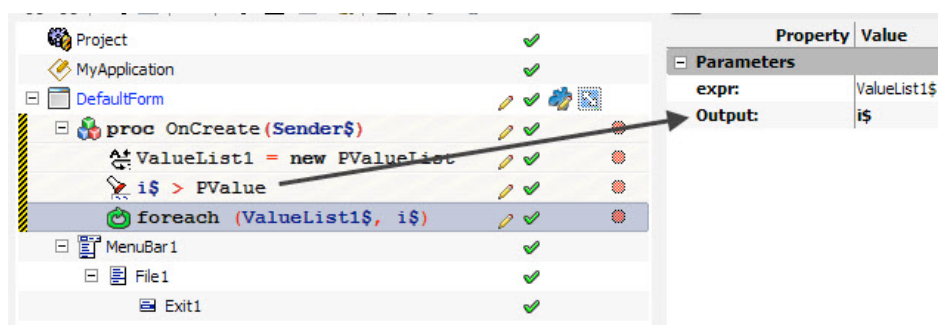


Figure 220: Drag PValue object to Output Property



- For the output, drag the **PValue** statement on the **ForEach** object while holding **ALT** key on the keyboard. In the **Code complete** window that appears, select **Show** to print the values on the screen.

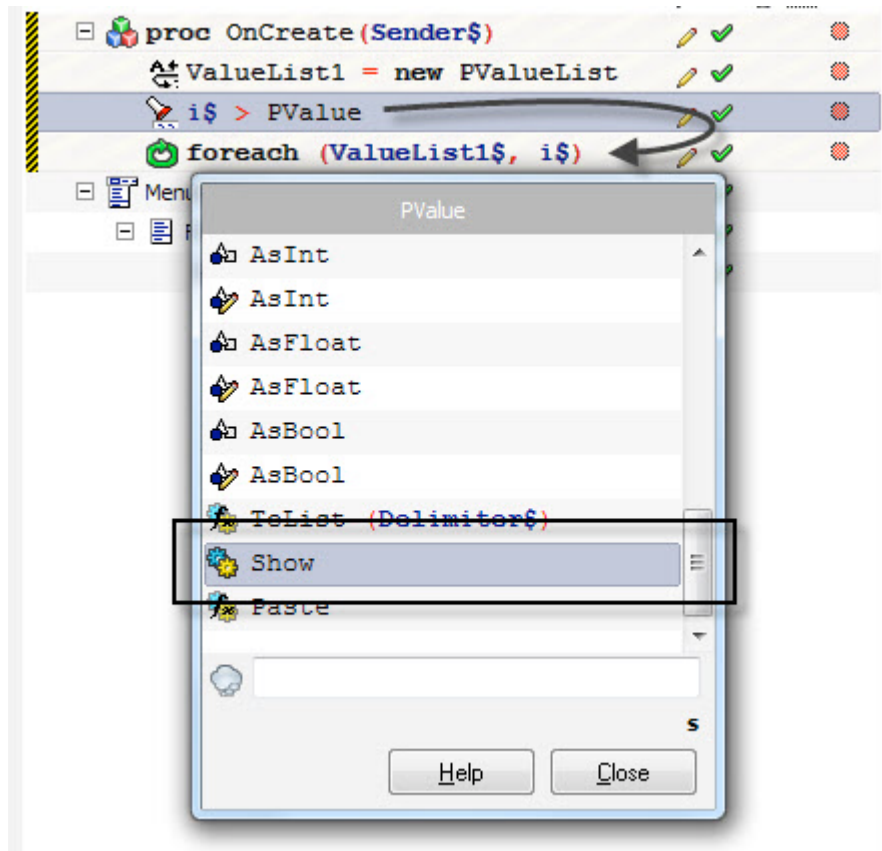


Figure 221: Select Show

- Press **Ctrl+S** to save the program. Alternatively, you can also go to **File>Save As** to save your project to wherever you want.

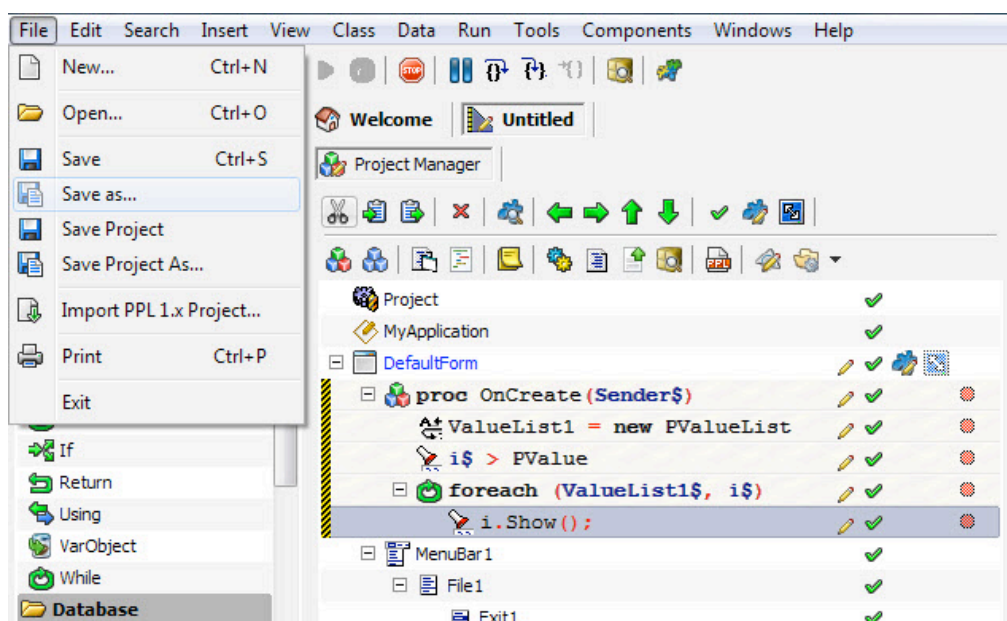


Figure 222: Save project

- Now, press **F5** and check your program for output. You will see different message boxes, each representing each list item available on the **PValueList**.

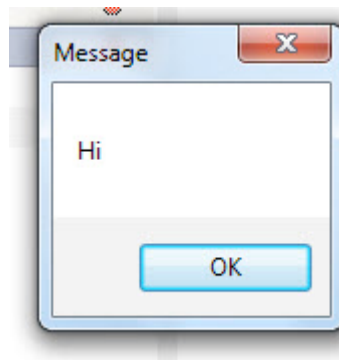


Figure 223: Output

## Loops and Conditions: If Condition

Usage of conditional statements is crucial for any program that requires a higher degree of processing and logic. PIDE provides the **IF** object as the very basic of making conditional decisions. With the use of **IF** condition, users get the facility to evaluate a specific condition and then specify its result after analysis of the outcome of their condition. In the example given below we will create a program to find the subtraction of two numbers. This program will allow a user to input two numbers and alert him/her if the first variable is lower than the second variable.

In this program, we will make use of two **PEdit** objects to enter two numbers for subtraction, One **PLabel** for the subtraction sign and one **PButton** to act as a trigger for the actions on the program.

- Start by creating a new **Desktop Form** project. To do so, start PIDE and press **Ctrl+N** and select **Desktop Form Project** from the “**New Project Type..**” window. Alternatively, you can also go to **File> New** and select **Desktop Form** from the “**New Project Type..**” window.

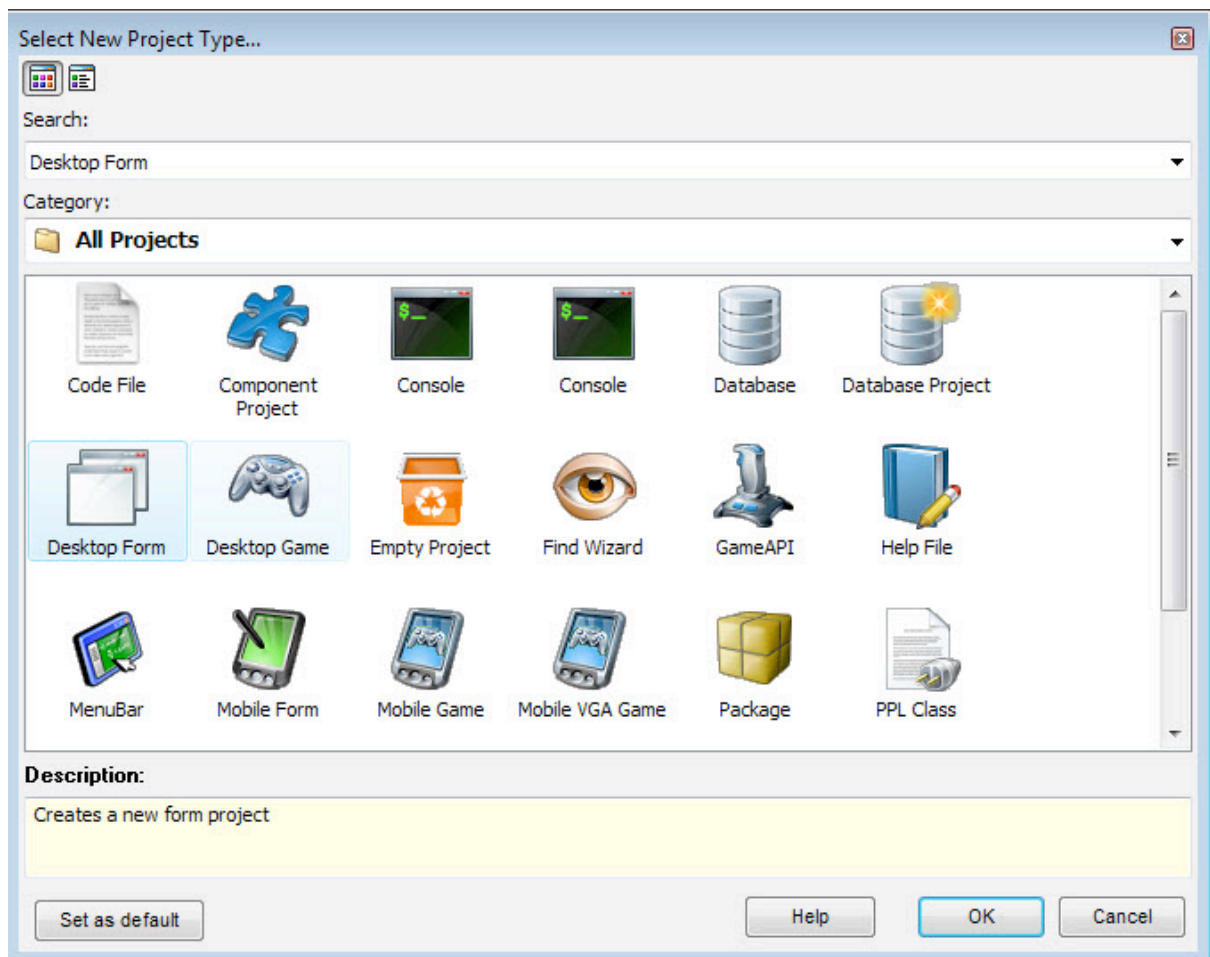


Figure 224: Create new project

- In the project, double click the **Default Form** object to enter the Form editor.

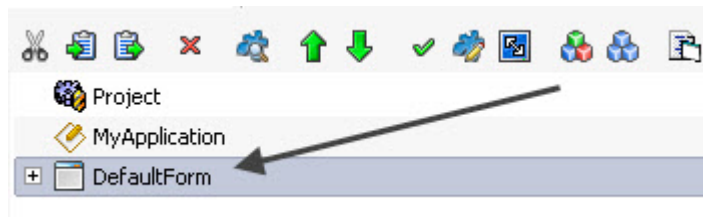


Figure 225: Double click

- In the **Default Form** Editor, we will place two **PEdit** objects along with one **PLabel** and one **PButton**. While the **PEdit** boxes will contain the numbers to subtract, the **PLabel** will be used to hold the subtract sign, **PButton** will be used for the initiation for all the actions we will have on the numbers. On the **Components Pane**, look for **PEdit** object and click on it; then, click on the form to place the **PEdit** object on the form. Similarly, put another **PEdit** on the form like shown in the figure. Now, find a **PLabel** on the **Components Pane** and place it in between the two **PEdit** boxes. After placing a **PButton** just like other components, you are ready to configure your components.

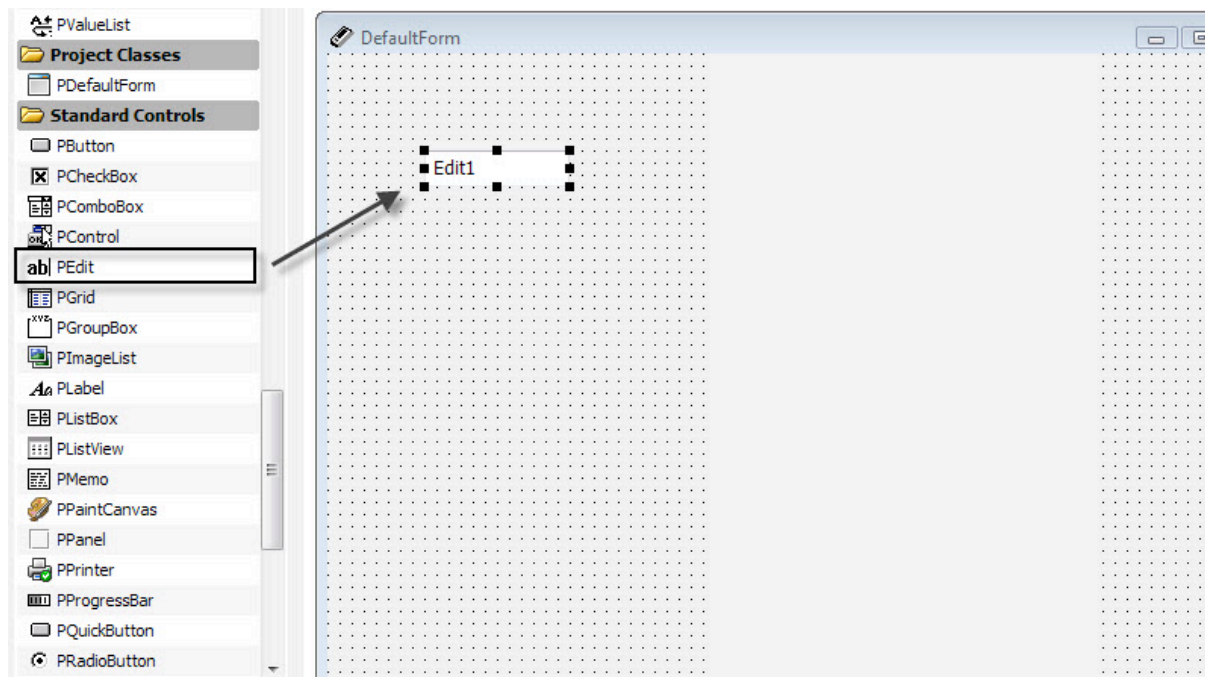


Figure 226: Place PEdit

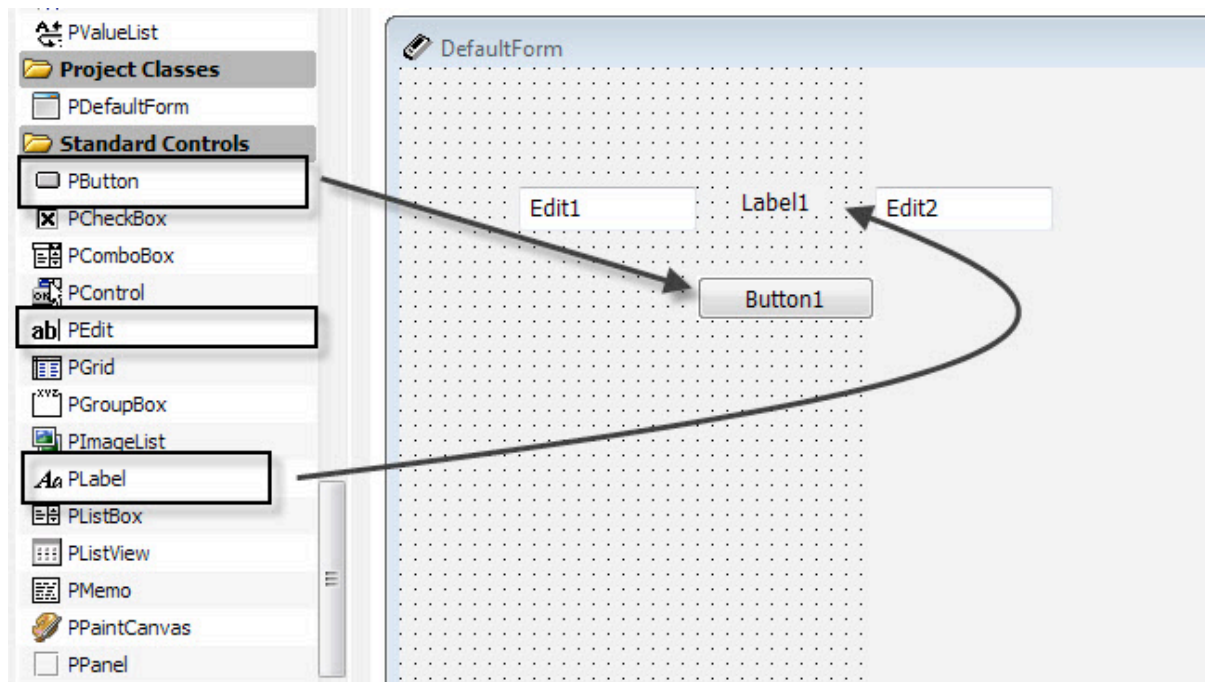


Figure 227: Place Components on Desktop form

- Click on the **PLabel** and change its caption property to represent subtraction sign by placing a – symbol there.

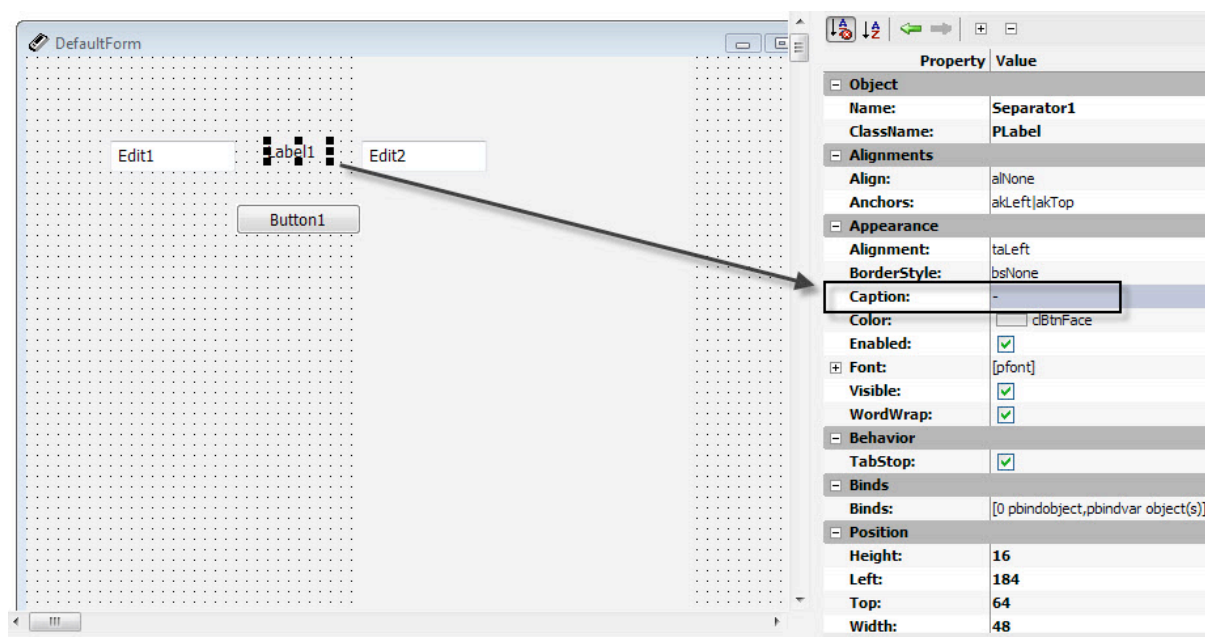


Figure 228: Place PLabel on Desktop Form

- You can also change the caption property in the **PButton** to say “Subtract”
- Go back to the **Project Manager** by clicking the **Project Manager** button at the upper left corner.

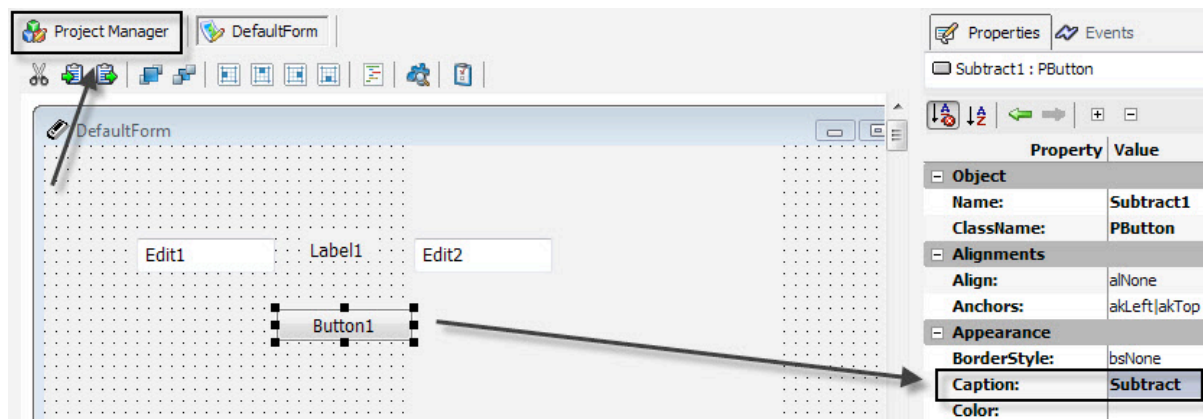


Figure 229: Project Manager

- In the **Project Manager**, double click the **PButton** to create an **OnClick** event. This event will be used to trigger actions once this button is clicked.

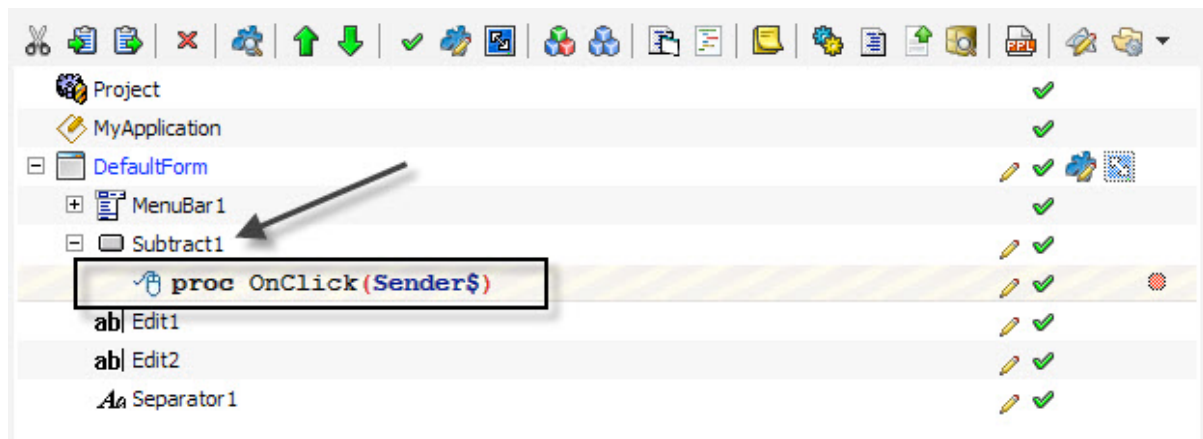


Figure 230: OnClick event

- Drag a **PVariable** onto the **OnClick** event. This will create variable definition that can be used to program actions in the program.

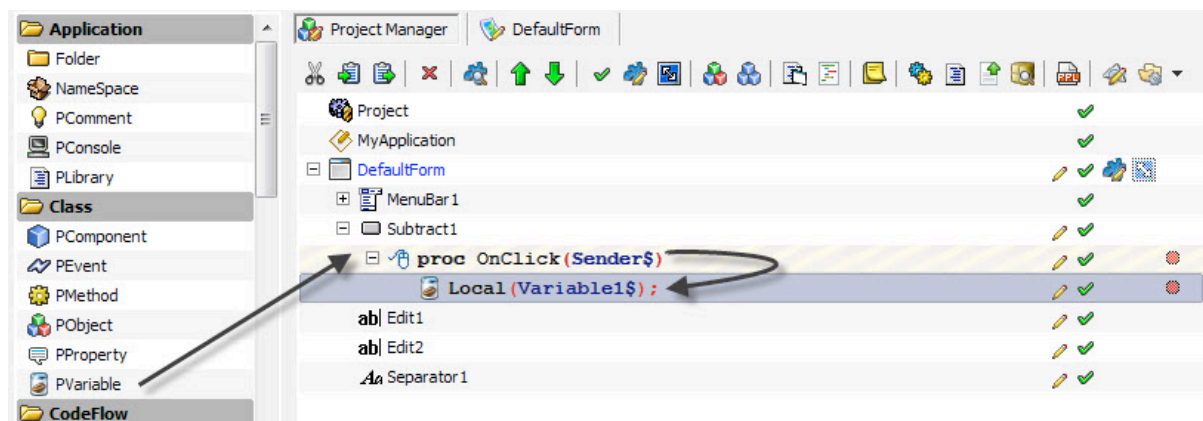


Figure 231: Drag PVariable



- Once you have declared a variable, you need to assign it to something. This is accomplished in PIDE by dragging the variable declaration and dropping it once more on the **OnClick** event while holding the **ALT** key on the keyboard.

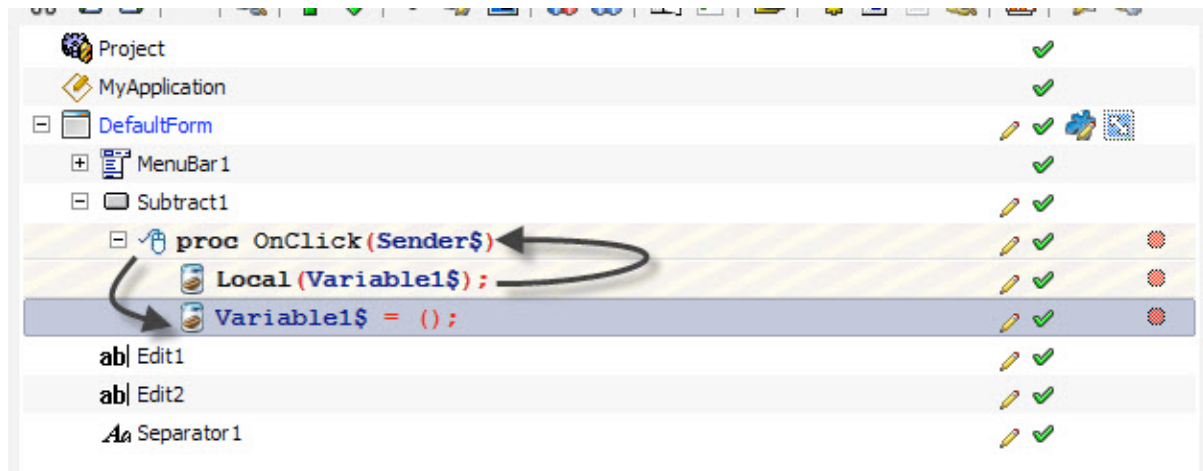


Figure 232: Alt+drag on OnClick

- Click the newly made variable and drag the **PEdit** box on its **Expr** property. In the **Code Completion** box that appears, click **Text** option to select **Edit1.Text**

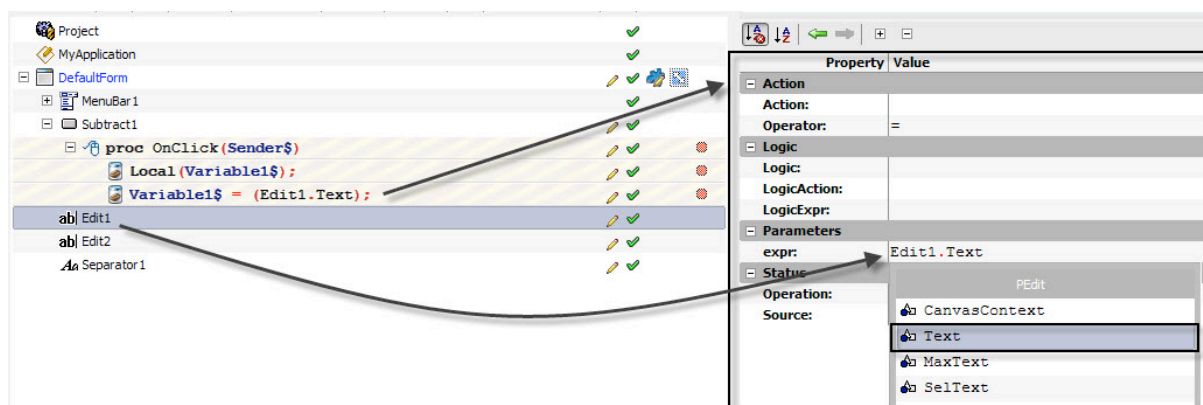


Figure 233: Drag PEdit to expr

- The variable created above will now contain the text entered in the **PEdit1** box. Just like the couple of steps given above, we will create a variable for the other **PEdit** box.

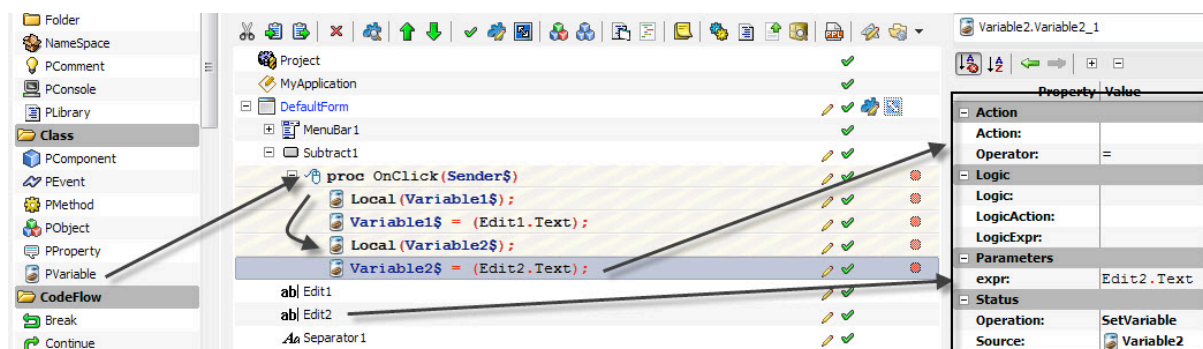


Figure 234: Create New variable

- After creating variables to store the value of both our **PEdit** objects we will check that the value of our first number is smaller than the value of the second number. To accomplish this, we will use an **IF** condition. This condition will check if the value of the first variable is larger than the second variable, it will display a message alerting a user to input a smaller value. For creating the **IF** condition, start by dragging and dropping the **PEdit1** box object to the **OnClick** object.
- The above action will initiate **Code Completion** window. Select the **Text** option in this window.

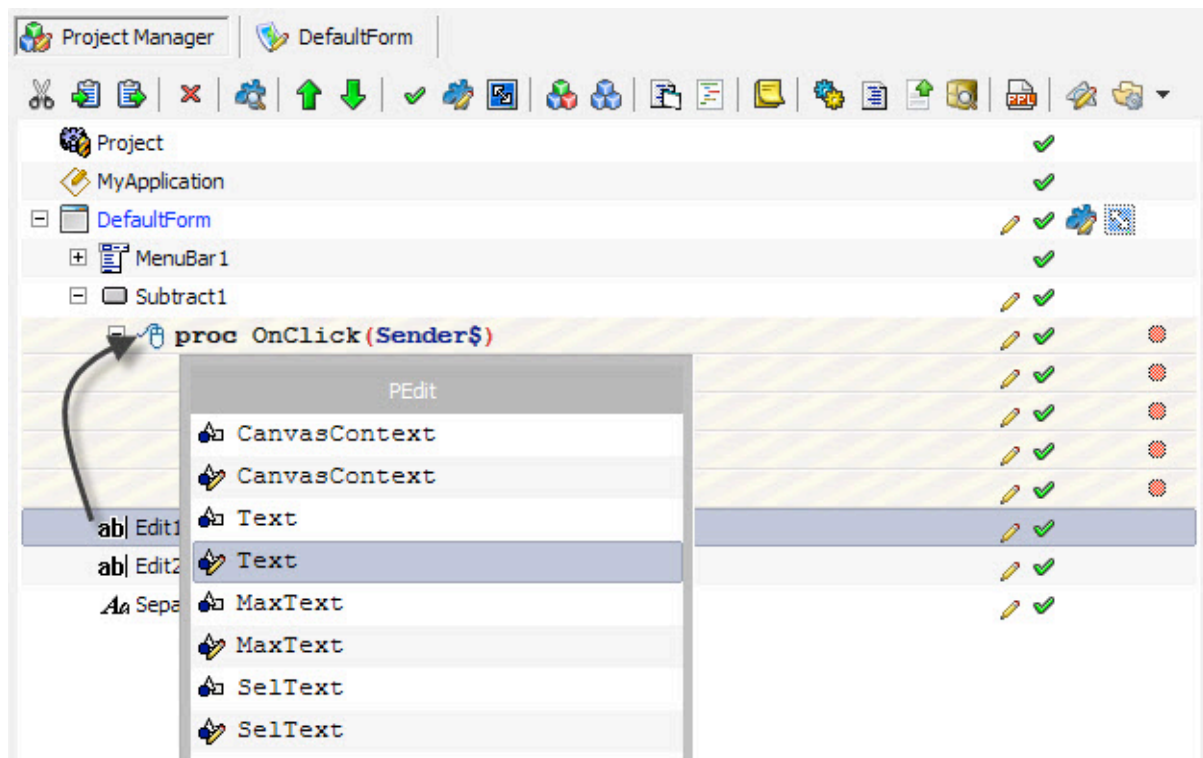


Figure 235: Select Text option

- Select the newly created statement and press **Ctrl+I** while it is selected. This will change this statement to an **IF** statement.

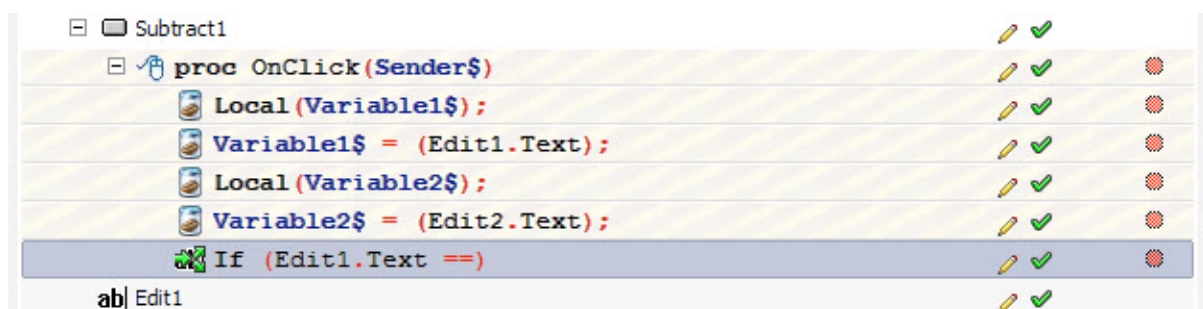


Figure 236: Press Ctrl+i for IF statement

- Now, while the **IF** statement is still selected, change its **LogicExpr** property to **<Variable2\$**. Doing this will check if the value entered in the **PEdit1** box is smaller than the value that is stored in **variable2** that holds the value stored in the **PEdit2** object.

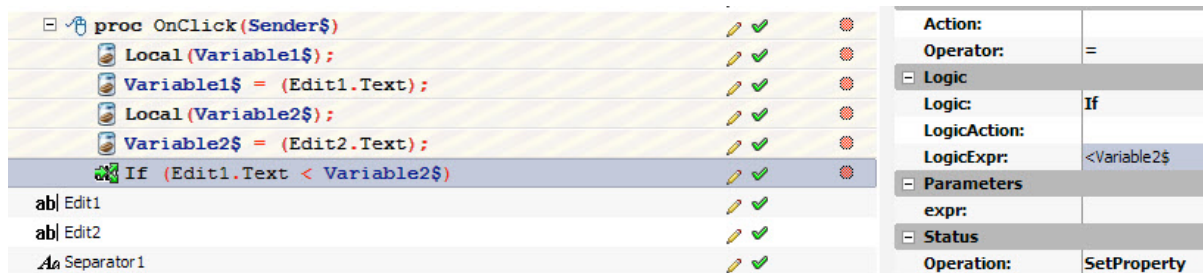


Figure 237: Change LogicExpr property

- Drag a PVariable to the **OnClick** event to create a variable declaration and then drag the variable to the **OnClick** event while holding the **ALT** key to create a new variable. Once this variable is created, drag and drop it on the **IF** statement so that it comes under it. A sure sign to make sure that you have put this variable under the **IF** statement is to check for the small - expansion mark on the left of the **IF** statement.

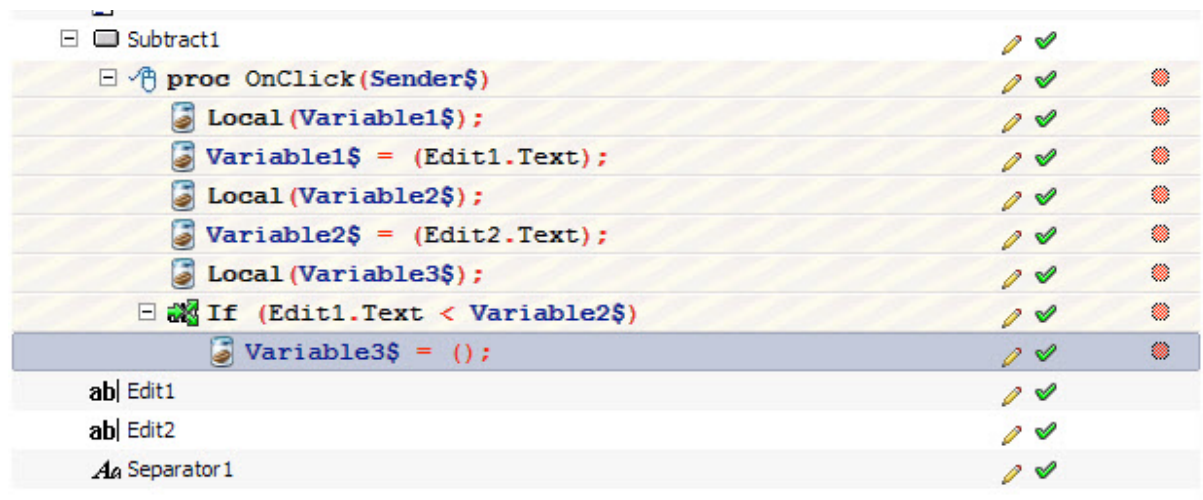


Figure 238: Create new variable

- Once you have this new variable in place. You need to change its **Expr** property to print "Sorry! Number1 cannot be smaller than Number 2". This statement will be executed if the first number is found to be smaller than the second number.

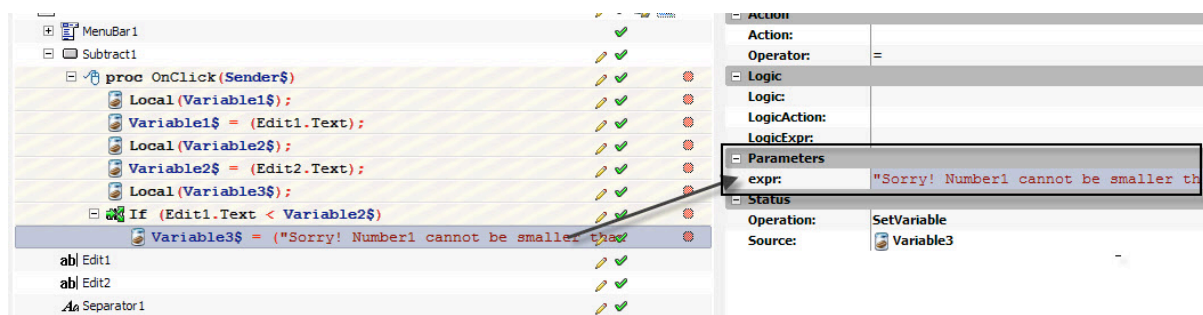


Figure 239: Change expr property

- For ensuring that the button press generates the subtraction of the two numbers, we will drag the **variable3\$** declaration again to the **OnClick** event. This will create another variable with the same name. Go to the **Expr** property of this variable and input **Variable1\$-Variable2\$**. Our newly created variable will hold the value

subtraction results of the two numbers. Drag this variable above the **IF** statement so that it does not interfere with its value.

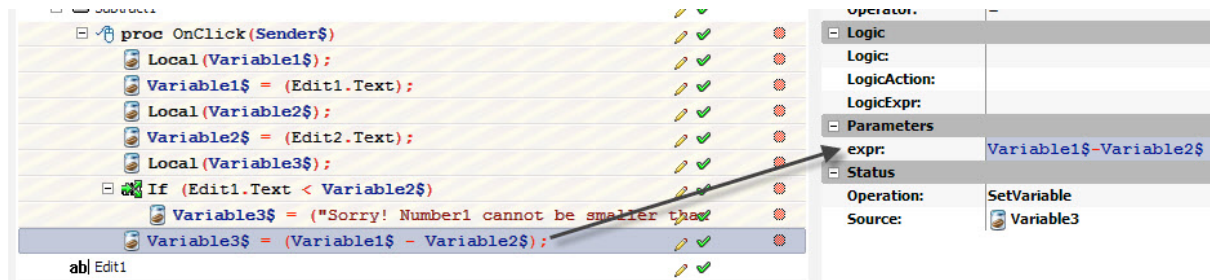


Figure 240: Change Expr property

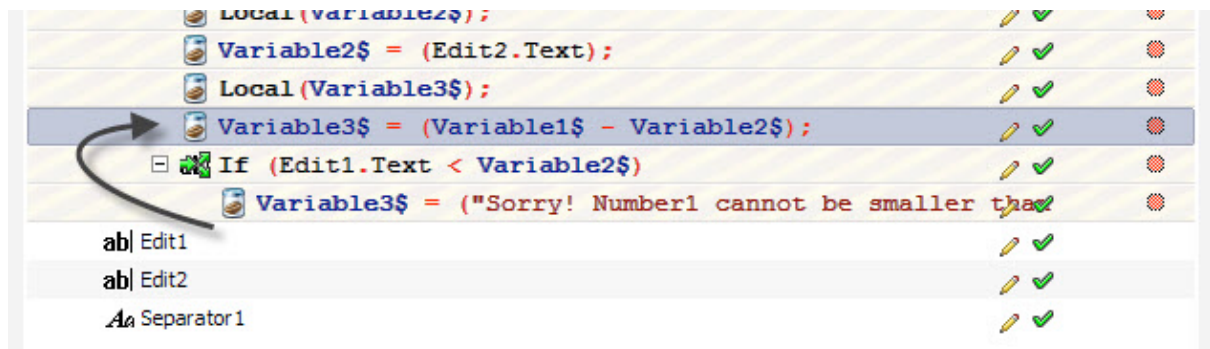


Figure 241: Drag and drop above IF

- Click on the **OnClick** event and press **Ctrl+Space** to initiate **Code Completion** window. Select **ShowMessage()** from it.

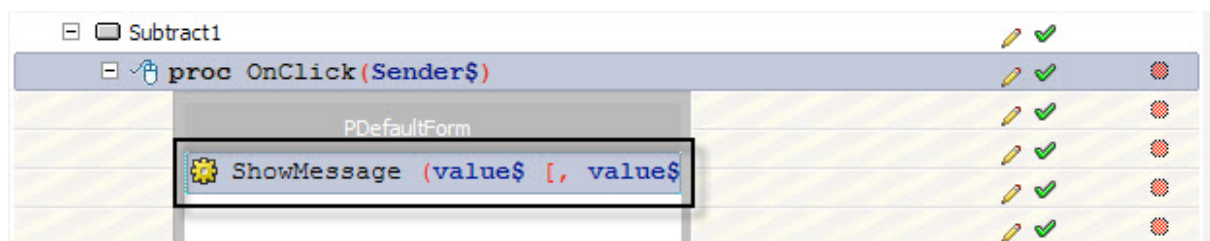


Figure 242: ShowMessage

- In the value property of **ShowMessage**, write **Variable3\$**. This will print the value that is contained within Variable3. If the condition of the first number being smaller than the second number is found to be true, Variable3 will have the alert but if it is not found to be true, variable3 will contain the result of subtraction of the first and the second number.



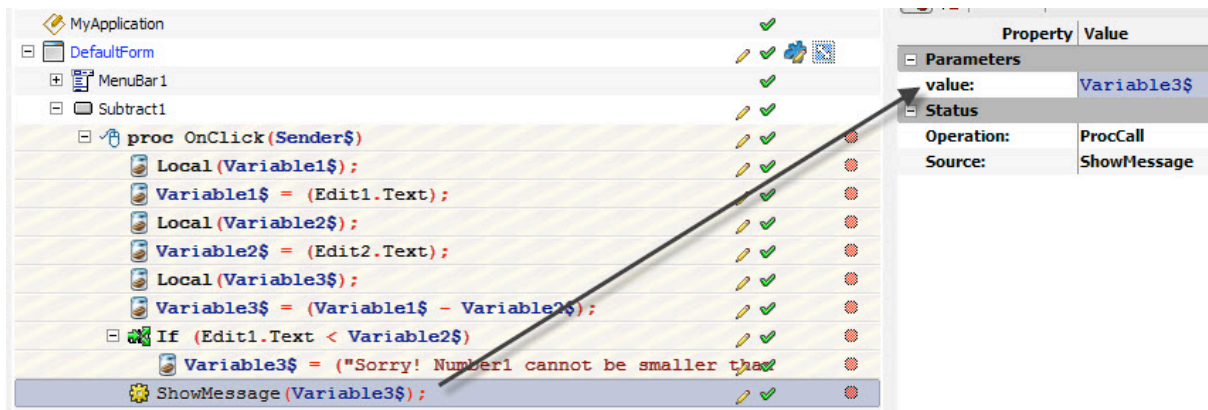


Figure 243: Change Value property

- After confirming that your program matches the below given screenshot, save your program and run it.

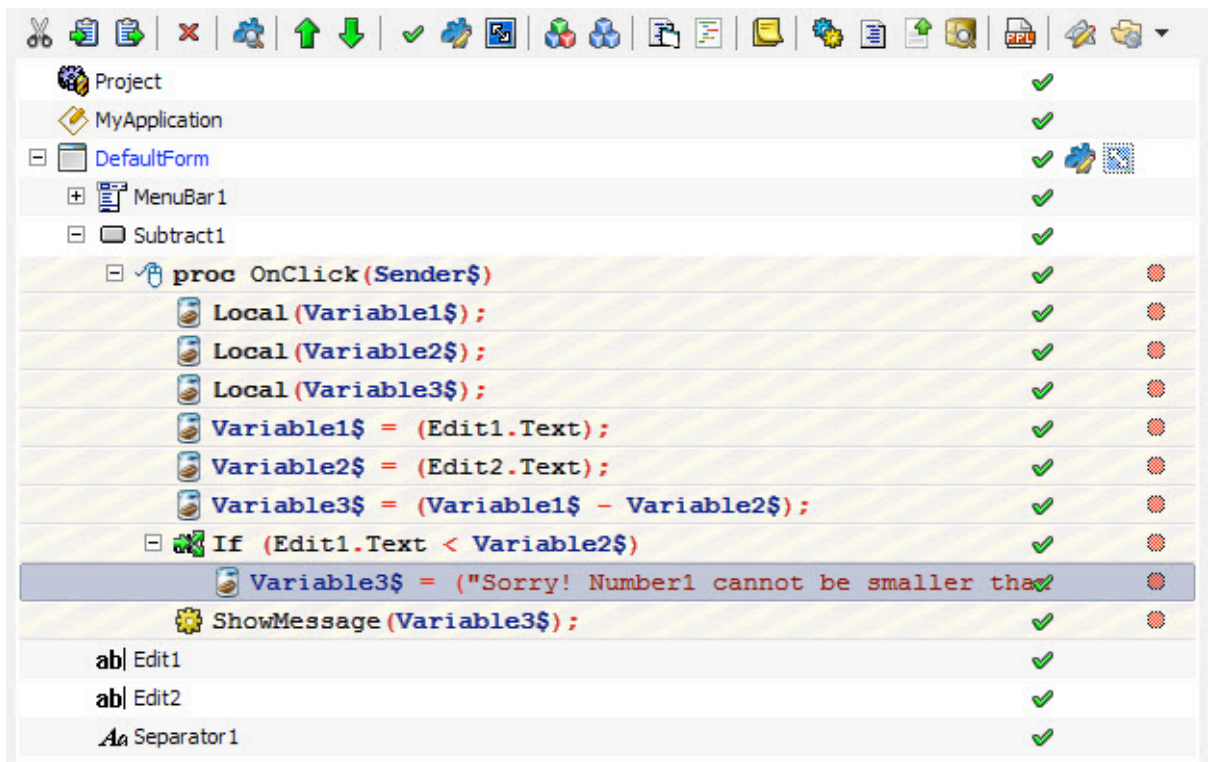


Figure 244: Project Screenshot

- Press **Ctrl+S** to save the program. Alternatively, you can also go to **File>Save As** to save your project to wherever you want.

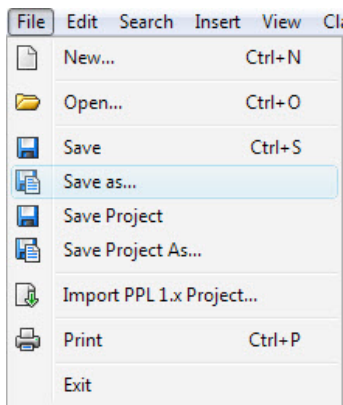


Figure 245: Save your project

- Now, press **F5** and check your program for results!

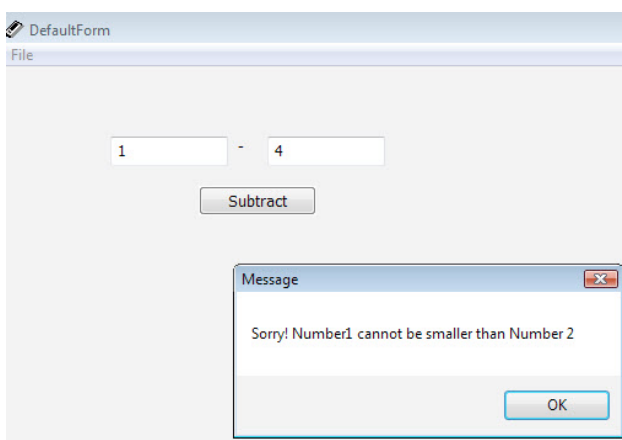


Figure 246: Output



## Loops and Conditions: Else Condition

The **Else** object is used to specify an alternate result or outcome of the **IF** condition. By using a combination of **IF** and **ELSE** object, users can completely evaluate their results in event any event.

Possibly one of the easiest to use components of PIDE, the **ELSE** statement is something that is only useable with the **IF** condition. To explain fully, a statement in the **IF** case is executed only if a condition is evaluated to be true but the else case is executed when the condition in an **IF** case if evaluated to be false. Rather than having two **IF** cases that would govern the execution or non execution of a statement, it is far better a programming practice to use in **ELSE** condition.

The example given below inputs two integers and gives an alert if the first integer is greater than the second integer or otherwise. In this example, an **IF** case will evaluate the condition and an else case will return the output if that condition is not fulfilled.

- Start with creating a **Desktop Form** Project. To do so, press **Ctrl+N** or go to **File>New** and select **Desktop Form** project in the **Select New Project Type..** window.

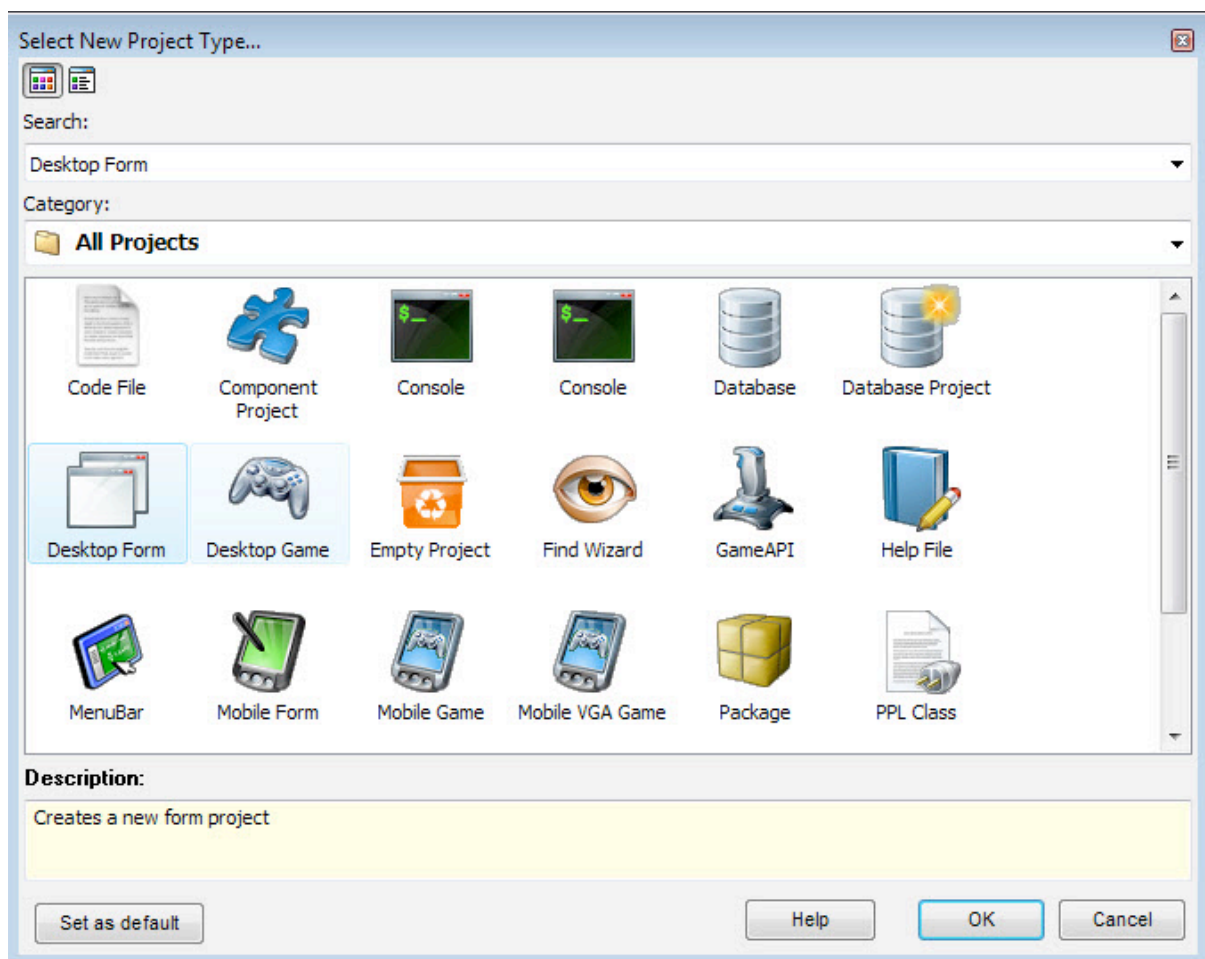


Figure 247: Create new project

- In the **Default Form Project Manager**, double click the **Default Form** to initialize the Form editor and place two **PEdit** boxes with three **PLabel**s and one **PButton**; like shown in the figure below. To create the aforesaid components on the **Form editor**, click on the component you are looking for in the **Components Pane**. After selecting the component, click on a place on the form editor to place that component there. Refer to the screenshot below to configure your components.

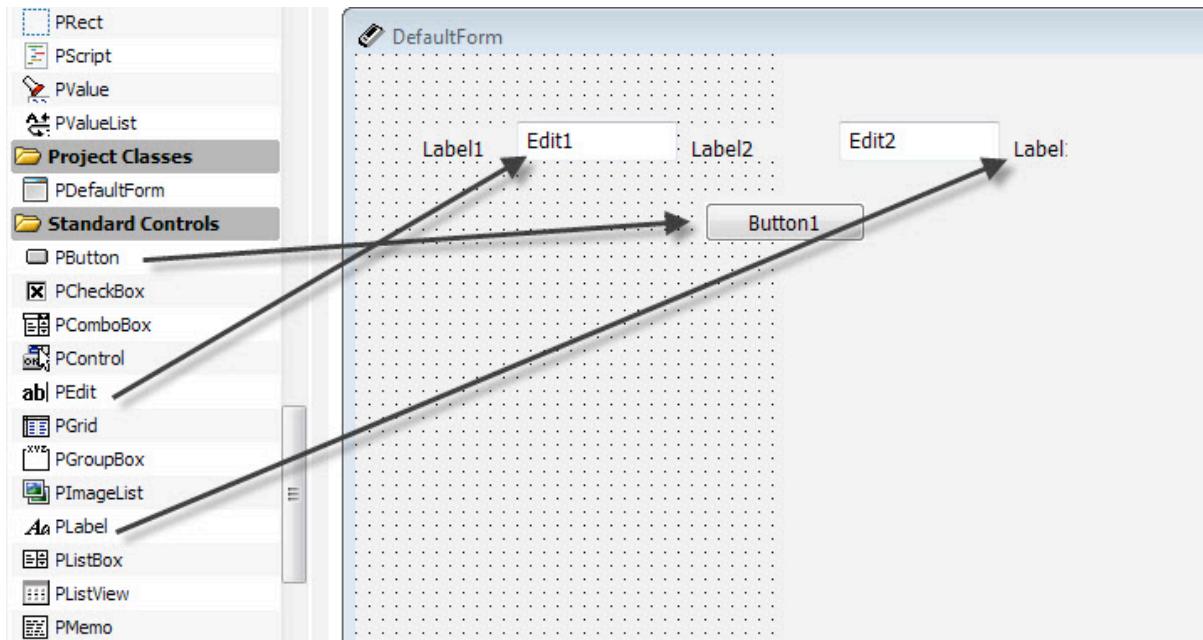


Figure 248: Place coponents

- After you have all the components placed on the form, you will be required to change their **Caption** property to something that makes sense. To change the **Caption** property of a component, click on the component and go to its properties in the properties panel on the right. In properties, find the **Caption** property and change it to anything you want. We have changed our **PLabel** to following:

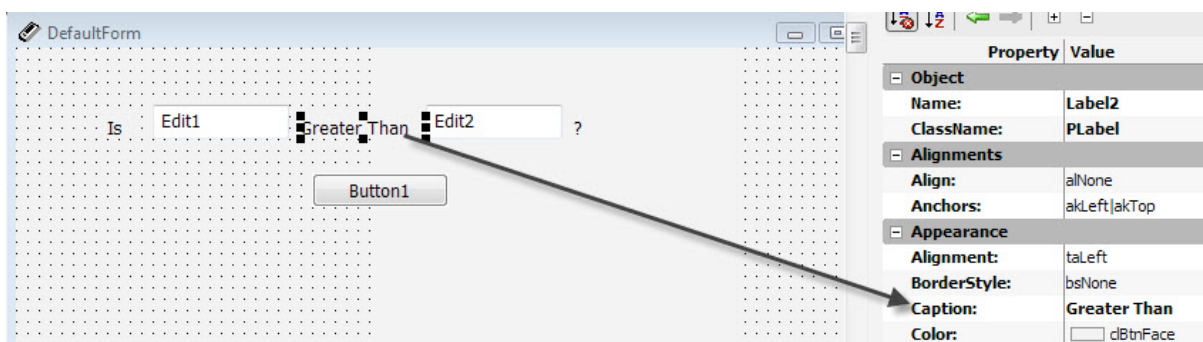


Figure 249: Change Caption property

- Once you have made the required changes to the form, revert back to the **Project Manager** by clicking the **Project Manager** button on the top left of the screen.

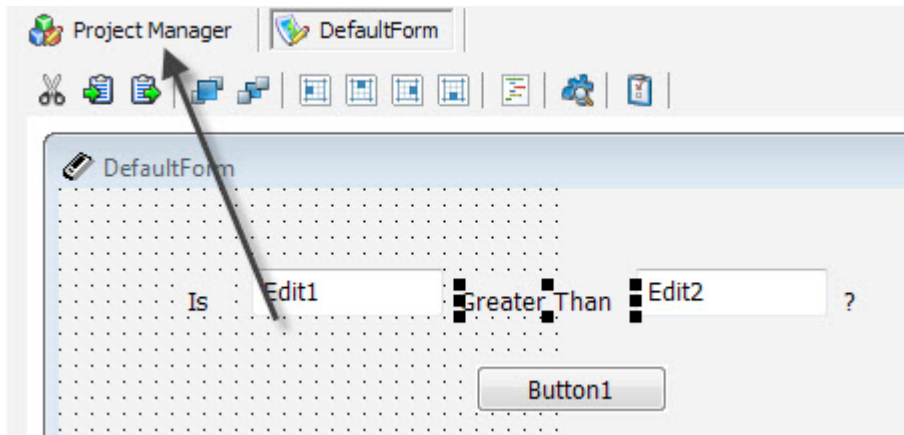


Figure 250: Project Manager

- In the **Project Manager**, select the **PButton** and double click it to create an **OnClick** object. This **OnClick** event will be used to process actions on the click of the button.



Figure 251: Create OnClick event

- Now, we will be required to create variables that would be used to store and process values for the program. Like every other programming language, PIDE also requires you to declare variables before storing values in them. Drag a **PVariable** to the **OnClick** event to create a variable definition.

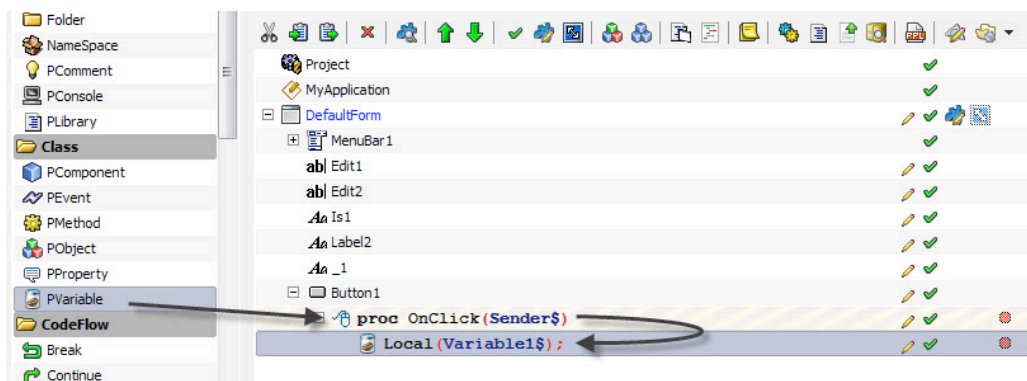


Figure 252: Drag PVariable

- Now, to create a variable, drag the newly created variable definition back to the **OnClick** event while holding the **ALT** key. This would create an empty variable ready to hold information.

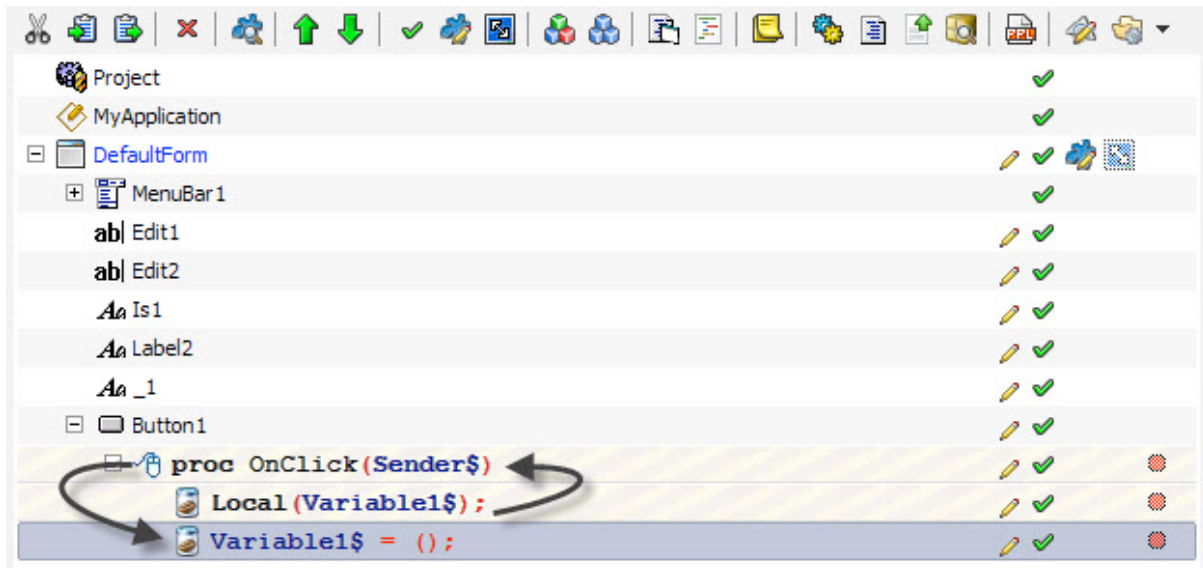


Figure 253: Create a new variable

- While the new variable is still selected, drag the **PEdit1** object to its **Expr** property. This will initiate the code complete window.

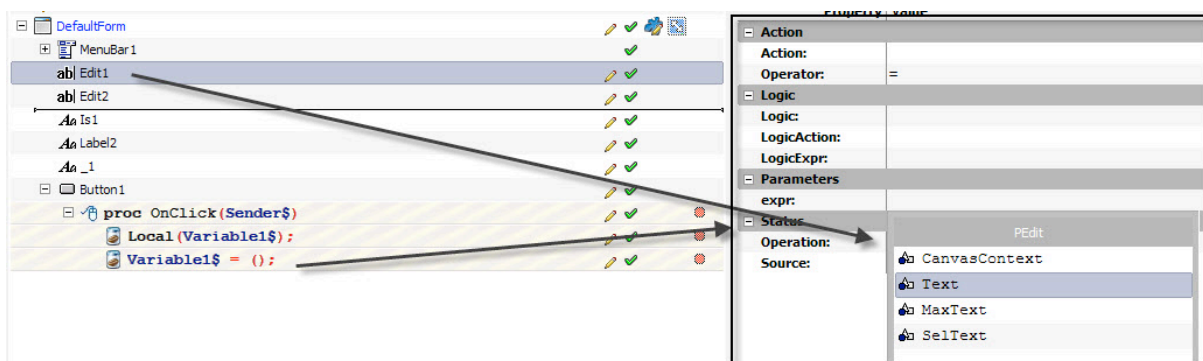


Figure 254: Drag PEdit to Expr property

- Select **'Text'** value from this window so that it now contains **Edit1.text**.

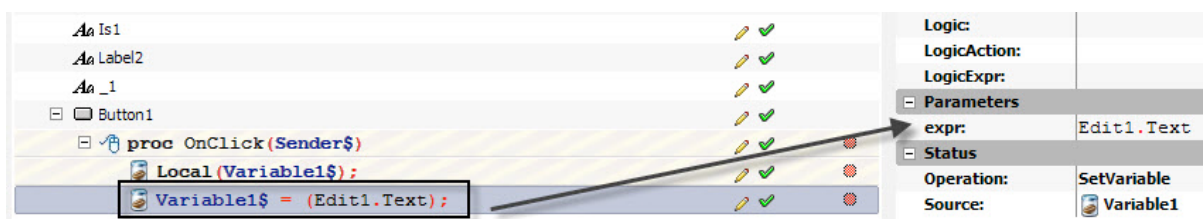


Figure 255: Change Expr Property

- Just like the last variable, create another variable by first declaring its definition and then assigning value to it.

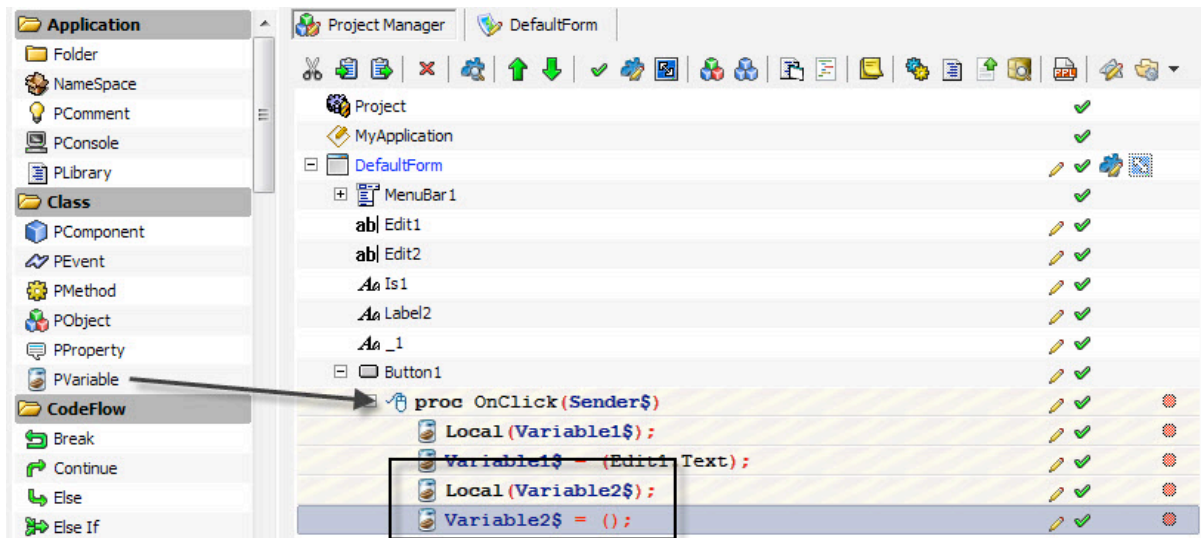


Figure 256: Create variable

- In this variable, drag **PEdit2** to the **Expr** property and select the **Text** property so that this variable contains the value of second input box.

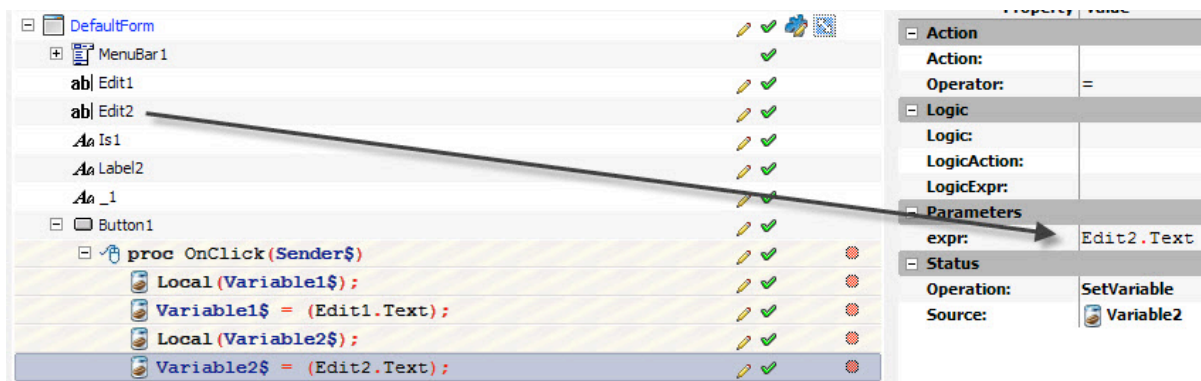


Figure 257: Change Expr property

- Once both the variables for the values are created, we now have to work upon the **IF** condition that would check if a number is greater than the other or not. For doing this, drag the **PEdit1** object to the **OnClick** event; this initiates a code complete window. In the code complete window, select the **Text** property.



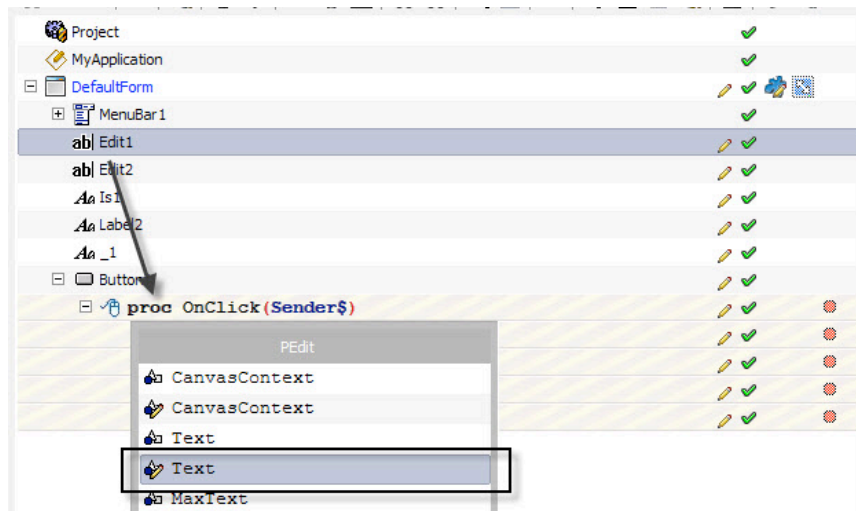


Figure 258: Select Text property

- Click on the newly created statement, and press **Ctrl + I** to change it to an **IF** statement.

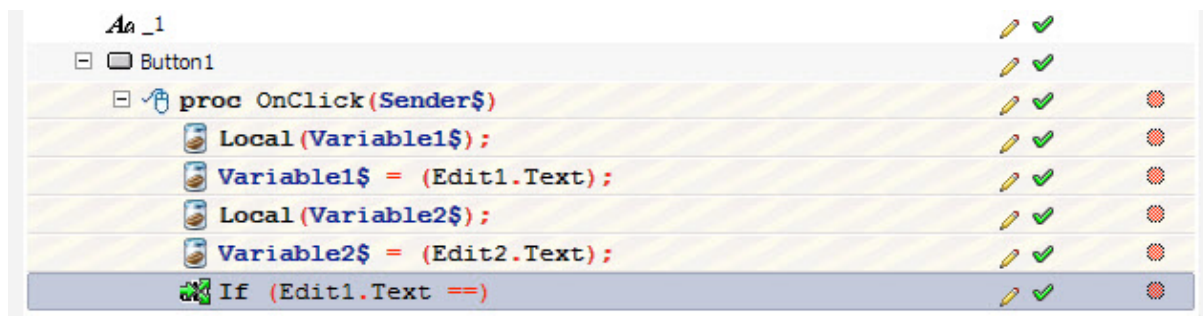


Figure 259: Press Ctrl+I to create IF statement

- Click the newly created IF statement and go to its **LogicExpr** property. Here, write **<Variable2\$**. This would trigger the **IF** statement if the text in **PEdit1** is smaller than text in the **Variable2\$**.

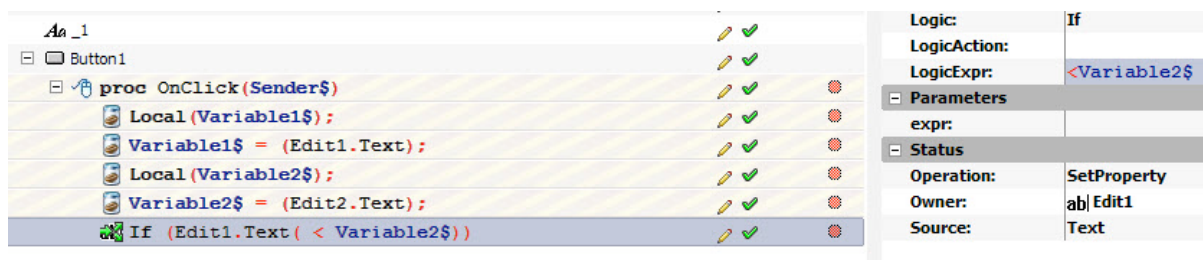


Figure 260: Change LogicExpr property

- Click on the **IF** statement and press **Ctrl+Space** bar. In the **Code Completion** box that appears, select **ShowMessage** option.



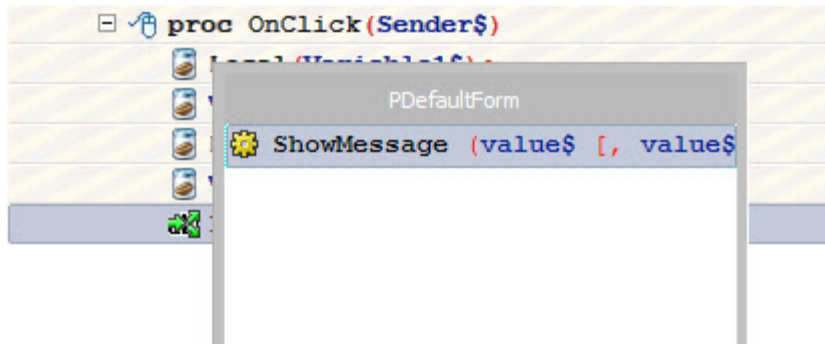


Figure 261: ShowMessage

- In the value property of **ShowMessage**, enter “*Number 1 is smaller than Number 2*”

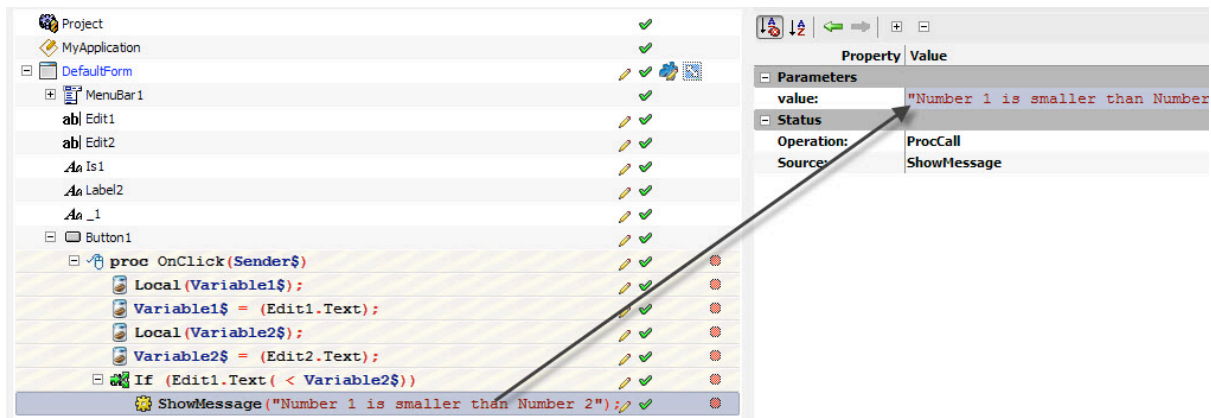


Figure 262: Change Value property

- Now, drag an **ELSE** object to the **OnClick** object and press **Ctrl+Space** Bar while it is selected. Select **ShowMessage** form the **Code Completion** window.

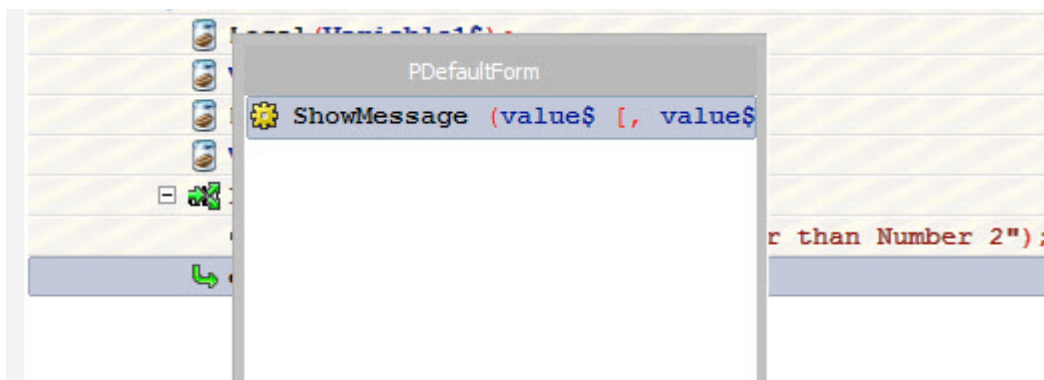


Figure 263: Use ShowMessage

- Select the newly created **ShowMessage** object and change its **Value** property to “*Number 1 is greater than Number 2*”

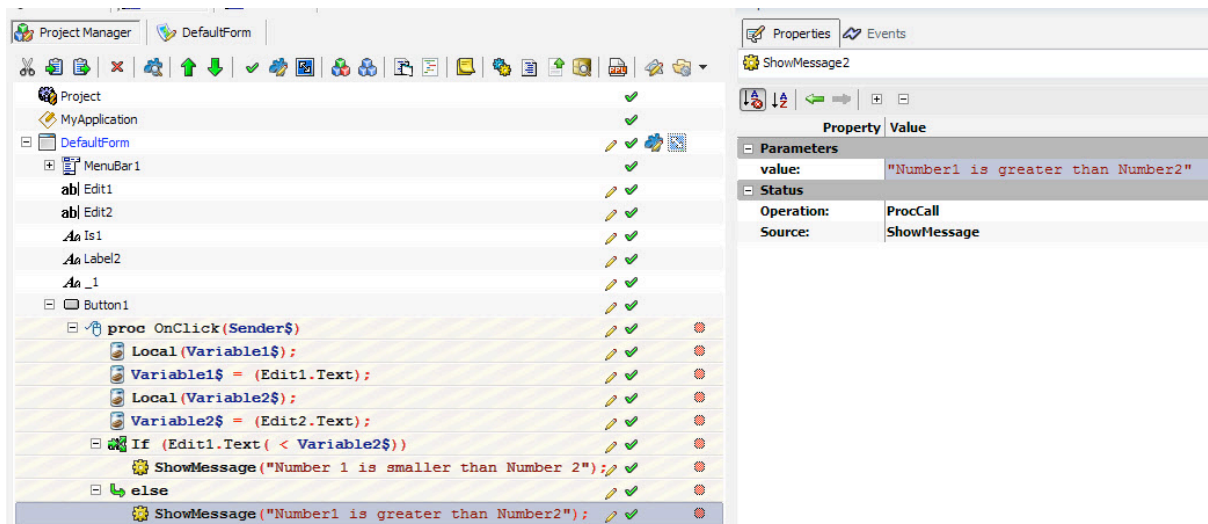


Figure 264: Change value property

- The application is complete! Save the application by pressing **Ctrl+S** or going to **File>Save As..** and save it in a folder of choice.

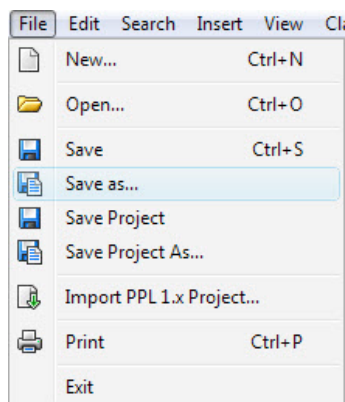


Figure 265: Save Project

- Press **F5** to run the project and see the output:

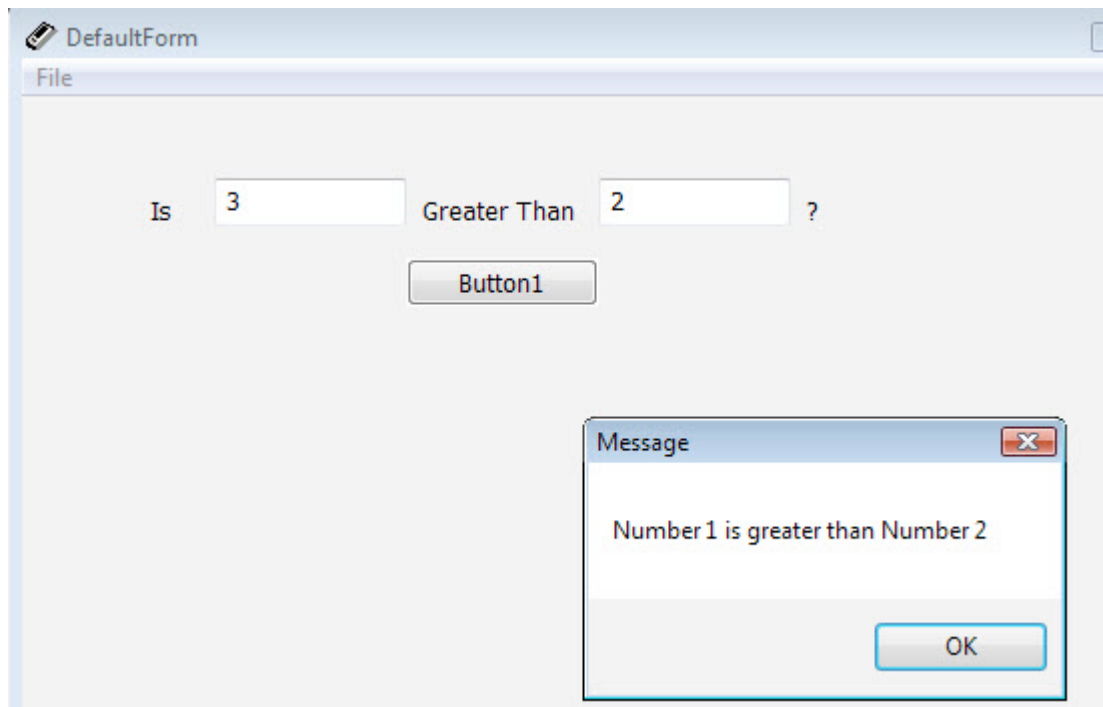


Figure 266: Run Project

## Loops and Condition: Else-If Condition

In the above sections, we have seen that a certain condition can be evaluated with the help of **IF** condition and if the condition is not true, we can use the **ELSE** condition to take care of the other situation also. While a combination of **IF** and **ELSE** works most of the times, we would still require a conditional statement that can be used to specify more than one outcomes of a condition.

There are many real world situations where one needs to have more than one outcome or choice for one statement. The given example allows a user to input an abbreviation of 'days of the week' and returns its full form. Here we will simply check for the various outcomes of a single user entry and output the result according to that.

- Start with creating a **Desktop Form Project**. To do so, press **Ctrl+N** or go to **File>New** and select **Desktop Form** project in the **Select New Project Type..** window.

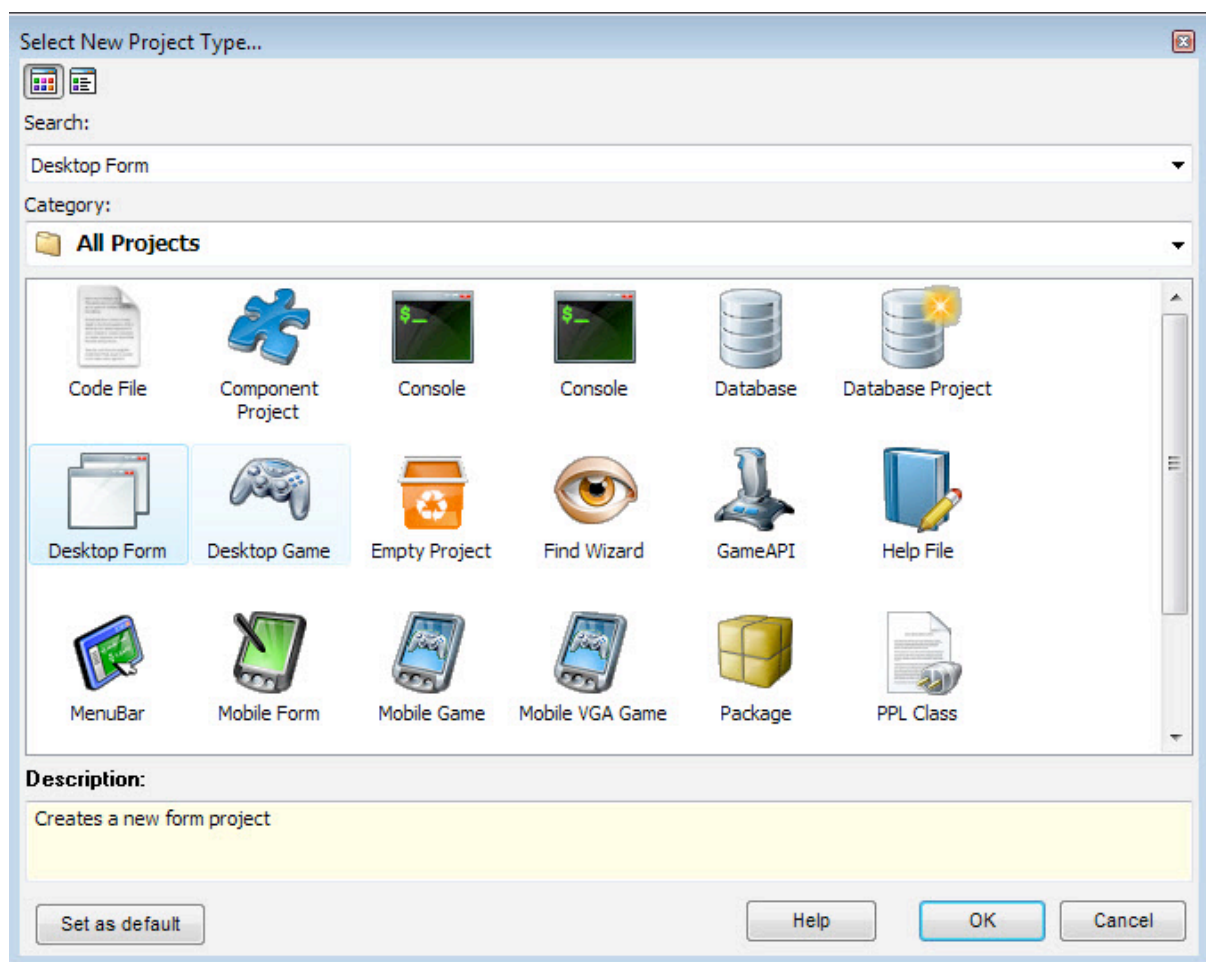


Figure 267: Create new project

- In the **Default Form Project Manager**, double click the **Default Form** to initialize the Form editor and place one **PEdit** object with one **PLabel** and one **PButton**; like shown in the figure below. To create the aforesaid components on the **Form editor**, click on the component you are looking for in the **Components Pane**. After selecting

the component, click on a place on the form editor to place that component there. Refer to the screenshot below to configure your components.

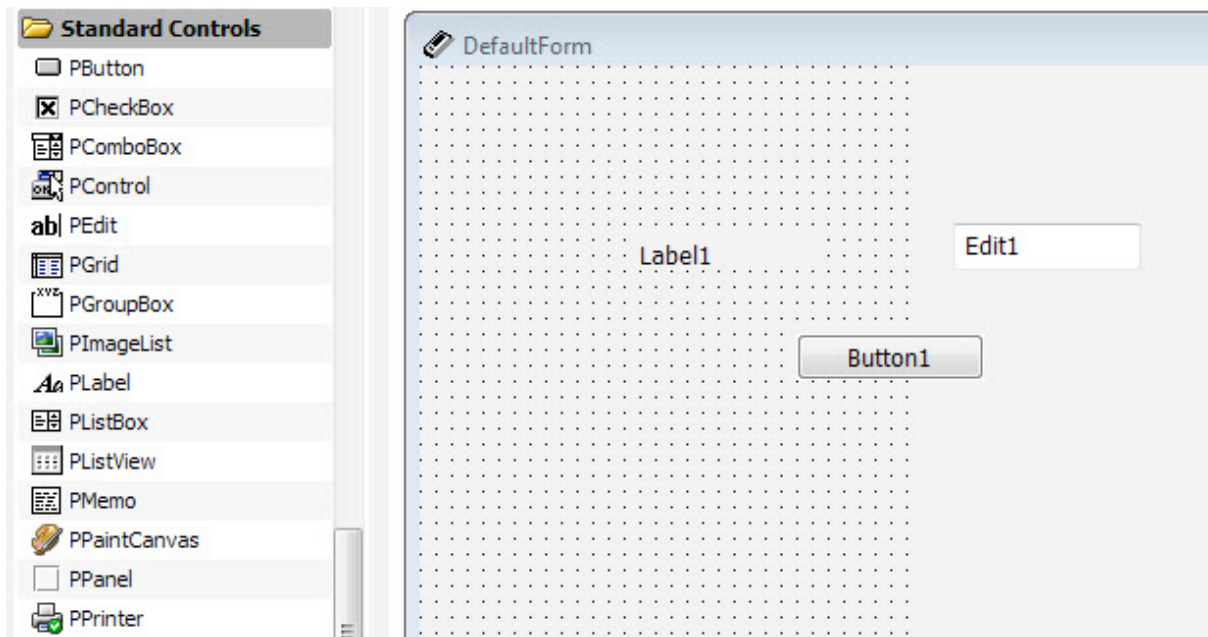


Figure 268: Place Components

- For viewing purposes, we will now change the **Caption** property of **PLabel1** to **Enter an abbreviation of any day in a week**

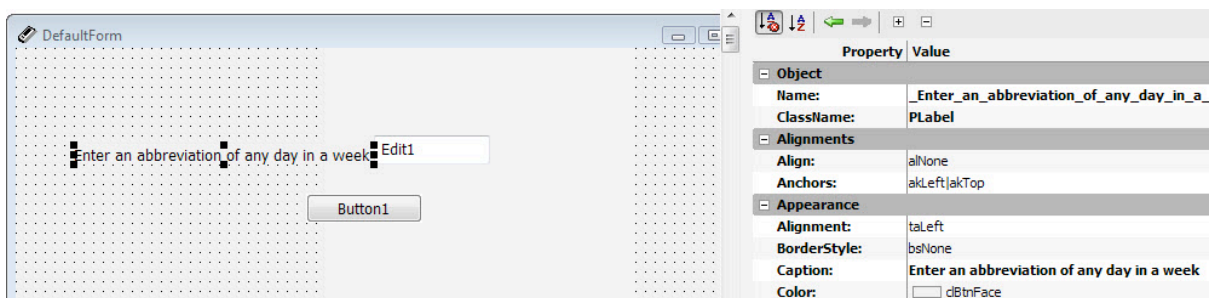


Figure 269: Change Caption

- Return to the **Project Manager** and double click the **PButton** to create an **OnClick** event that will initiate the actions that are to be performed.

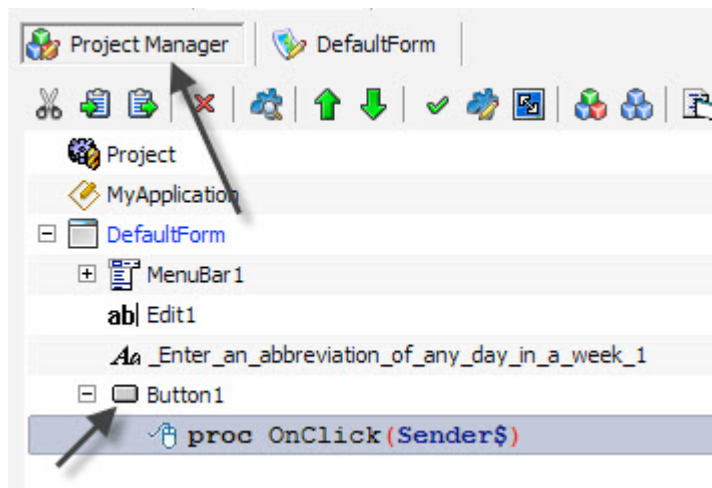


Figure 270: Create OnClick event

- Now, we will be required to create variables that would be used to store and process values for the program. Like every other programming language, PIDE also requires you to declare variables before storing values in them. Drag a **PVariable** to the **OnClick** event to create a variable definition.

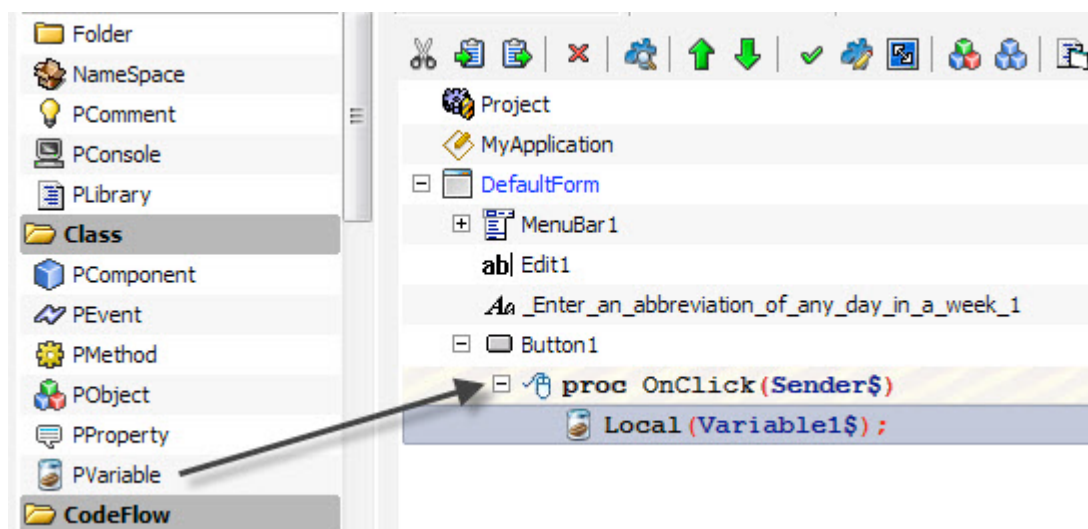


Figure 271: Drag PVariable to OnClick

- To create the variable, drag and drop the variable definition to the **OnClick** event while holding the **ALT** key on the keyboard. This will create a variable ready to hold values.



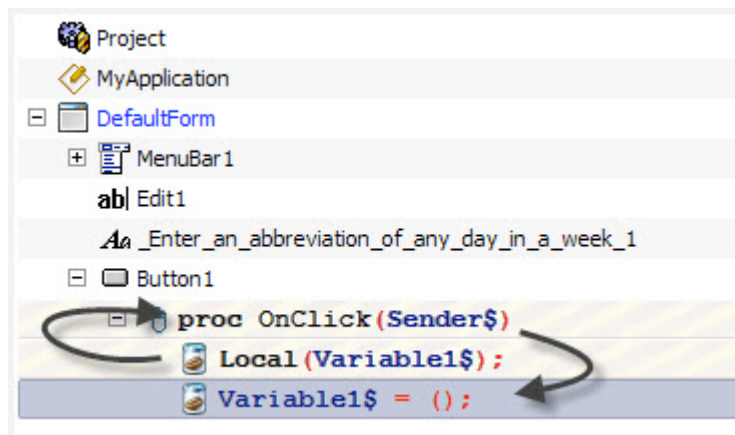


Figure 272: Drag variable

- Drag the **PEdit1** object to the **Expr** property of this newly created variable to associate it with the **PEdit** box. This action will initiate a **Code Completion** window; select **Text** option in the window as it will ensure that our variable holds the value that is being input in the **PEdit** Input box.

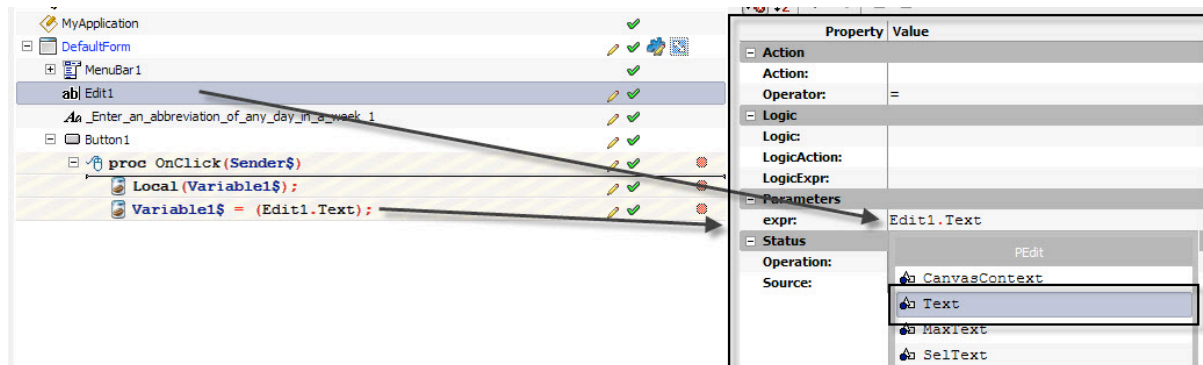


Figure 273: Drag PEdit to expr property

- Now to check the entered value in the **PEdit** box we will have to have an **IF** condition. To do this, drag the **PEdit1** object on the **OnCreate** event and select **Text** property in the **Code Completion** window that appears.

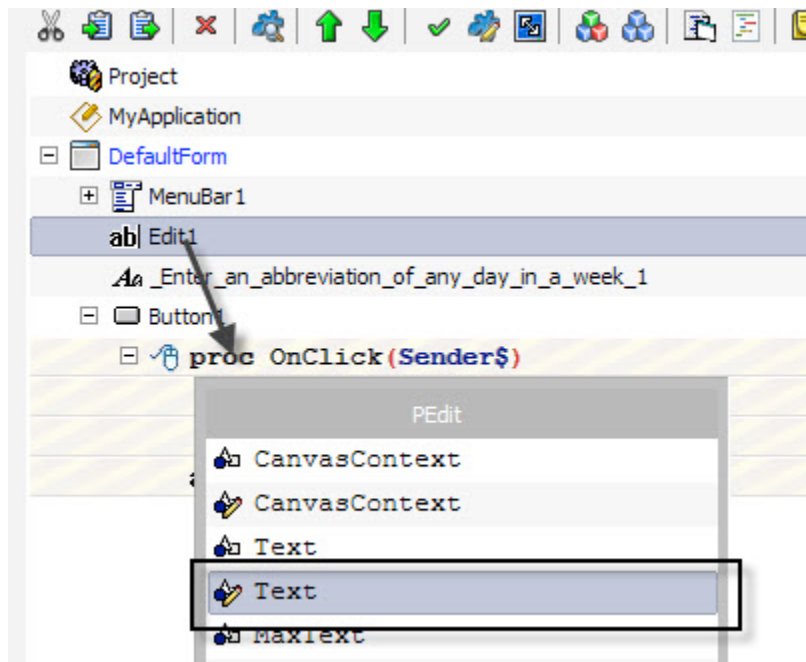


Figure 274: Drag PEdit and select Text

- Click on the newly created statement and press **Ctrl+I** while it is selected. Doing this would change the statement to an **IF** condition.

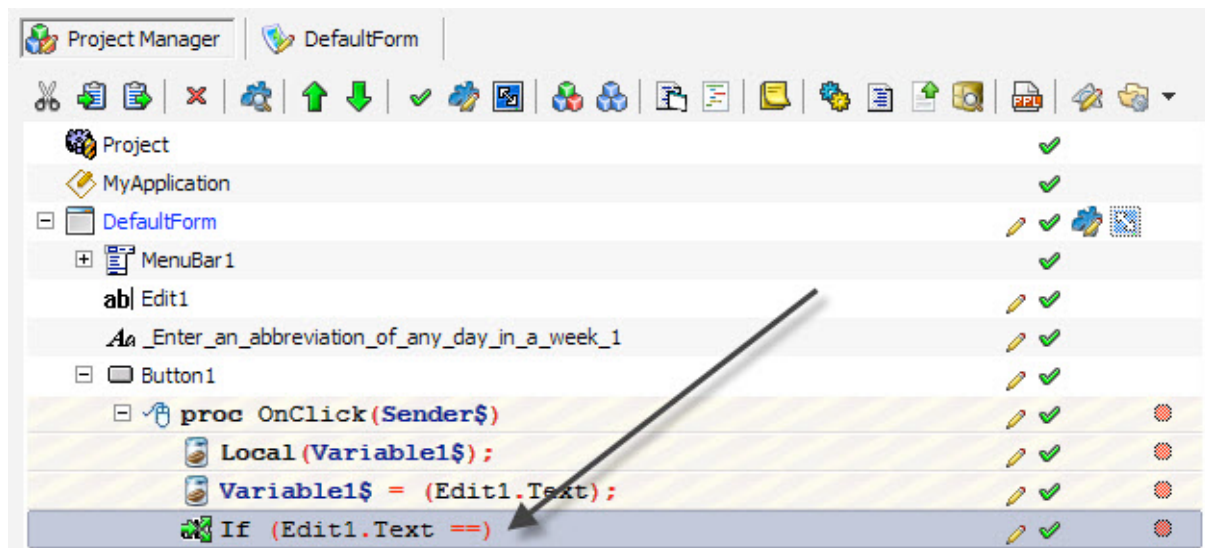


Figure 275: Press Ctrl+I for IF statement

- While it is still selected, go to the **LogicExpr** property of our **IF** statement and write *“mon”*.

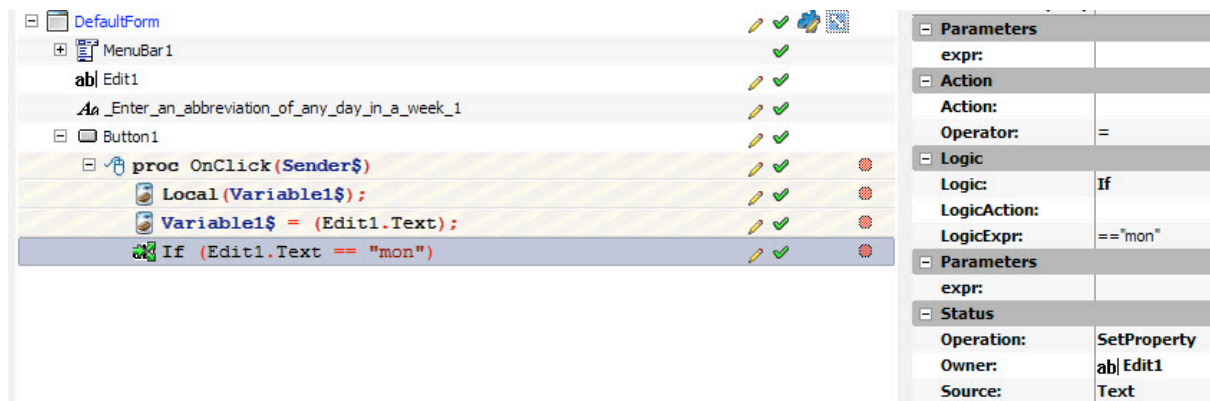


Figure 276: Set LogicExpr

- While the **IF** statement is still selected, press **Ctrl+Space Bar** on the keyboard. This will bring the code complete window. Select **ShowMessage** from it.

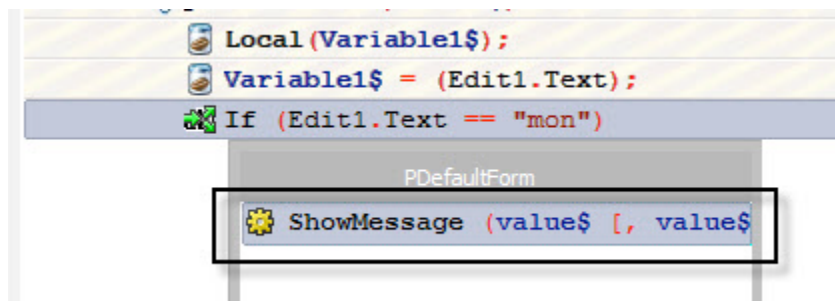


Figure 277: ShowMessage

- Change the Value property of the **ShowMessage** statement to "Monday". This will be displayed if the IF conditions finds "mon" written in the **PEdit** box.

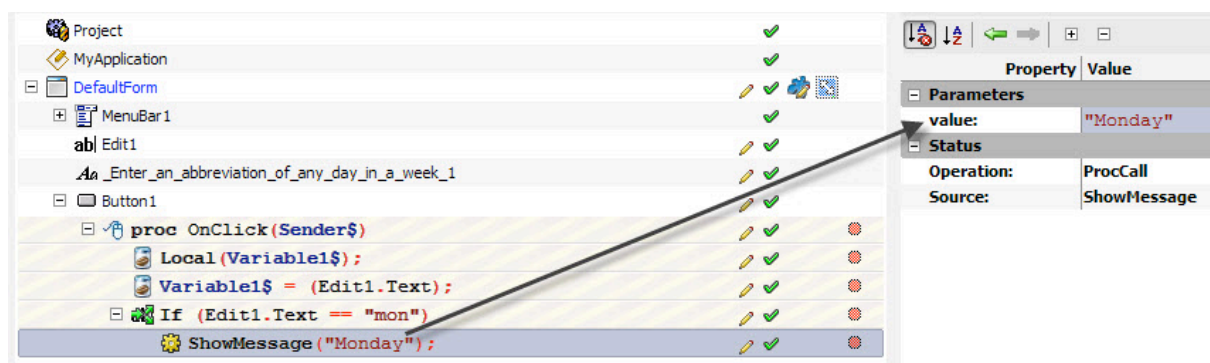


Figure 278: Change value property

- Now we will program for all the other days of the week. Drag an **ELSE IF** object from the **Components Pane** to the **OnClick** event. This will create an empty Else **IF** statement. In its **Expr** property, write **(Variable1\$ == "tue")**. As the **Variable 1\$** already holds the data in the **PEdit** box, if its value is found to be same as "tue" it becomes certain that the abbreviation is for Tuesday.

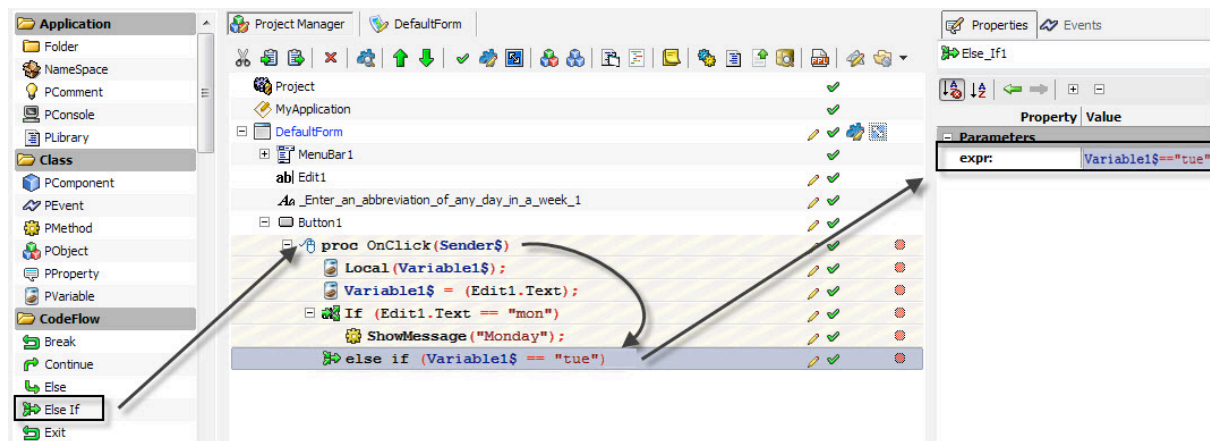


Figure 279: Create Elself Object

- While it is still selected, press **Ctrl+Space** bar on the keyboard to initialize code complete window. In this window, select **ShowMessage** for raising an alert.

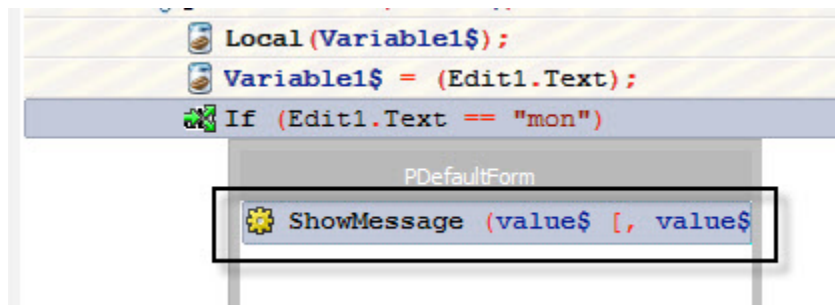


Figure 280: ShowMessage

- In the Value Property of **ShowMessage**, write “Tuesday”.

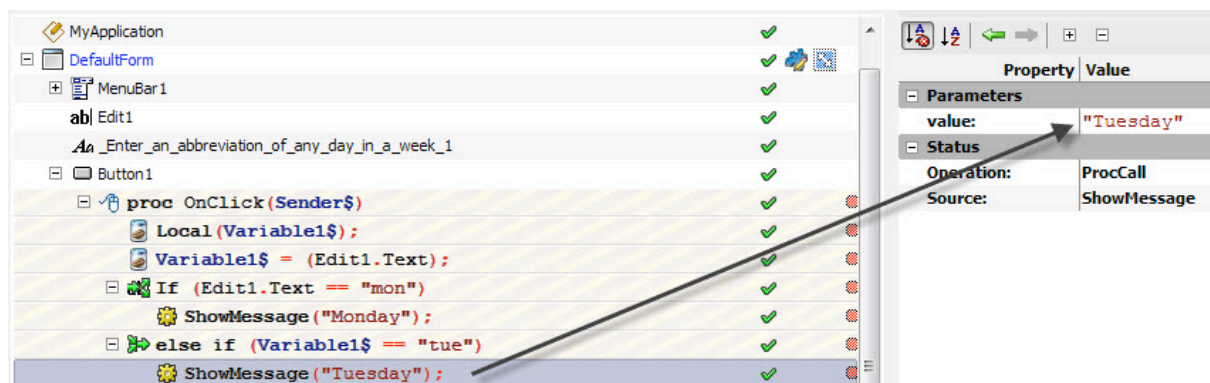


Figure 281: Change value property

- Just like the above **Else IF** statement, we can create statements for all the days of the week.

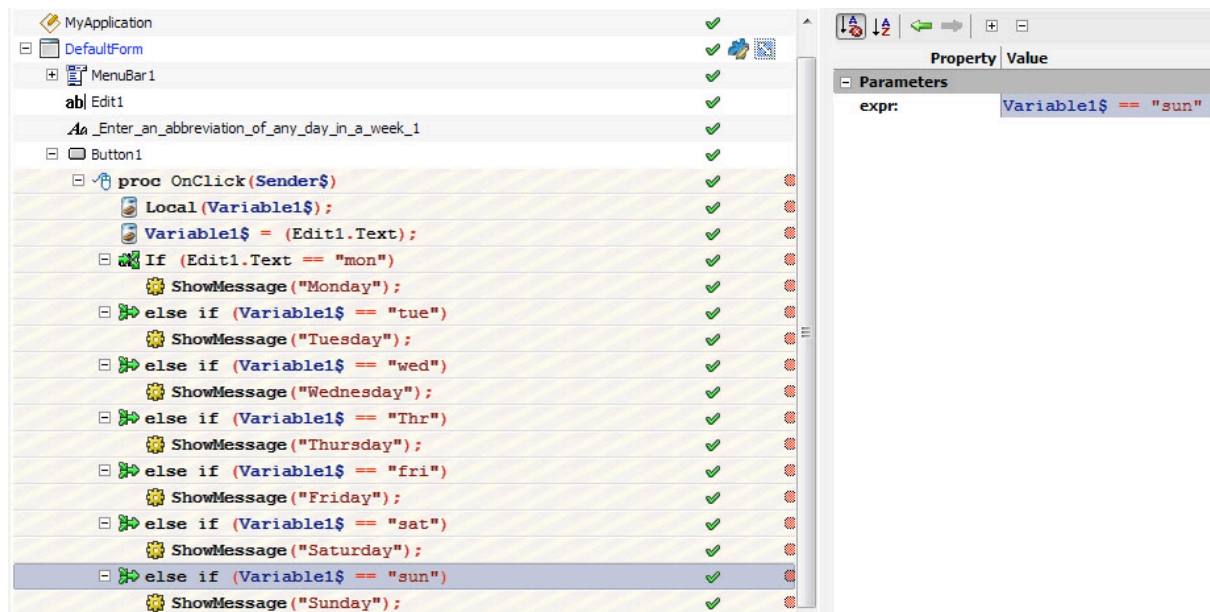


Figure 282: Project screenshot

- The application is complete! Save the application by pressing **Ctrl+S** or going to **File>Save As..** and save it in a folder of choice.

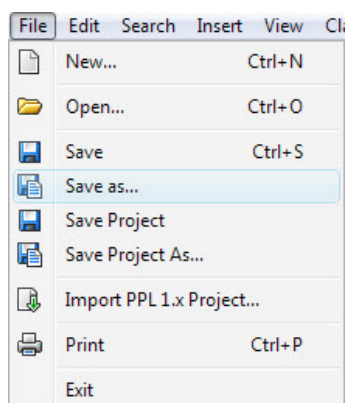


Figure 283: Save Project

- Press **F5** to run the project and see the output:

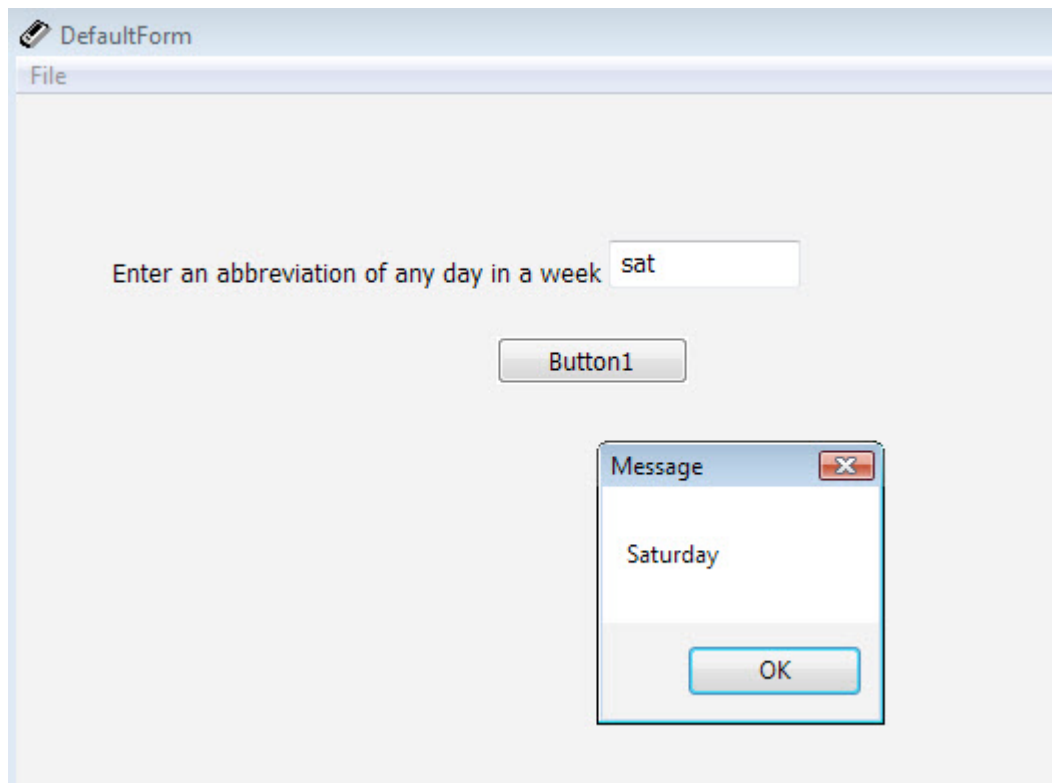


Figure 284: Output



## PVariable

A **PVariable** in PIDE can be used to add a variable to an application. This variable can be used to act as a standard variable with different methods and properties attached to it. In this simple example given below, we will use **PVariable** to declare a variable and then use that variable to store values. The variable will then transfer its value to another variable which will then display it on the screen.

- Start with creating a **Desktop Form** Project. To do so, press **Ctrl+N** or go to **File>New** and select **Desktop Form** project in the **Select New Project Type..** window.

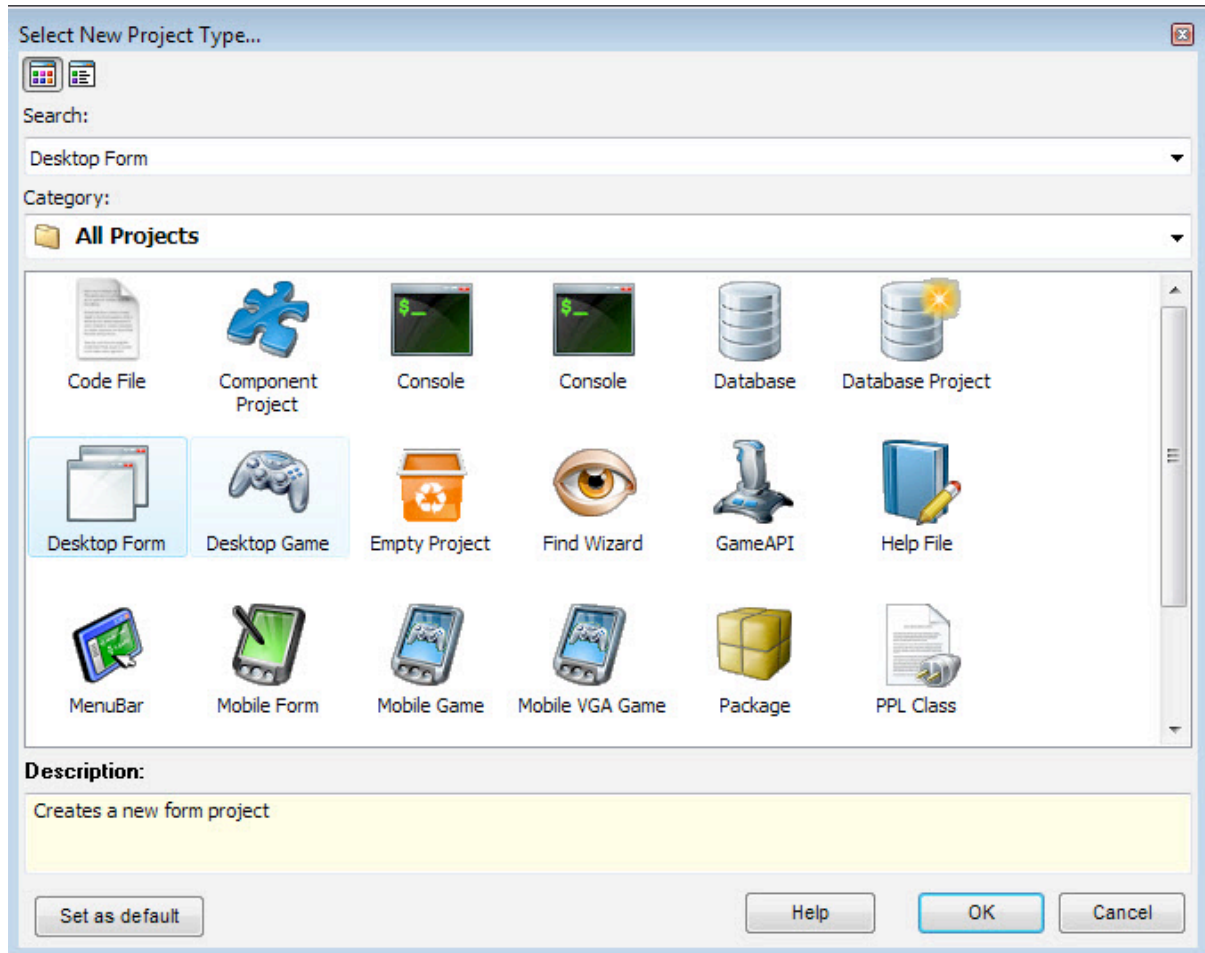


Figure 285: Create New Project

- In the **Default Form Project Manager**, double click the **Default Form** to initialize the Form editor and place one **PEdit** object with one **PLabel** and one **PButton**; like shown in the figure below. To create the aforesaid components on the Form editor, click on the component you are looking for in the **Components Pane**. After selecting the component, click on a place on the **Form Editor** to place that component there. Refer to the screenshot below to configure your components.

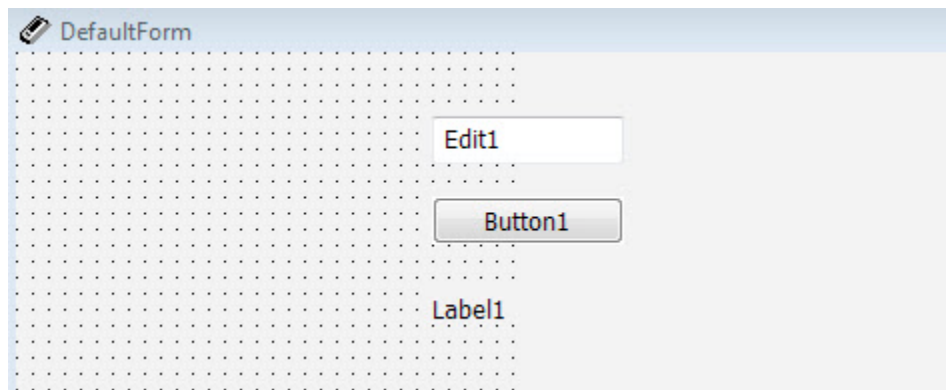


Figure 286: Place components

- Now we can go back to our **Project Manager**. Click the **Project Manager** button on the top left and double click the **Button1** to create an **OnClick** event. This event will be our basis for all the actions because they will only be started when the button is clicked.

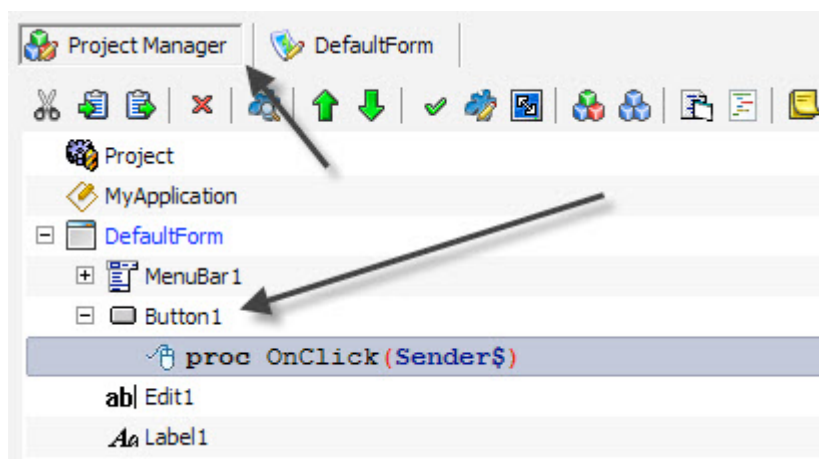


Figure 287: Double click for OnClick

- All programming languages are using variables to store data; these variables can then be used to manipulate data with the help of various programming tools. For creating a variable in PIDE we first need to declare it. For doing so, we will use the **PVariable** object from the component pane. Drag a **PVariable** from the component pane to the **OnClick** event. This will create a definition for a new variable.

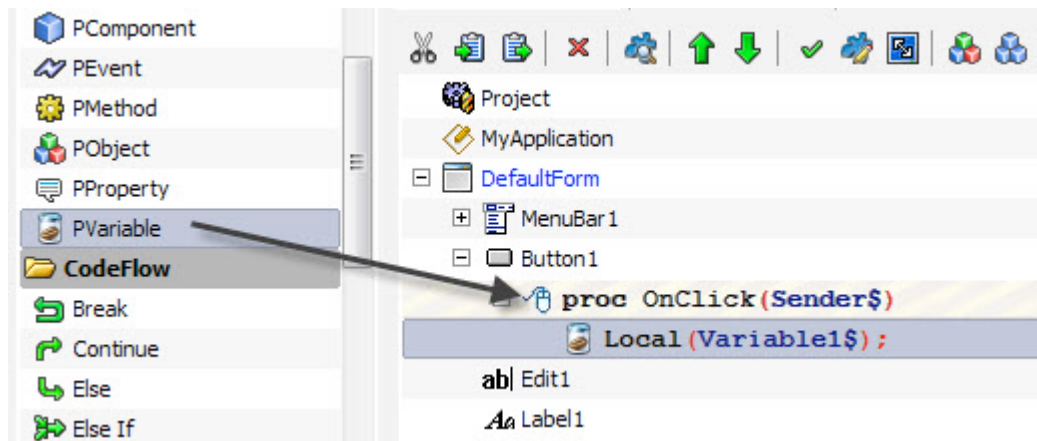


Figure 288: Drag PVariable

- Now, to create a variable, we will simply drag our newly created variable definition back on the **OnClick** procedure while pressing the **ALT** key. This will create a new variable ready to hold any value.

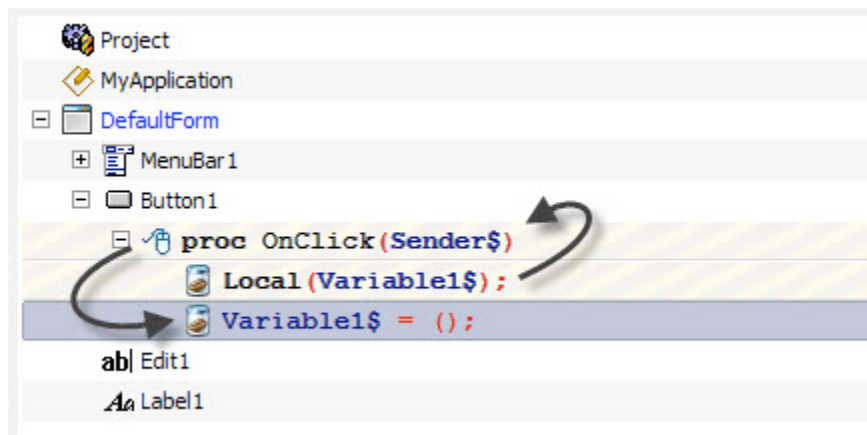


Figure 289: Create Variable

- The newly created variable does not contain any value as for now. Drag the **PEdit** object to the **Expr** property of Variable1\$ and select the Text property in the **Code Completion** box that appears.

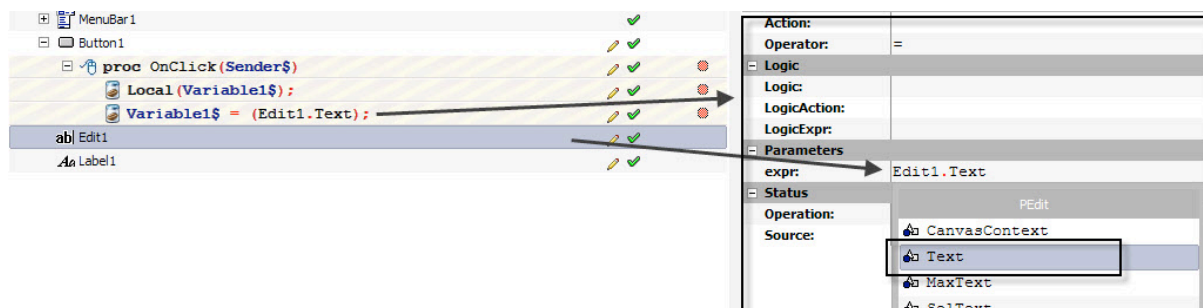


Figure 290: select Text in source

- Now **Variable1\$** holds the value entered in the **PEdit** box. We now have to transfer that value to **PLabel1**. For doing this, we will have to change the Caption property of

**PLabel** to be equal to the value contained in the **Variable1**. Drag the PLabel1 object to **OnClick** event and select Caption from the code complete window.

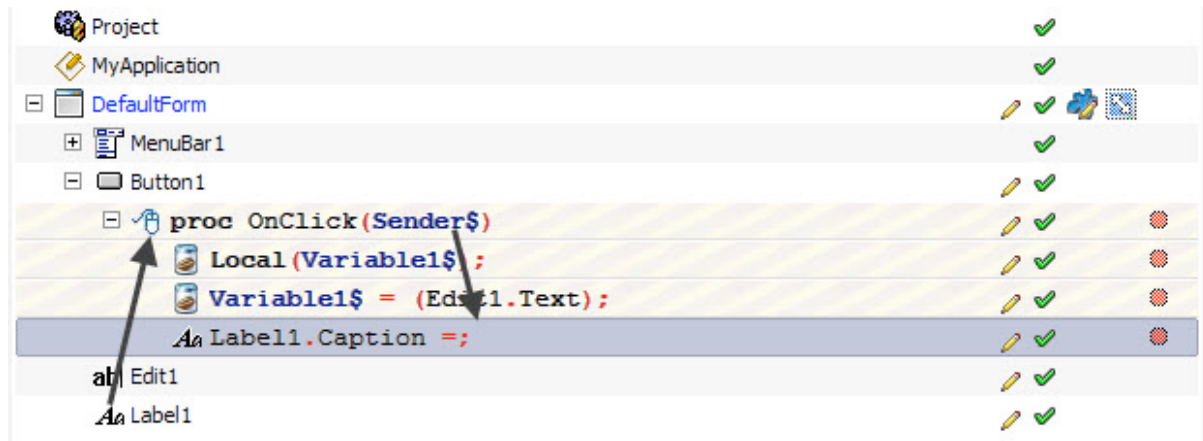


Figure 291: Drag PLabel1 to OnCreate

- All that remains now is to assign **Variable1\$** to the Caption of **PLabel1**. Drag Variable to the **Expr** property **PLabel** object.

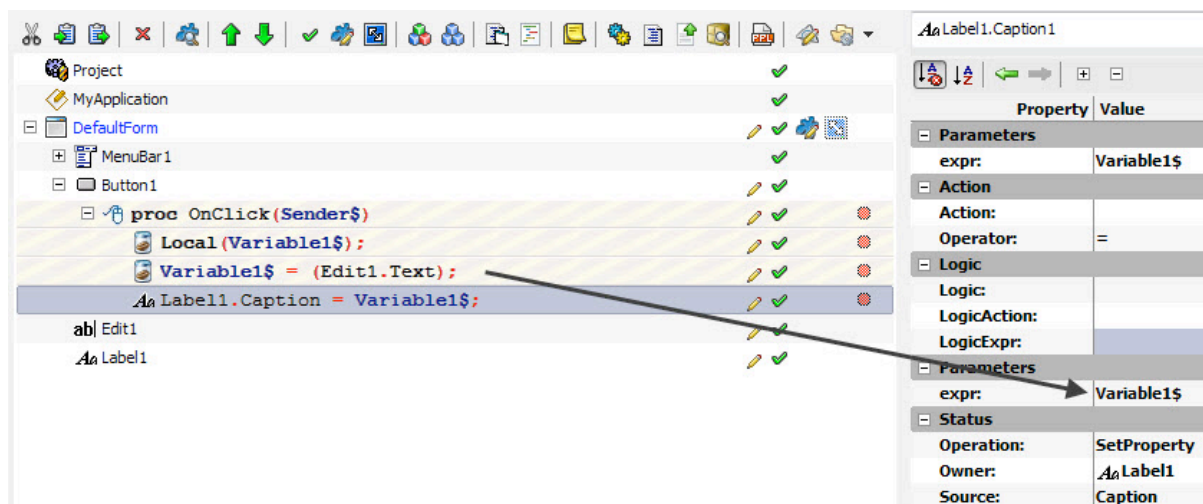


Figure 292: Drag Variable to Expr property

- Doing the above steps, we have given the value contained in a **PEdit** box to a PLabel. Save your project by pressing **Ctrl+S** key on the keyboard or go to **File>Save As..** to save it to your desired place.

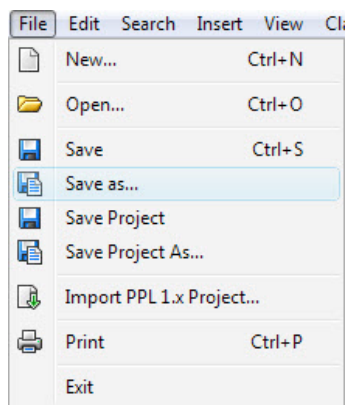


Figure 293: Save Project

- Press **F5** to run the project and see the output:

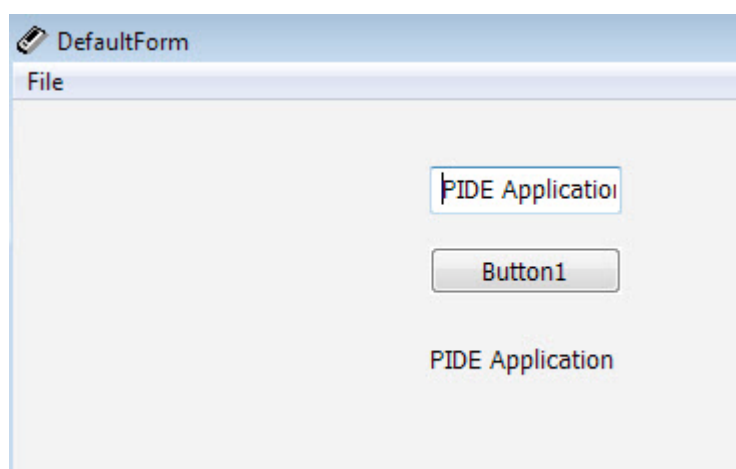


Figure 294: Project Output

## PMethod

A **PMethod** is used to add a method to a program. Methods in programming are used to add varied functionality to an application. Methods in an application do not only allow a programmer to add additional functionality, it also allows him/her to make necessary adjustments according to the programming structure. Given below is an example that would illustrate how to use **PMethod** to add two numbers and print their sum on a message box.

- Start with creating a **Desktop Form Project**. To do so, press **Ctrl+N** or go to **File>New** and select **Desktop Form project** in the **Select New Project Type..** window.

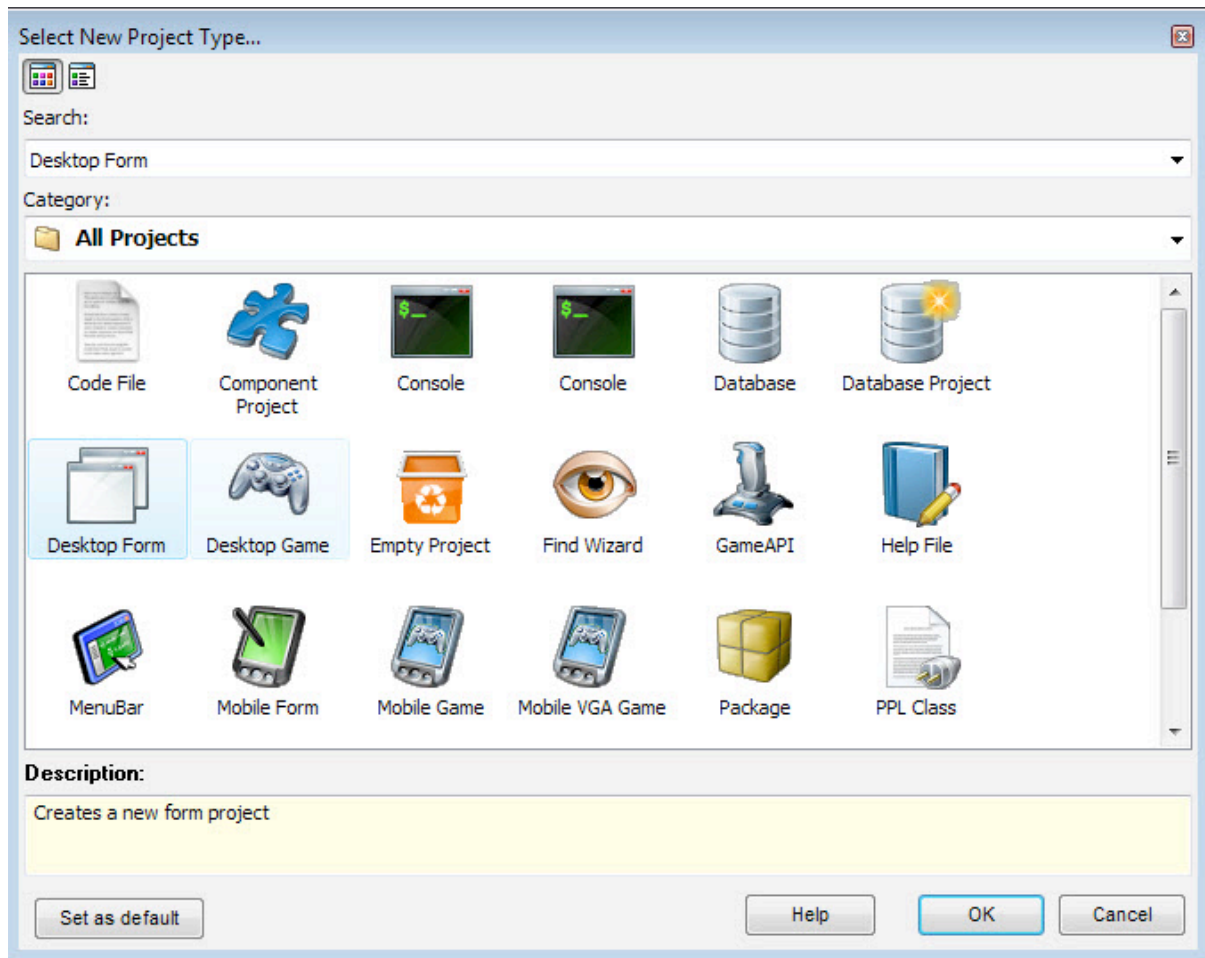


Figure 295: Create New Project

- In the **Default Form Project Manager**, double click the **Default Form** to initialize the Form editor and place two **PEdit** objects with one **PButton**; like shown in the figure below. To create the aforesaid components on the Form editor, click on the component you are looking for in the **Components Pane**. After selecting the component, click on a place on the form editor to place that component there. Refer to the screenshot below to configure your components.



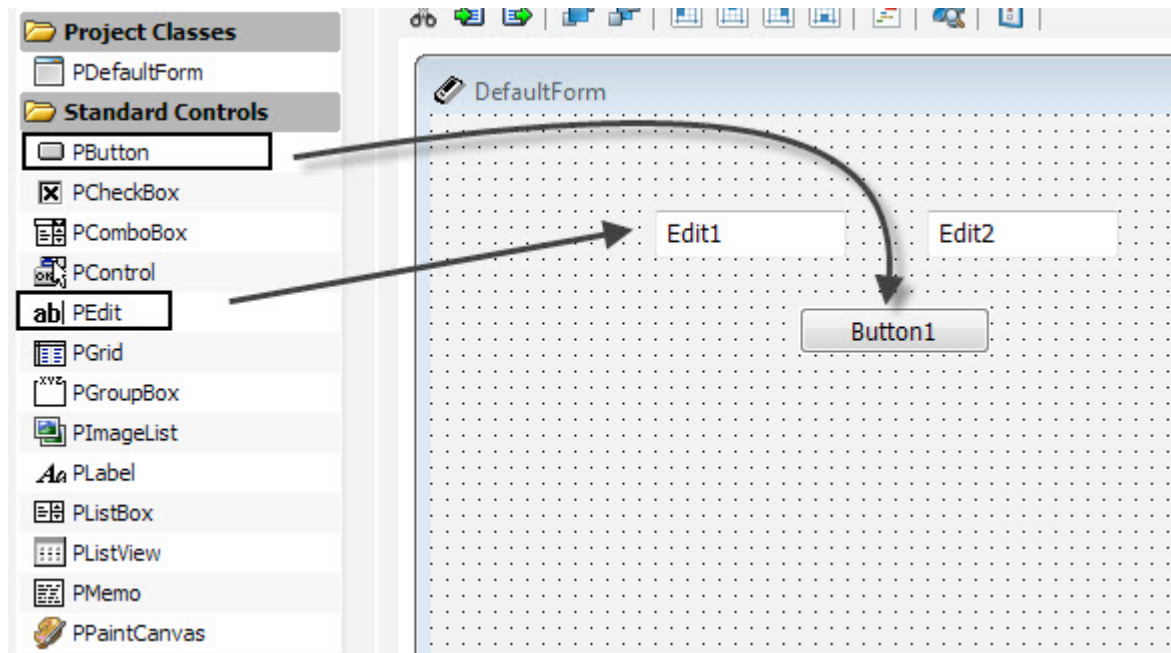


Figure 296: Place components

- After placing the **PEdit** objects and the **PButton** object on the form, click on the **Project Manager** button to go back to the **Project Manager**. You would notice there are 2 **PEdit** objects and 1 **PButton** object added to the **Project Manager**.
- Once in the **Project Manager**, drag a **PMethod** object and drop it outside the **DefaultForm** code.

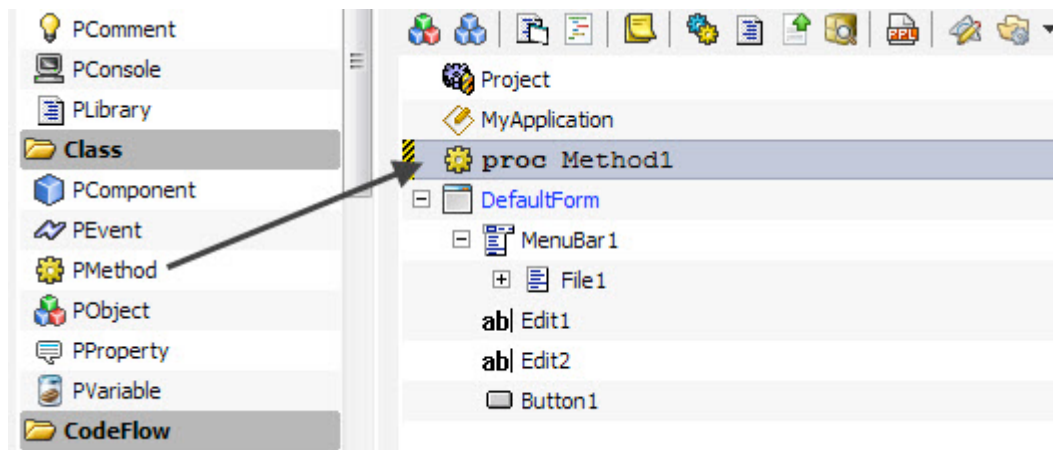


Figure 297: Drag PMethod

- Click on the **PMethod** object and change its **Parameters** property to **x\$,y\$**. This would specify that this method would take two variables, namely x and y.

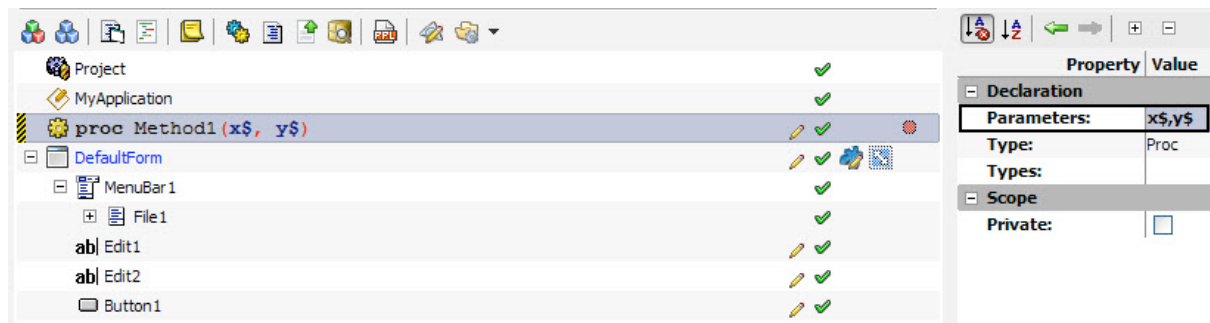


Figure 298: Change Parameters property

- Now, while the **PMethod** object is still selected, press **Ctrl+Space** to bring the **Code Completion** window.

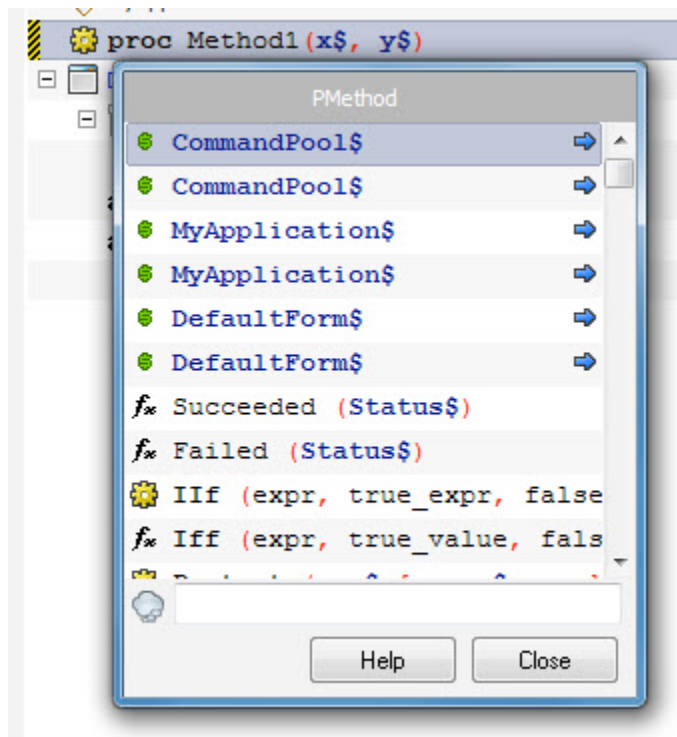


Figure 299: ShowMessage

- In the **Code Completion** window, select **ShowMessage**.

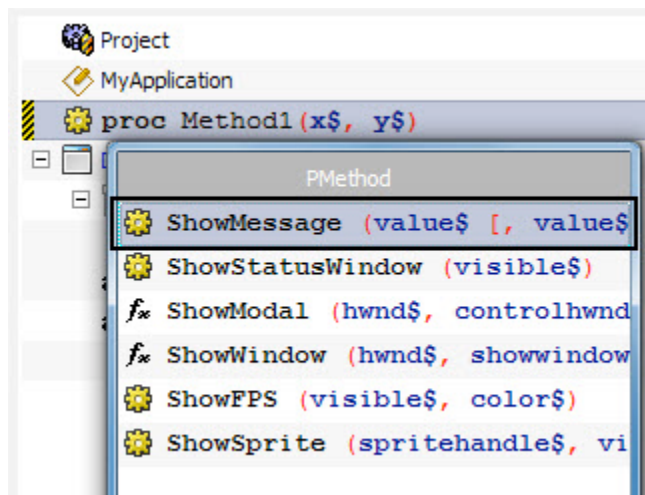


Figure 300: ShowMessage

- Select the **ShowMessage** object and write `x$+y$` in its **Expr** property. This would print the addition of both variables in a message window.

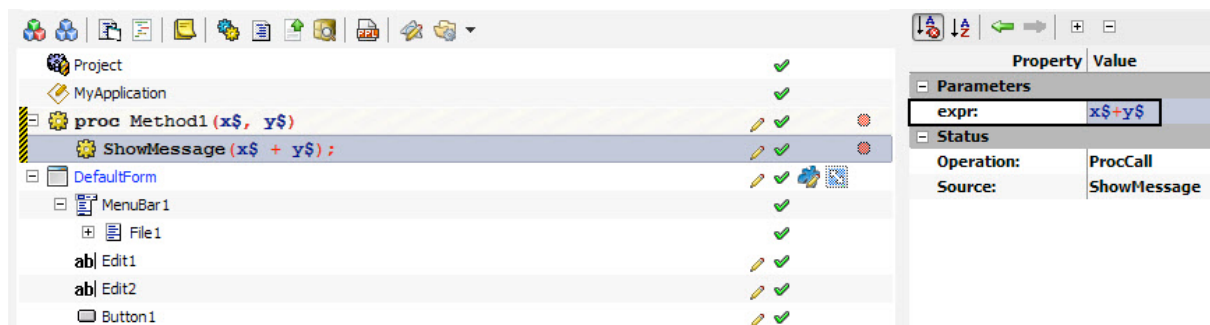


Figure 301: Change Expr property

- Now, double click on the **PButton** object to create an **OnClick** event. This event will trigger the addition of both numbers.

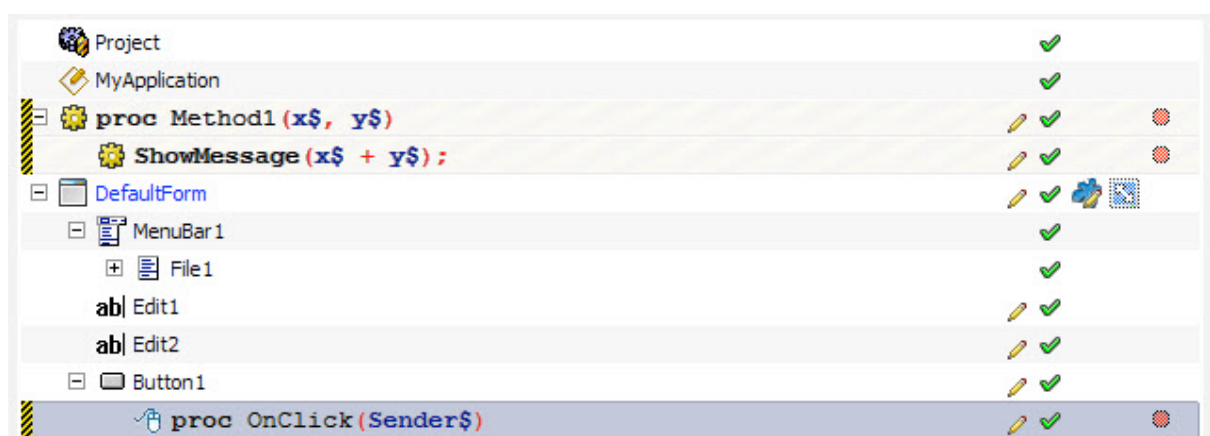


Figure 302: Double Click for OnClick event

- Drag the **PMethod** object to the **OnClick** event.

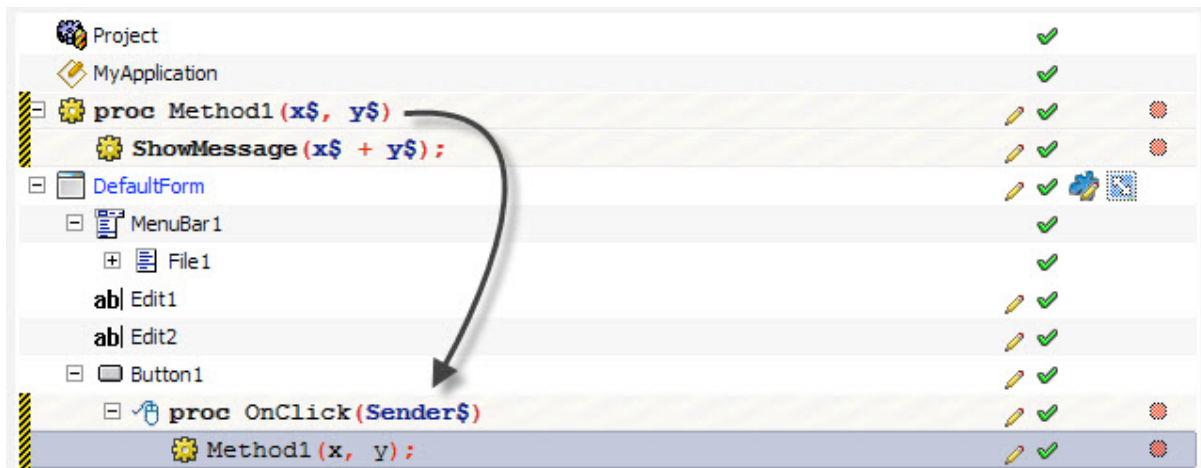


Figure 303: Drag PMethod

- While the **PMethod** object is still selected, drag and drop the **PEdit1** and **PEdit2** objects to the x and y property of **PMethod**. This would trigger a **Code Completion** window. Select the text property in them. Doing so will allow x and y parameters to have the values input by the user in **PEdit** objects.

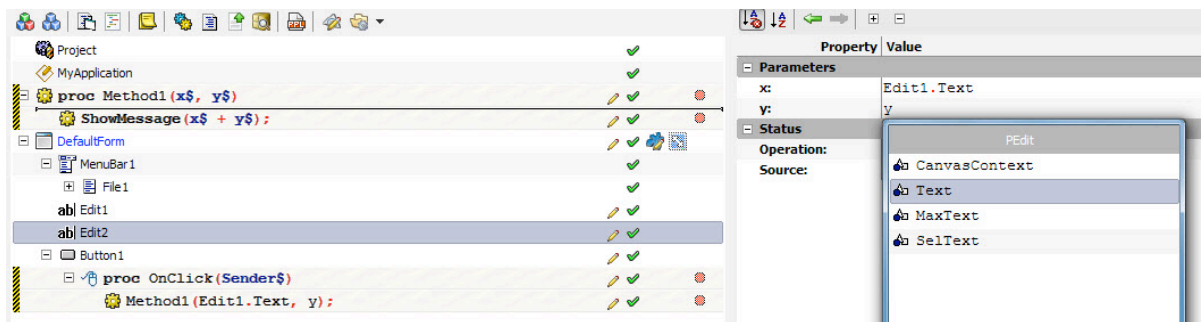


Figure 304: Drag PEdit objects to x,y property

- Save the application by pressing Ctrl+S on the keyboard or go to File>Save Project menu item to save the application.

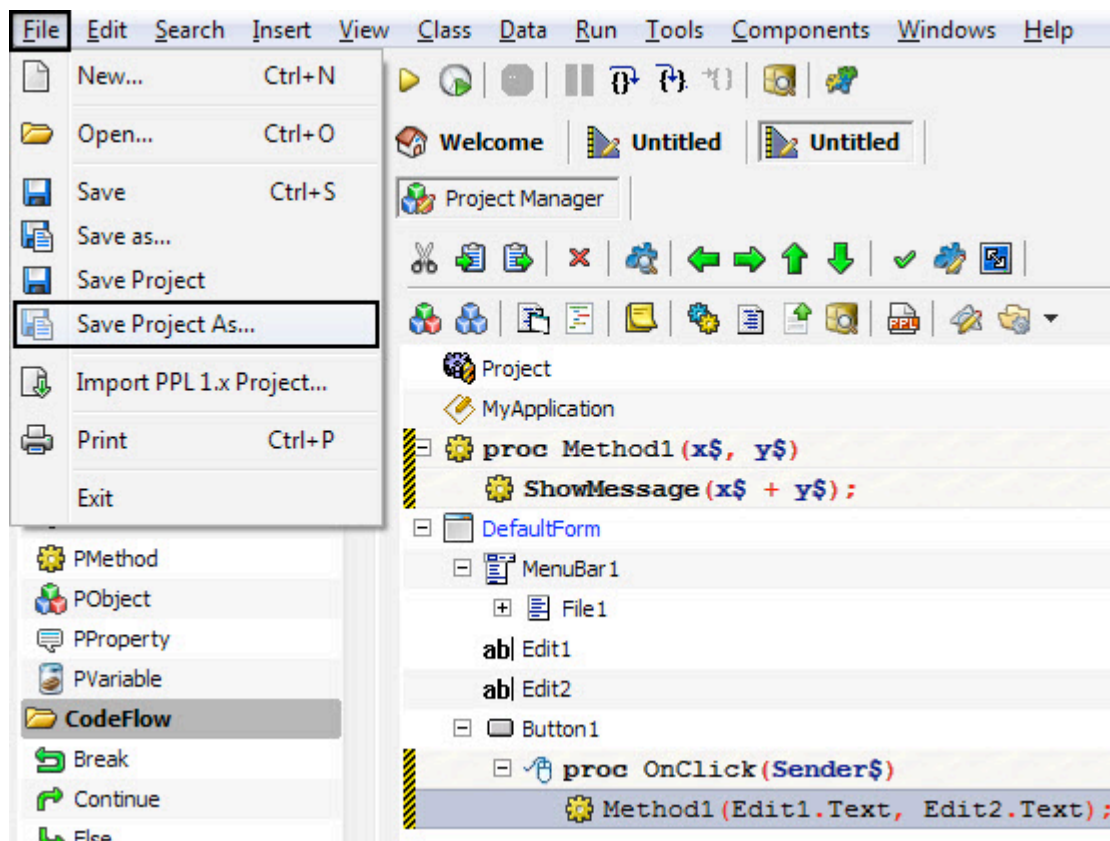


Figure 305: Save Project

- Press **F5** to see the result! This application takes in two numbers and prints the addition by using a **PMethod** object.

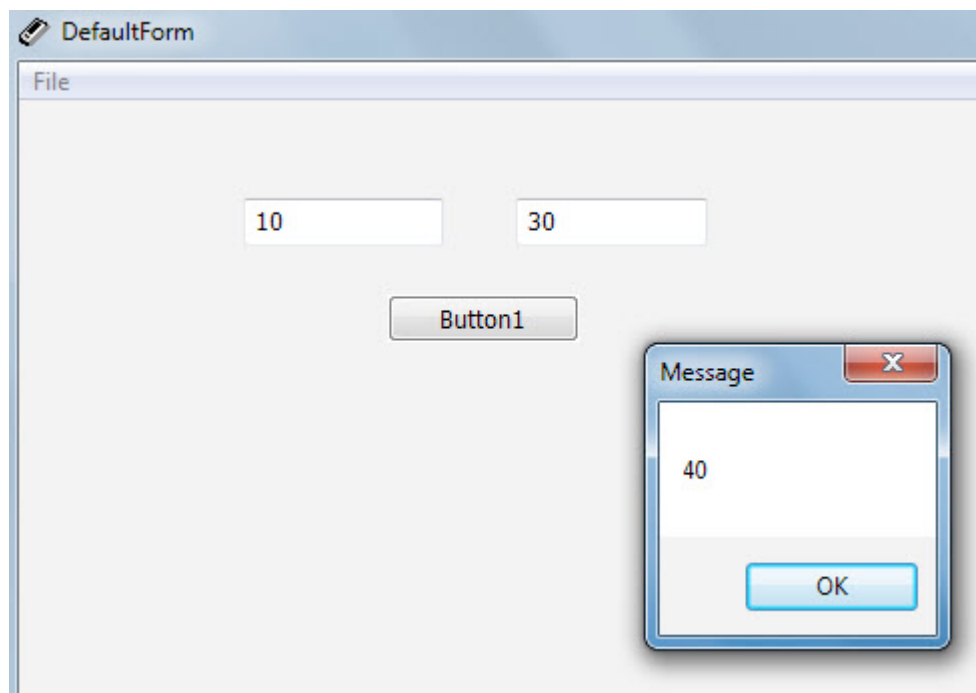


Figure 306: Output

## Creating Your Own Components And Distributing Them

Creating applications is all about creating programs that can work like you want them to be. Just like every other programming language, PIDE too allows you to build your own components and reuse them whenever and where ever you feel like. Apart from the many uses of creating your own components in PIDE, you can also create your own components and provide them to other PIDE developers for inclusion in their library. There are a certain number of steps and procedures involved in development of components in PIDE; here we will know the method to create your own components in PIDE as well as distributing them.

Components are individual packages that have to be installed into a PIDE system for it to be used while programming. Once installed, programmers can use the components in their programming needs.

Creating a component in PIDE basically requires three things, they are:

- **PPL source code files** – This is where you create the component and save it as a PPL source code so that it can be acted upon during the program execution b PIDE.
- **XML definition files** – These are required by PIDE in order to complete the component packaging. You do not have to create the XML files by yourself; once a PPL source code is created, PIDE can be used to create a XML file to use in component creation.
- **Help files** – While not always necessary, you would need help files for your components if you want to distribute them or want to use the component's help for future reference.

Now that you know the things you need to create a component in PIDE, we will start with each component in detail.

### PPL Source Code

There are two ways to create a .ppl file or application; wether you can go with a manual ppl coding or use visual coding to create equally stable applications within no time. For our example we will use visual coding.

In this section, we will learn about creating PIDE classes that can be used to deliver specific functions in a program. These classes would contain their own properties, methods, logic etc. The **DbNavigator** class example is the perfect example of a class that can be created with the help of PIDE programming and then used with other projects.

The DbNavigator class uses a toolbar interface and allows a user to navigate through the fields of a table easily. Given below are the steps that would allow you to create a DbNavigator class in the **Components Panel**.

- Start by creating a new **Component Project** in PIDE2 and deleting the existing components class as we will be creating that on our own. Drag a **Pcontrol class** to the project and rename it as DbNavigator to make it look like a different class.



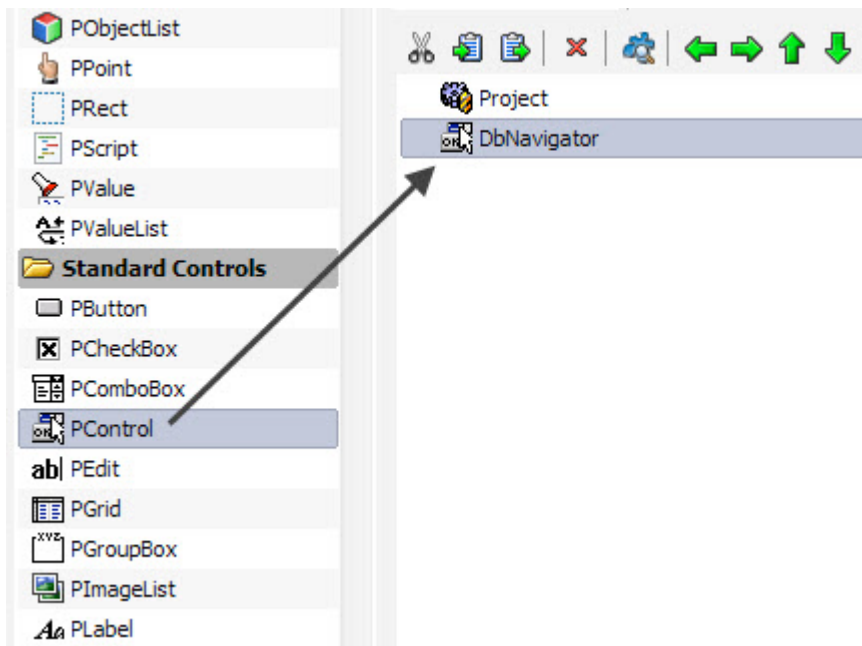


Figure 307: Drag a PControl to the Project Manager

- The DbNavigator will work by going to the previous or the next data element and this requires a dataset to work with. For providing the dataset to our **DbNavigator Class**, we will include a property by dragging and dropping a **PProperty** to the **DbNavigator Class**.

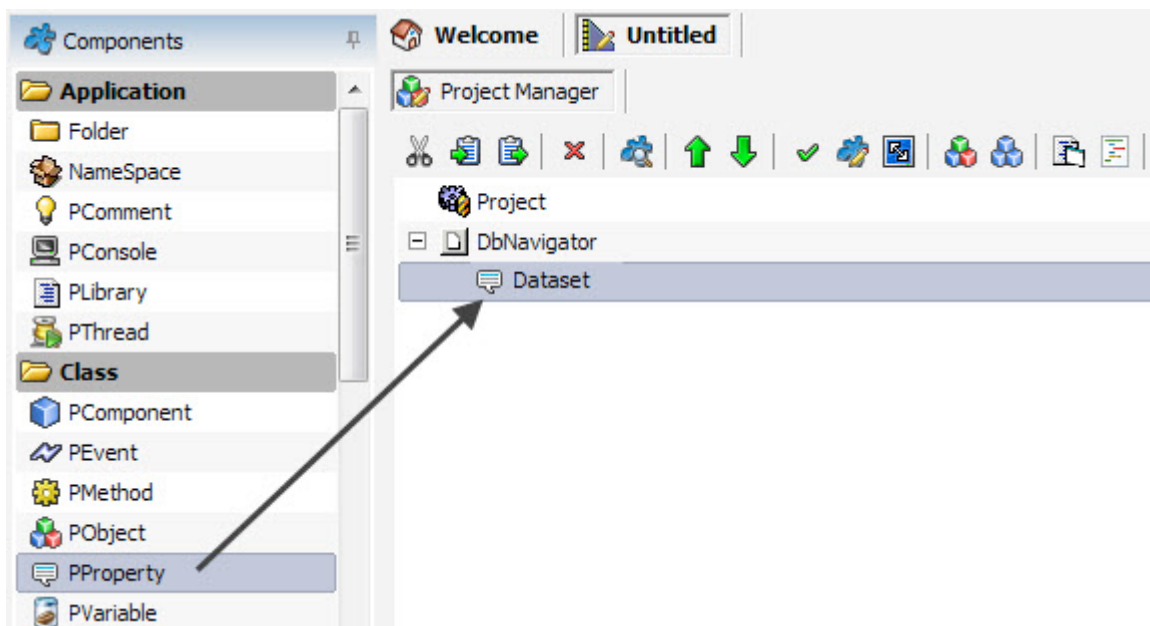


Figure 308: Drag PProperty to DbNavigator

- After dropping the **PProperty**, rename it as Dataset and set its **Type Property** to **Control**.

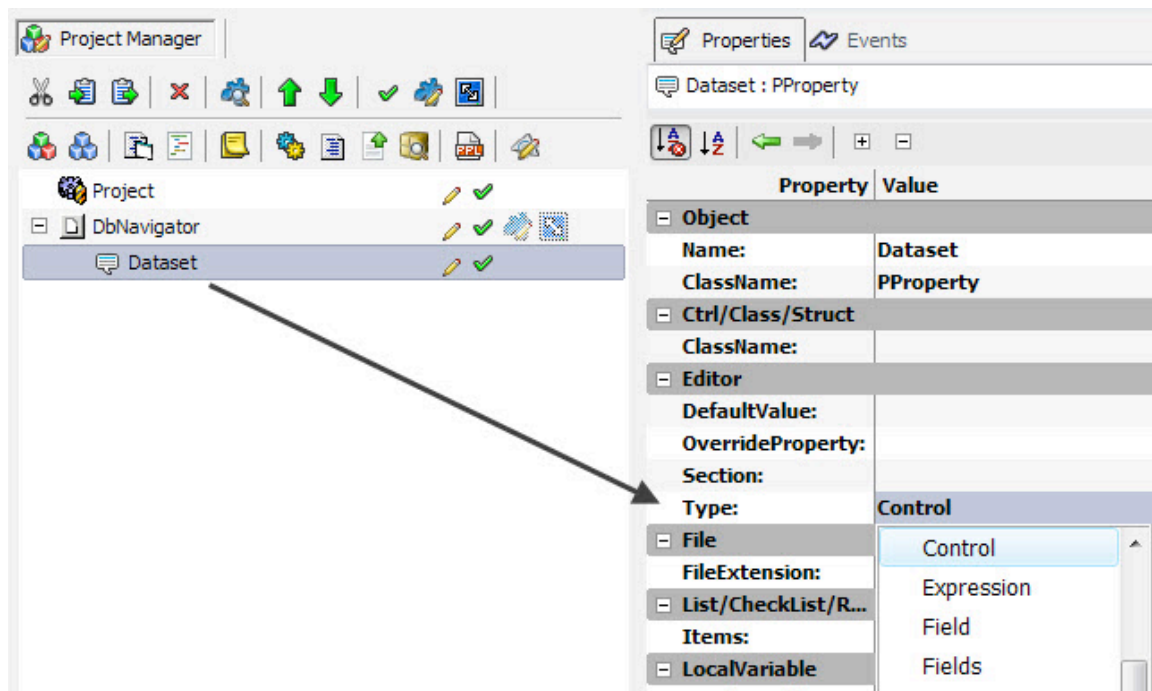


Figure 309: Set the Type property

- After change the Type property, select PDataSet in the **ClassName Property**.

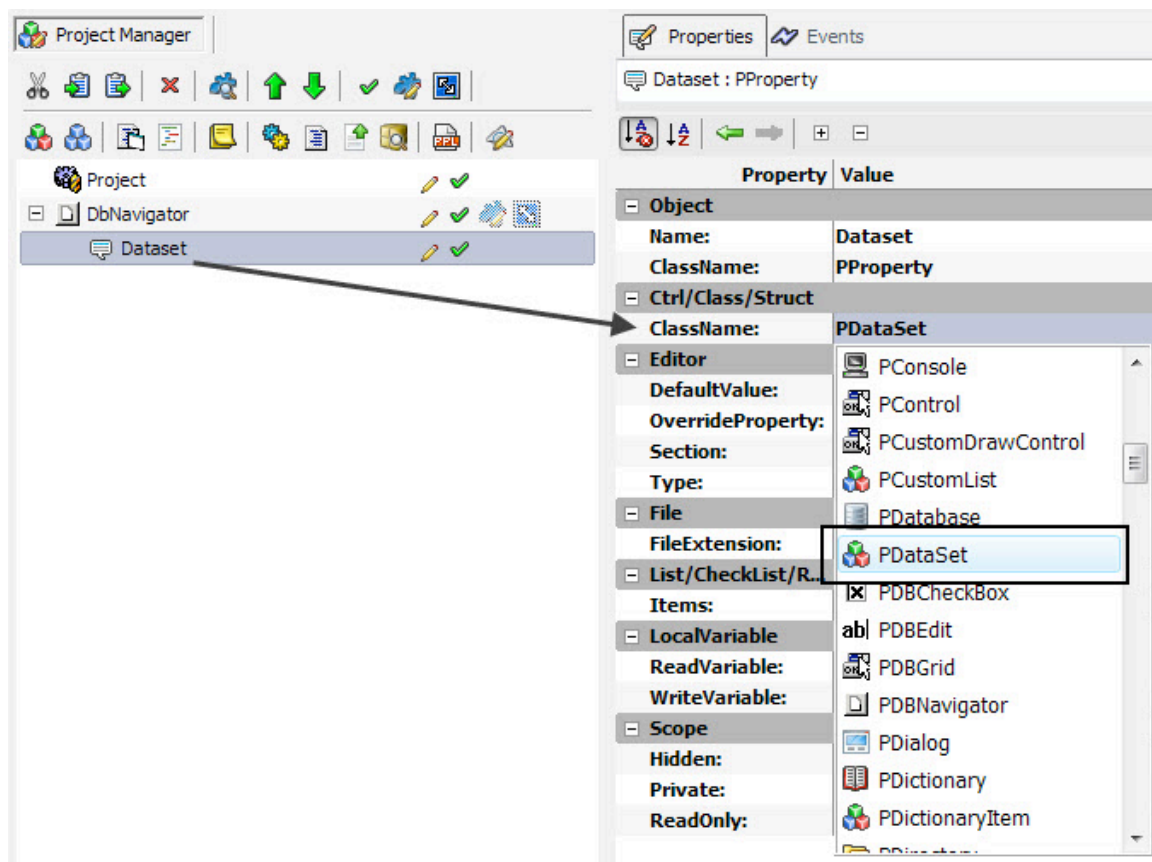


Figure 310: Set the ClassName property

- Because we want to create a navigator, we will also have to create some buttons for performing tasks. For having buttons in the **DbNavigator**, drag 4 **PToolBar** objects to the DbNavigator and name them **First**, **Last**, **Prior** and **Next**.

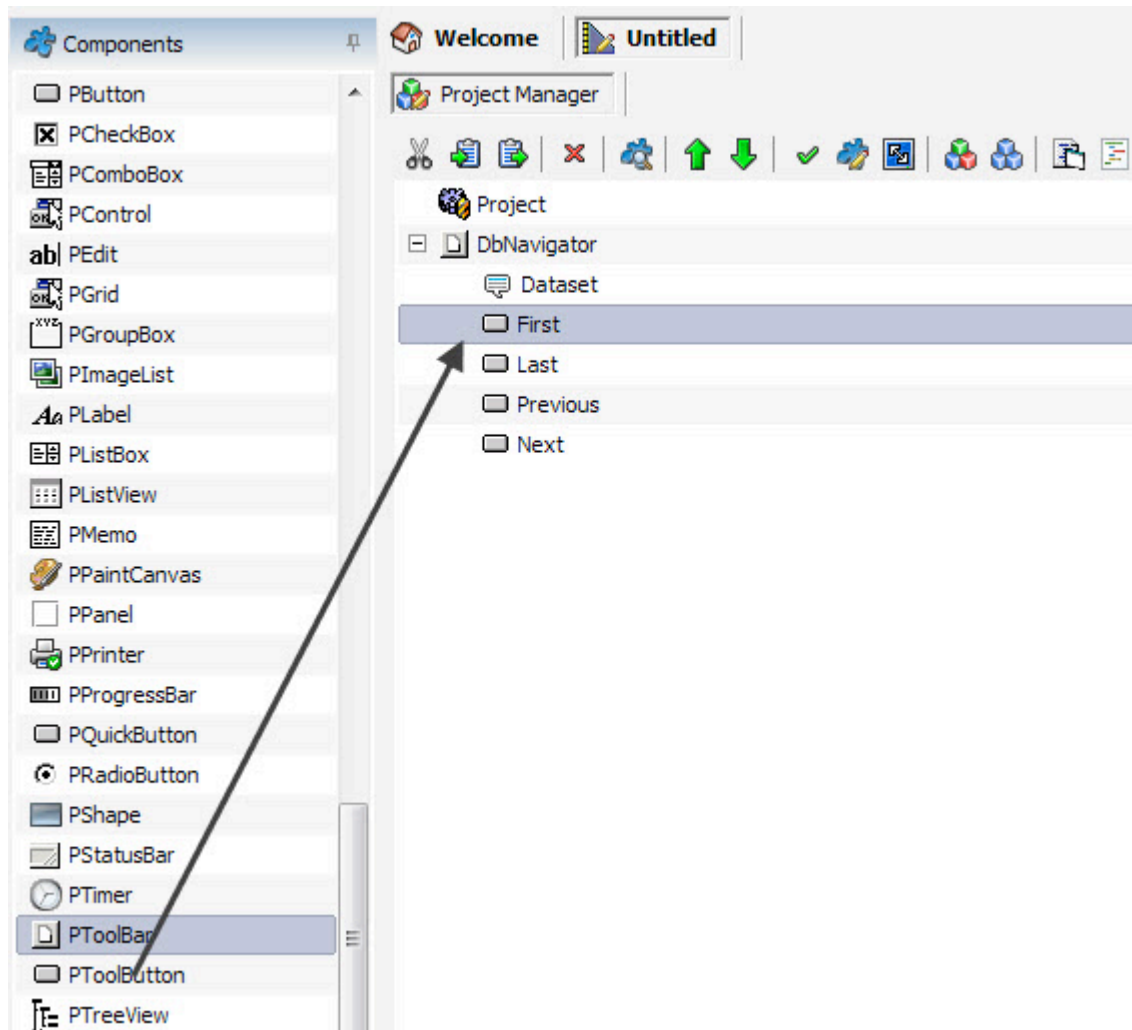


Figure 311: Drag four PToolBar Objects

- After creating the buttons, create events for the buttons by double clicking on the buttons.

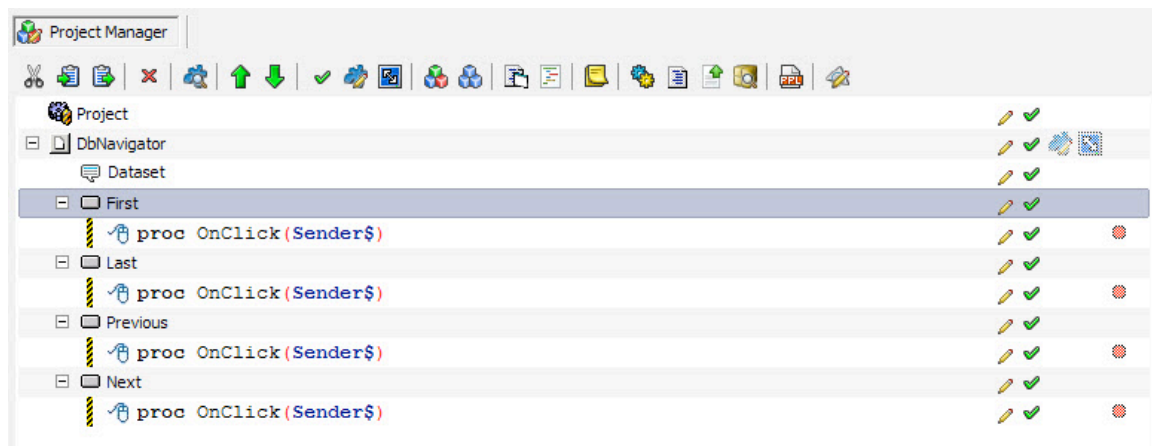


Figure 312: Double click the buttons to create events

- Now drag the **Dataset Property** to the First button's event, type 'first' and select the **first message** in the context menu that appears. Similarly, drag **Dataset Property** to the events of other buttons and select the appropriate messages.

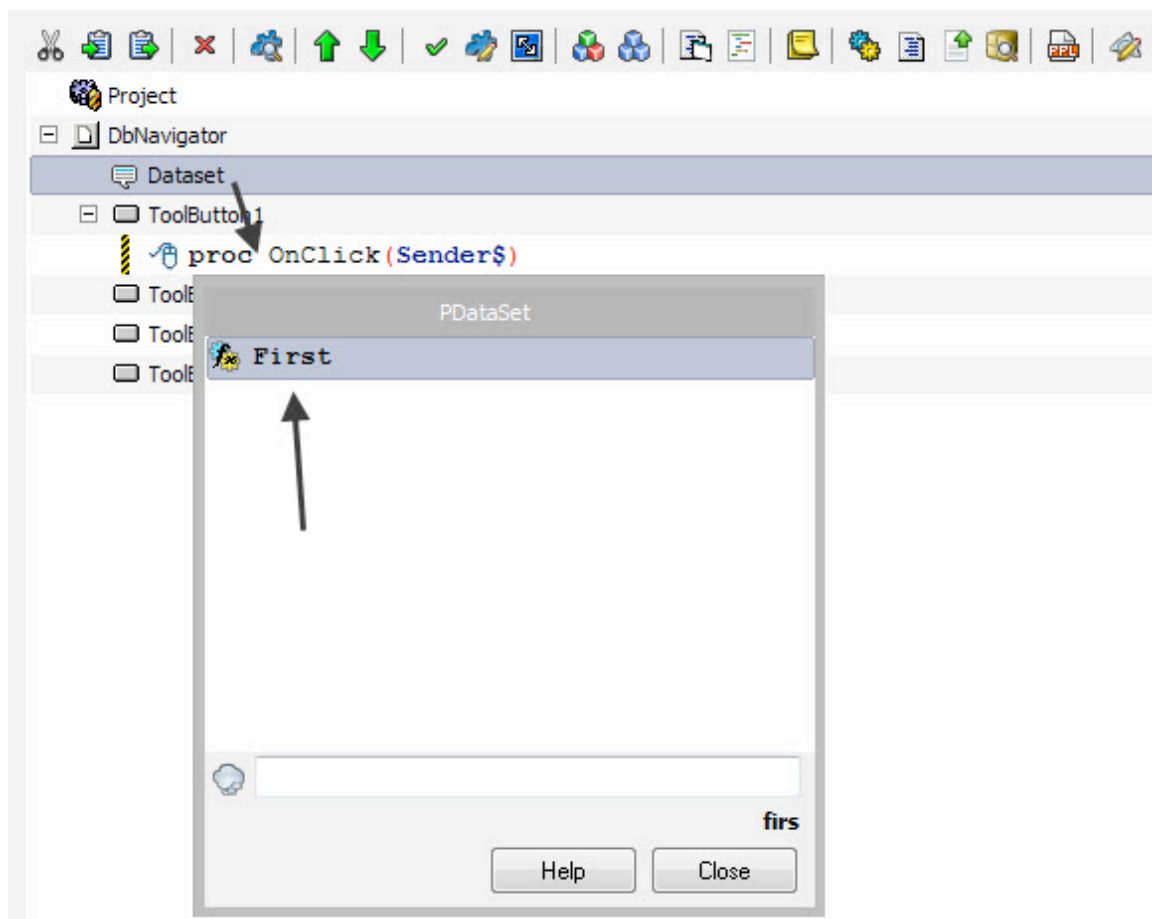


Figure 313: Select a message

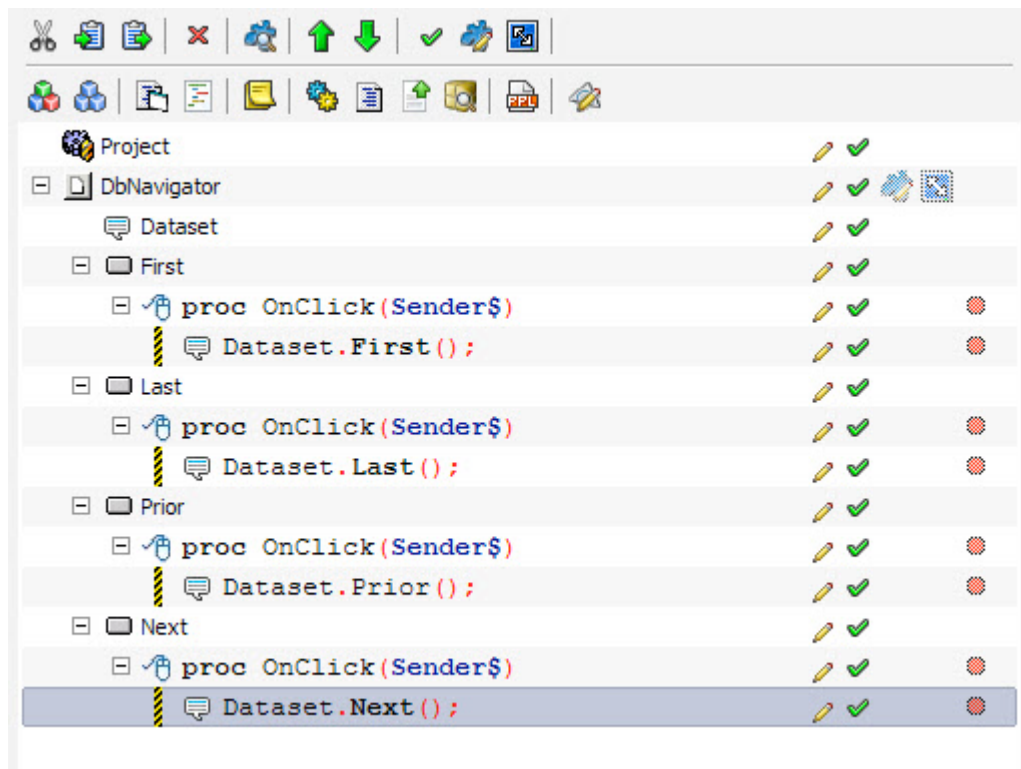


Figure 314: Select messages for all buttons

- The class component is almost complete. You can add images, sounds and other objects to the class to enhance its functionality and look. After you are completed with the **DbNavigator**, save the file with the desired name by pressing **Ctrl+S** or by using the **Save As..** Option from the file menu.
- After you have saved the file, some changes have to be made in the **DbNavigator properties** for it to become a **Class Component**. In the **DbNavigator properties**, check the **AutoCreate Property**, specify the category in which **DbNavigator** will be included in by writing it in the **Category Property**, write **PForm** in the **ChildOf Property** to determine where this class will work on. **PForm** as well as **Ppanel** can be placed on almost all visual codes. After checking the **HasOwner, HasParent and Visual Property**, save the file again. Users can also tweak other properties like the **library Property** and the **Icon Property** if they are using other options. **HasOwner** means the object will be created and owned by the root object (like form, panel, etc...). Simply speaking, selecting this property specifies that this object has a parent object that would contain it. **HasParent** is the same as **HasOwner** and ties the control to the parent control. If the parent control is moved, it would also move the child control which has been specified with the **HasParent** property. **HasParent** property can be very useful in many functions. For example, if you hide the parent control, all child controls will hide as well, etc...

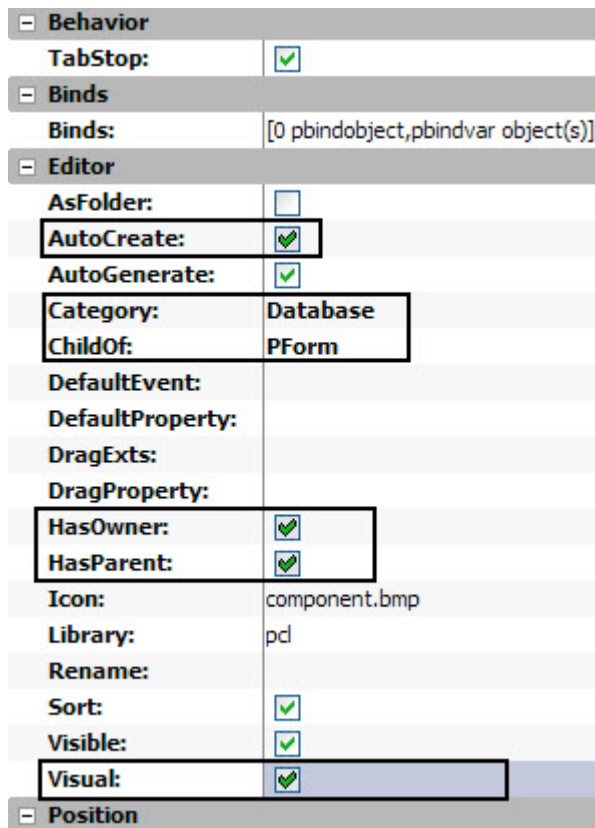


Figure 315: The property panel

- After tweaking the properties of the class, go to the components menu and select **Install components from the current project**. This will create a DbNavigator in the **Components Panel** under the database column.

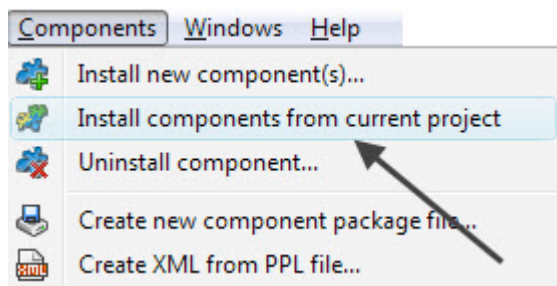


Figure 316: Components Panel

- To use the newly created **DbNavigator Class**, create a **New Project** and add a database along with the tables and fields to it.

Drag the **DbNavigator Class** to the **Project Manager** and change the **Dataset Property** of DbNavigator to the **table** in the database. This will provide the data available in the fields of the given table to the DbNavigator.

## XML Definition Files

PIDE includes a built-in function that will automatically generate a XML file from a .ppl file. When a component is created, PIDE automatically creates the XML definition that is needed for its creation.



## Generate the Component Package

Once you have the .xml file definition, the help(.hlp) file(optional) and the .ppl source code, you can generate a component package that can then be installed in other PIDE installations. To generate the package,

- Select the **Create Component Package Option** from the **Components Menu**. This will open the **Select Component Definition File... Window**.

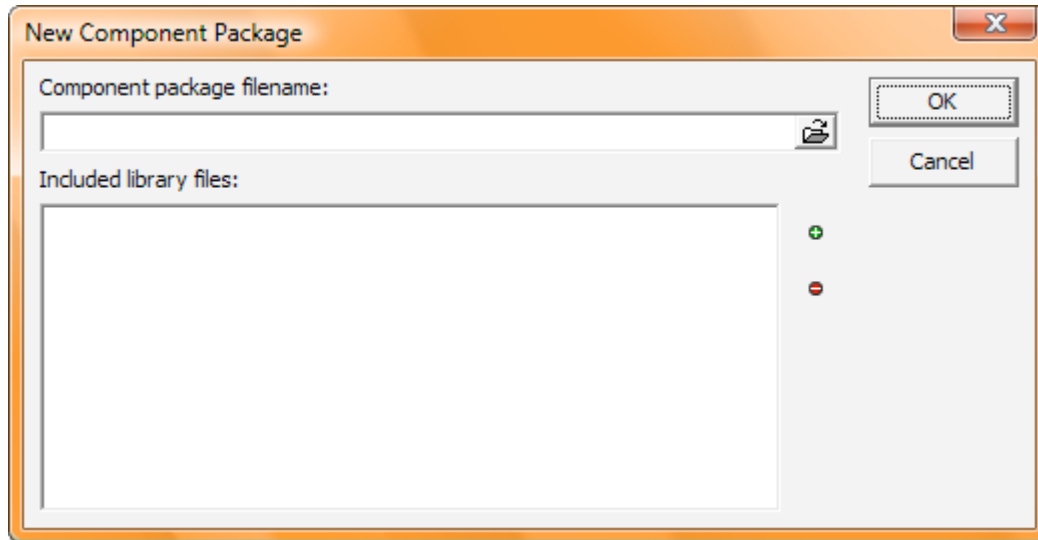


Figure 317: creating a new component package

The Select Component Definition File Window allows you to locate the XML definition file associated with the .ppl source code.

When you select the .xml definition file, the **Select Help Definition File... Window** will open allowing you to locate the help file for the component.

When you select the help file, the **Save Component Package Where... Window** will open allowing you to locate a location to save the component package file.

This will create a **ZIP** file containing the **PPL, XML and HLP files** to distribute. If you or someone else would want to install that component package file that is just created in PIDE, select **Install New Components**, not Install Components from current project.

## More In-Depth Object Binding

Object binding allows a programmer to bind two objects in PIDE with each other, with the use of object binding, object properties can be bound by properties or by variables so as to provide more control over the program and provide interactivity.

The example given below illustrates how the text property of a **PEdit** object can be bind with the caption property of a **PButton**.

- Start with creating a **Desktop Form** Project. To do so, press **Ctrl+N** or go to **File>New** and select **Desktop Form** project in the **Select New Project Type..** window.

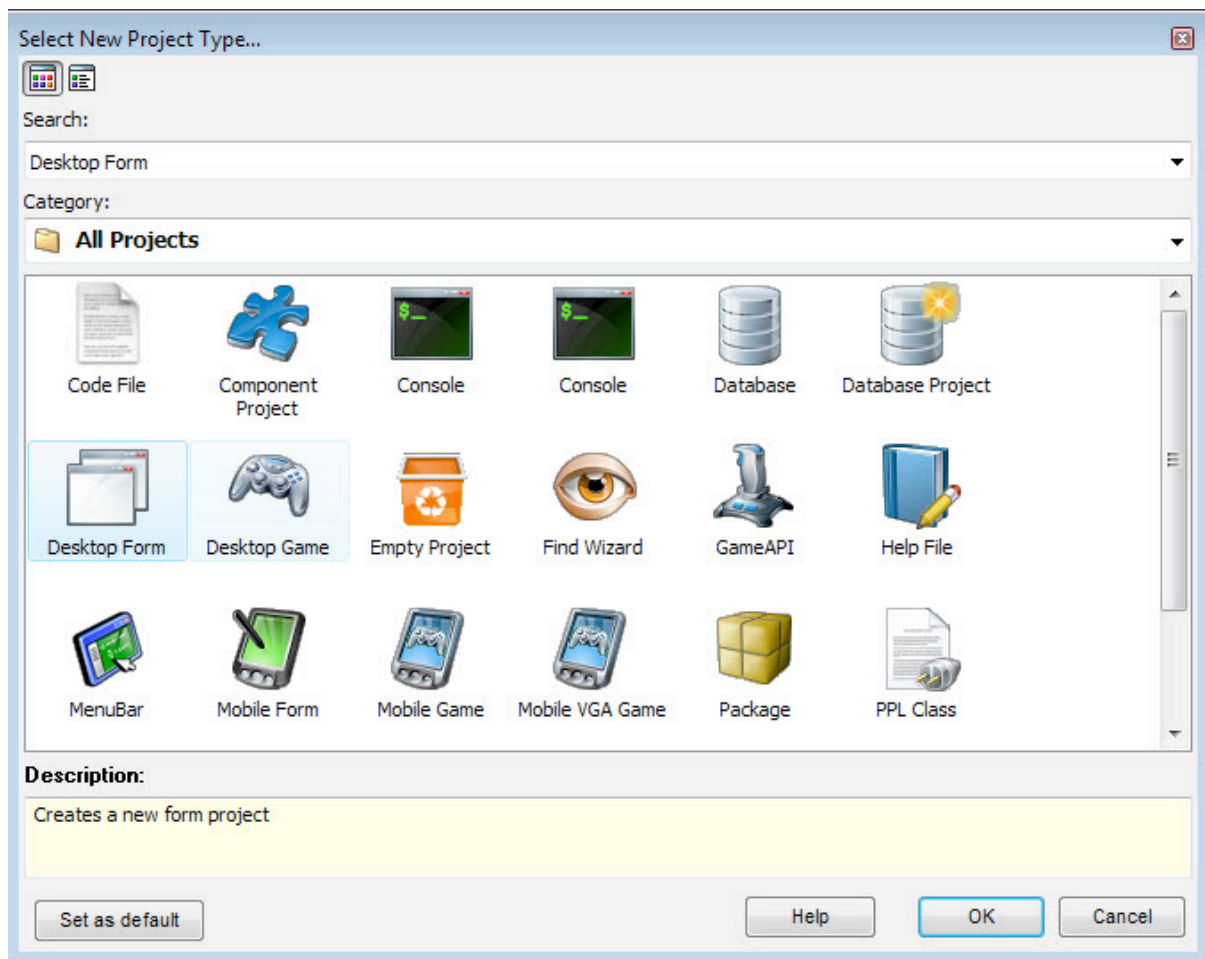


Figure 318: Create New Project

- In the **Default Form Project Manager**, double click the **Default Form** to initialize the Form editor and place one **PEdit** object with one **PButton**; like shown in the figure below. To create the previously mentioned components on the Form editor, click on the component you are looking for in the **Components Pane**. After selecting the component, click on a place on the form editor to place that component there. Refer to the screenshot below to configure your components.

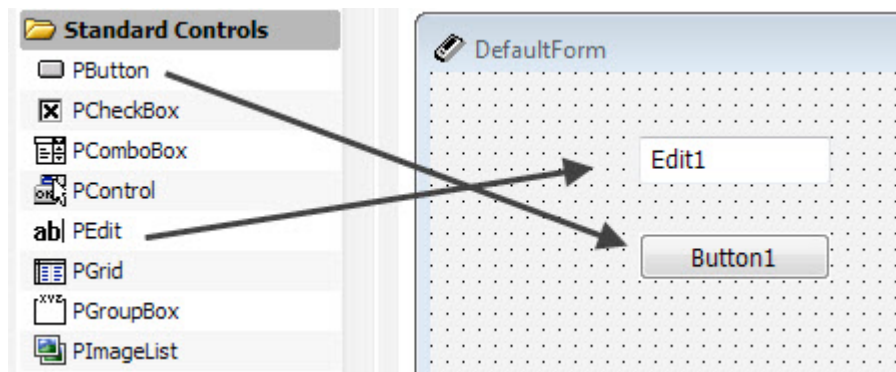


Figure 319: Place components

- For binding the property values of two objects with each other, **right click** on the source object and select **View Binds** or press **Shift+Ctrl+B** to open the **Edit Binds** window.

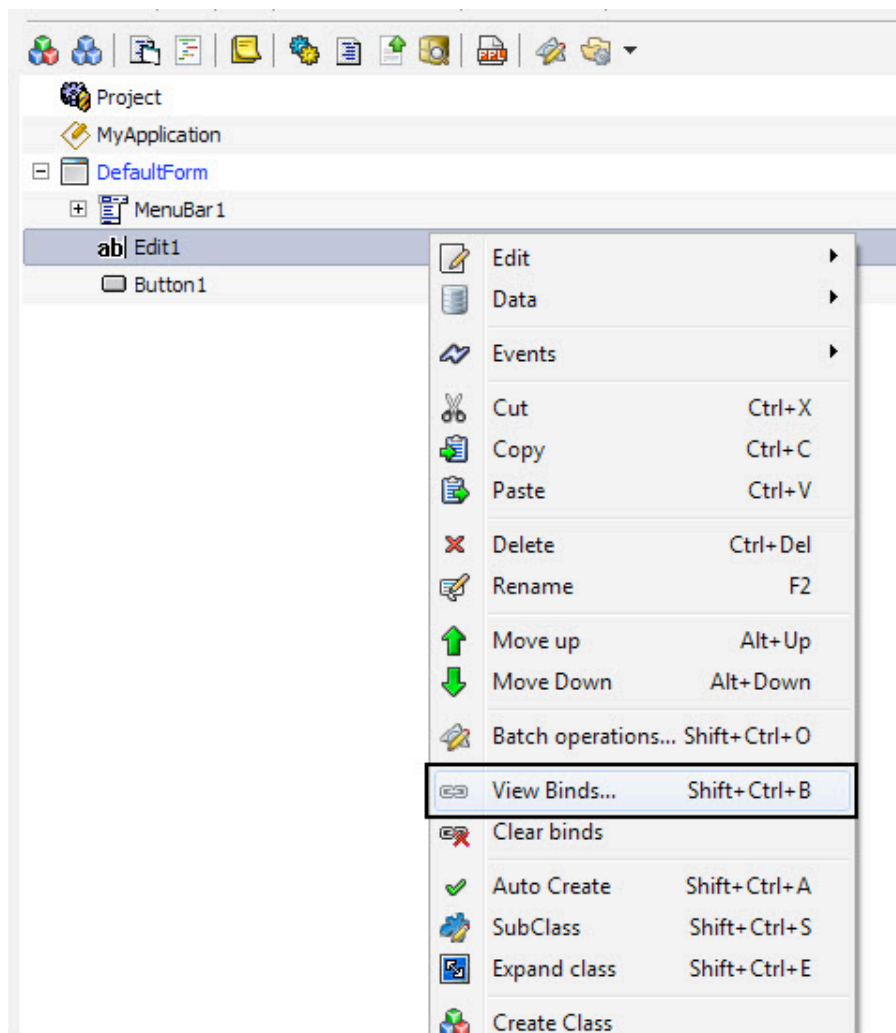


Figure 320: View Binds option

- In the Edit Binds window, select the **New Button** to open a 'New Component Class' menu and select **pbindobject**. The **pbindvar** option is used to link a variable with the object property.

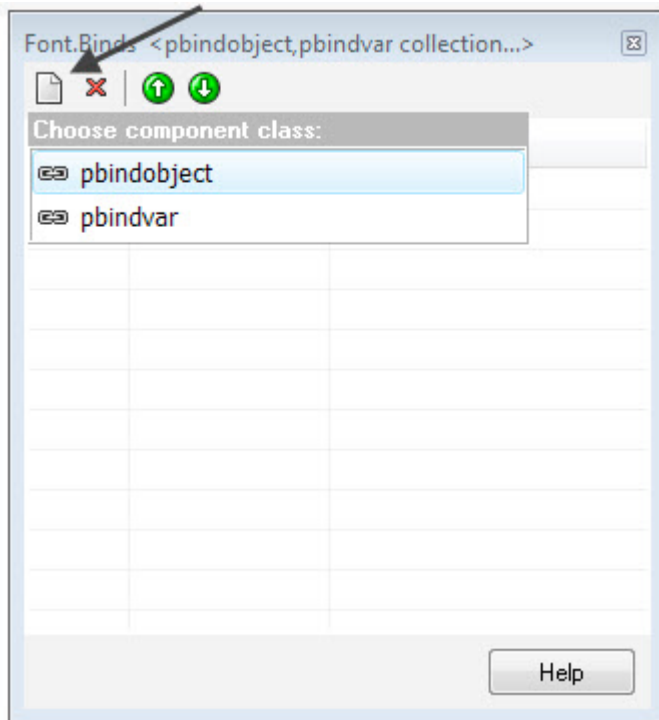


Figure 321: choosing components class

- Property panel for object binding is divided in two parts i.e. the **source** and the **target**. For binding two objects with each other, fill in all the property values in the given properties.

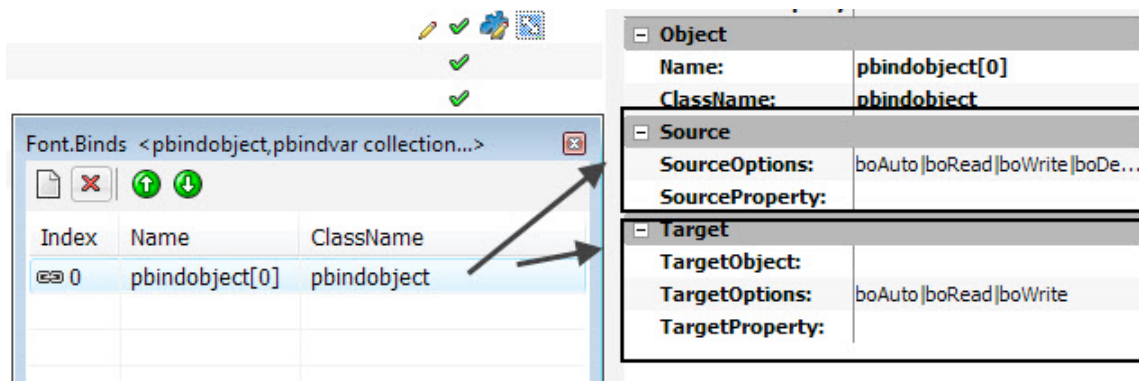


Figure 322: Source and Target sections

- In the **Property Panel**, select the **SourceOptions** property and check or uncheck the property you want. The **SourceOptions** property consists of four choices, namely,
  - boAuto** – Specifies whether property will change automatically or not
  - boDefault** – Specifies whether this is the default value for the target or not
  - boRead** – Specifies whether it will read from the target or not
  - boWrite** – Specifies that the target value will be written to the source property value

- The **SourceProperty** property specifies the source property that will be linked with the target property.

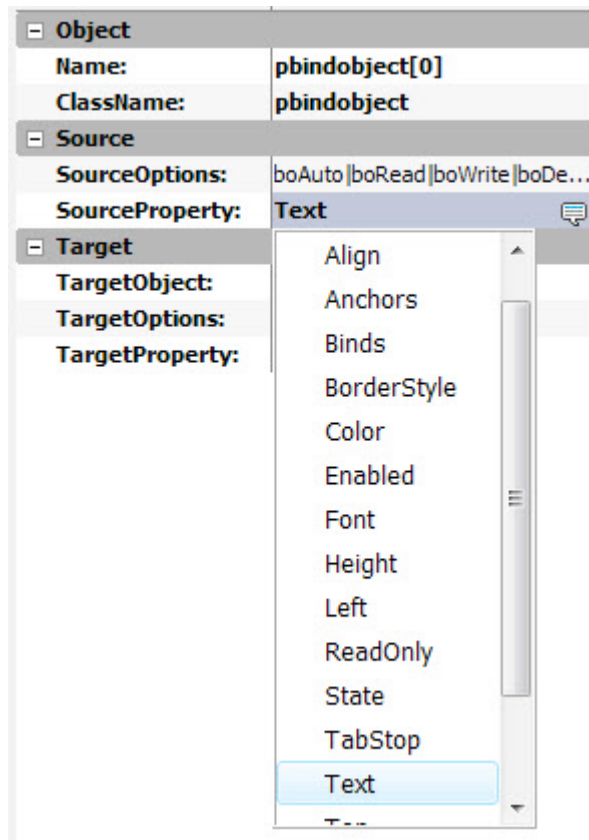


Figure 323: SourceProperty

- The TargetObject property allows a user to specify the **Target Object** that will be linked with the **Source Object**.


| Property        | Value   |
|-----------------|---|
| <b>Object</b>   |   |
| Name:           | pbindobject[0]  |
| ClassName:      | pbindobject   |
| <b>Source</b>   |   |
| SourceOptions:  | boAuto   boDefault   boWrite  |
| SourceProperty: | Text  |
| <b>Target</b>   |   |
| TargetObject:   | Button1    |
| TargetOptions:  | <div> <input type="radio"/> None </div>   |
| TargetProperty: | <div> <input type="checkbox"/> DefaultForm </div> <div> <input checked="" type="checkbox"/> DefaultForm.Button1 </div> <div> <input type="checkbox"/> DefaultForm.Edit1 </div> <div> <input type="checkbox"/> DefaultForm.Exit1 </div> <div> <input type="checkbox"/> DefaultForm.File1 </div> <div> <input type="checkbox"/> DefaultForm.MenuBar1 </div> <div> <input type="checkbox"/> MyApplication </div> |

Figure 324: Selectiong a TargetObject

- The **TargetOptions** property allows a user to specify the options that will determine the object binding options for the **Target Object**. Just like SourceOptions, these options govern how the **Target Object** will function with **source object**.

| Property        | Value   |
|-----------------|---|
| <b>Object</b>   |   |
| Name:           | pbindobject[0]  |
| ClassName:      | pbindobject   |
| <b>Source</b>   |   |
| SourceOptions:  | boAuto   boDefault   boWrit   |
| SourceProperty: | Text  |
| <b>Target</b>   |   |
| TargetObject:   | Button1   |
| TargetOptions:  | boAuto   boWrite  |
| TargetProperty: | <div> <input checked="" type="checkbox"/> boAuto </div> <div> <input type="checkbox"/> boDefault </div> <div> <input type="checkbox"/> boRead </div> <div> <input checked="" type="checkbox"/> boWrite </div> |

Figure 325: Target options

- The **TargetProperty** property specifies the property of the target object that is linked with the property of the **source object**.



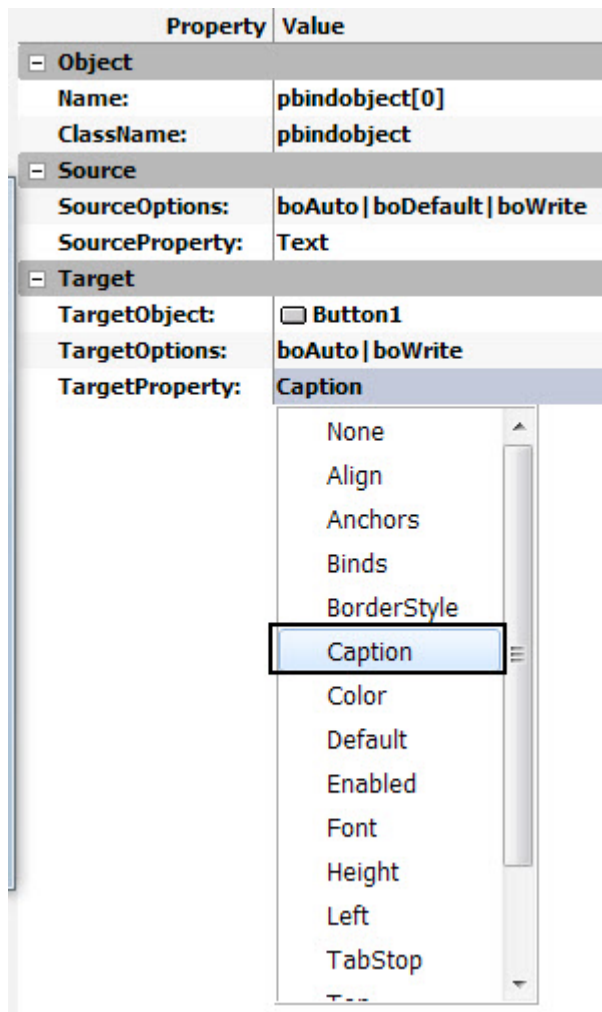


Figure 326: Target property

- After all the object properties are filled, you can close the object binds window and run your project to see the results.

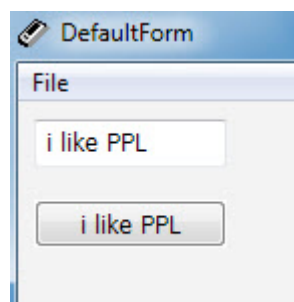


Figure 327: PEdit text binded with PButton object

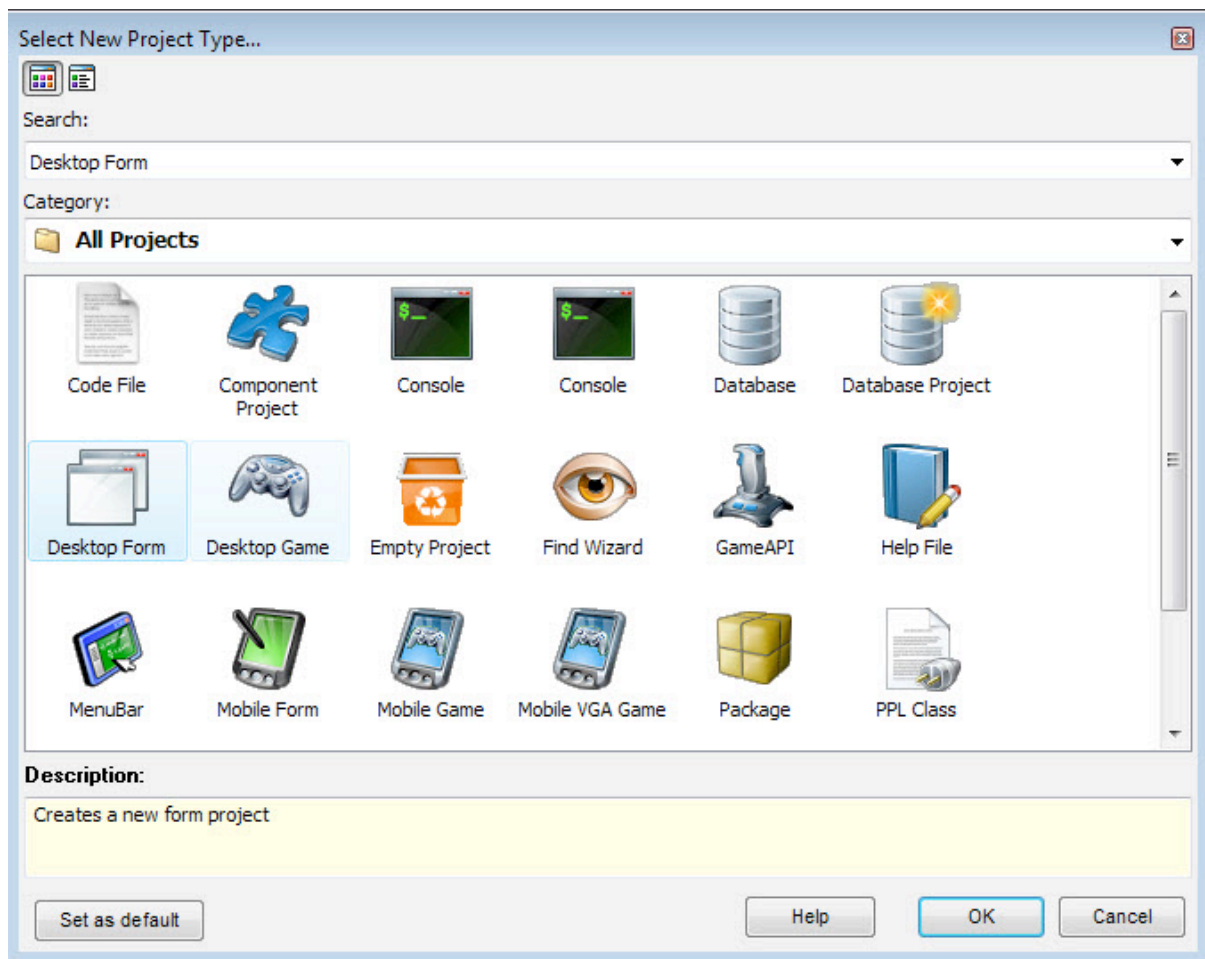
## PDirectory, PFile And PFileList Example

File handling is an integral part of any comprehensive application. Believe it or not, anything from a small text editor application like notepad to an hefty one like PIDE itself needs to save data to and Load Data from files and that's where file handling comes in. Being a useful programming ability, PIDE too has the ability to open, save and close files as well as folders with the help of **PDirectory**, **PFile** and **PFileList** objects play an important role in it.

In this section of the manual, we will get to know about such objects and what PIDE users can do with them.

### ***Displaying the number of files available in a directory with PDirectory Object***

- Start by creating a new **Desktop Form** project. For doing this, press **Ctrl+N** or Go to **File>New** and select **Desktop Form** from **Select New Project Type...** window.



**Figure 328: Create new project**

- Drag a **PButton** object to the **Project Manager** and double click it to create an **OnClick** event. This event will help us create actions once the button is clicked.

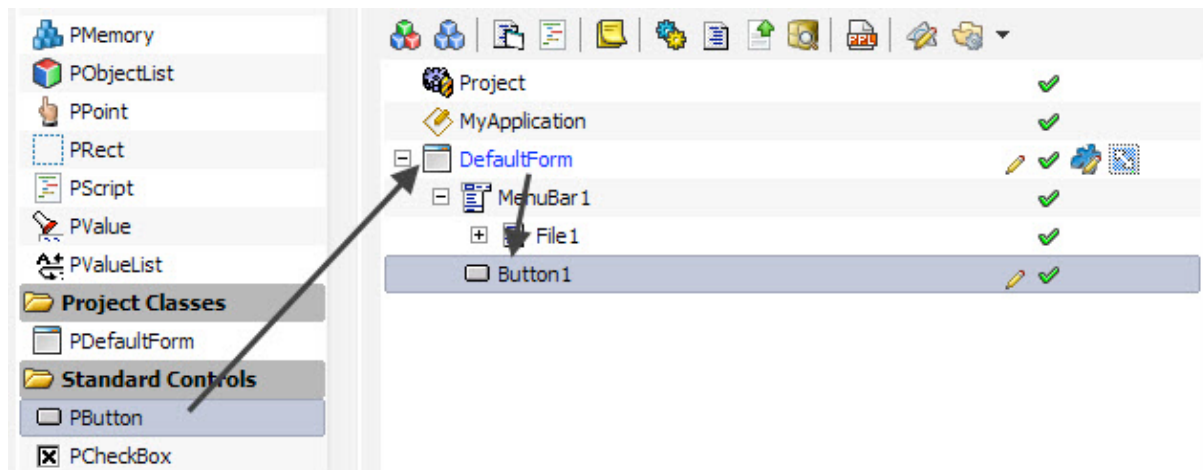


Figure 329: Drag PButton to Form

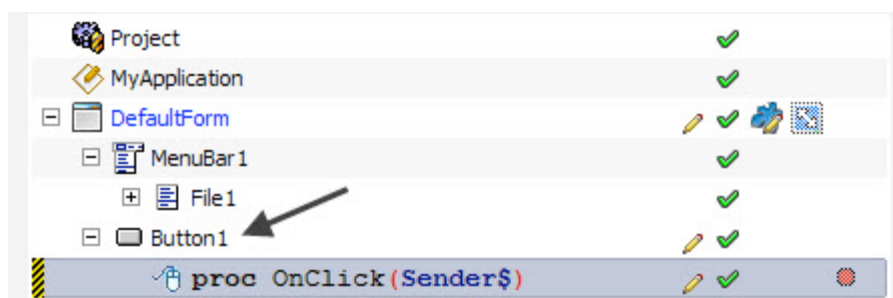


Figure 330: Create OnClick event

- Drag a **PDirectory** object from the File section of **Components Pane** to the **OnClick** event. This would declare a new **PDirectory** object.

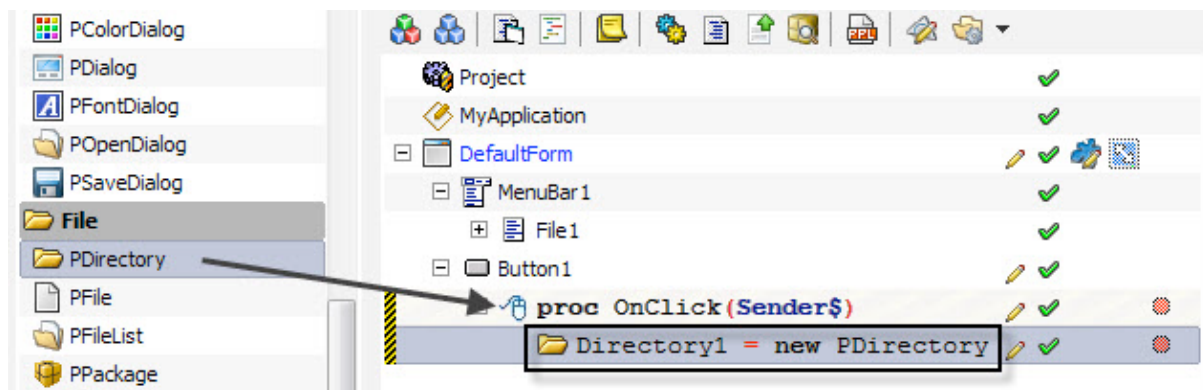


Figure 331: Drag PDirectory

- Change the **PathName** property of **PDirectory** to point to the directory you want. For our example, we will point it to the default PPL folder.

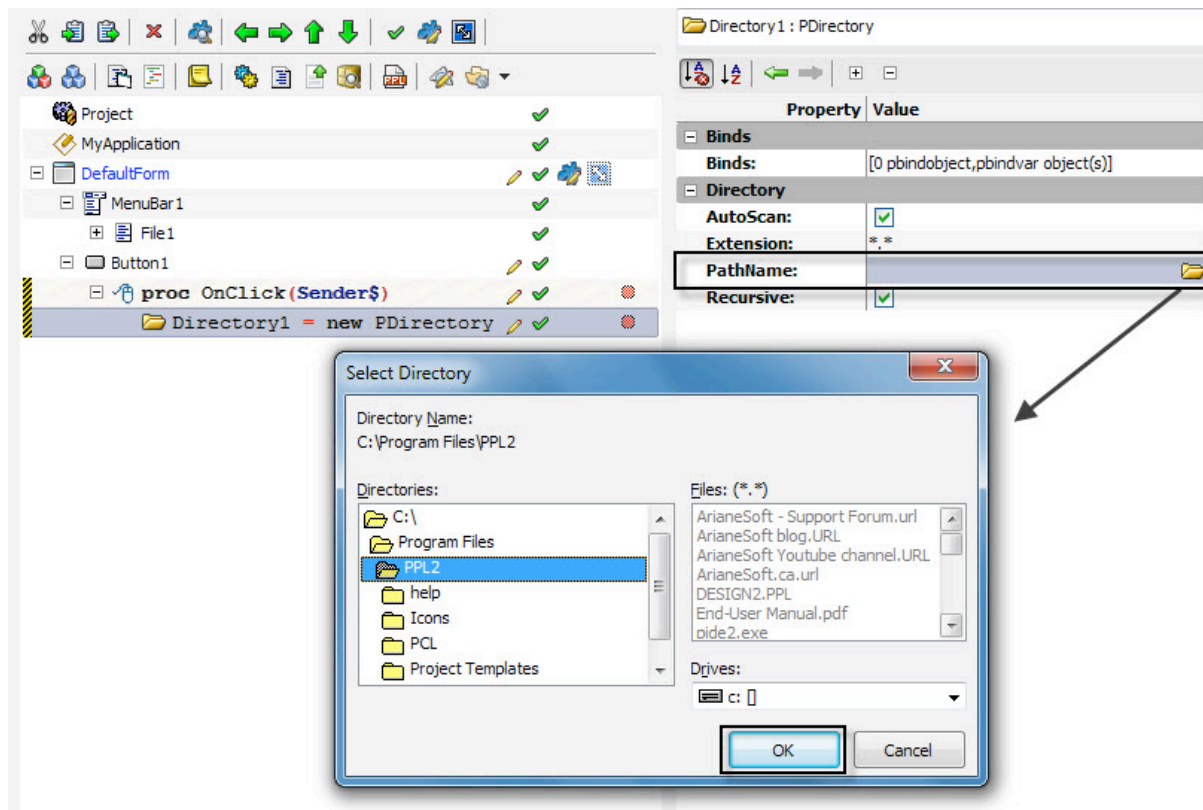


Figure 332: Change PathName

- The **Extension** property allows us to select the files we want to grab with the help of **PDirectory** object. While you can keep the **Extension** Property pointing to \*.\* to account for every file, you can also change this property to account for specific type of files. In our example we will change this to \*.url. This would ensure that only files with .url extension are taken by the **PDirectory**.

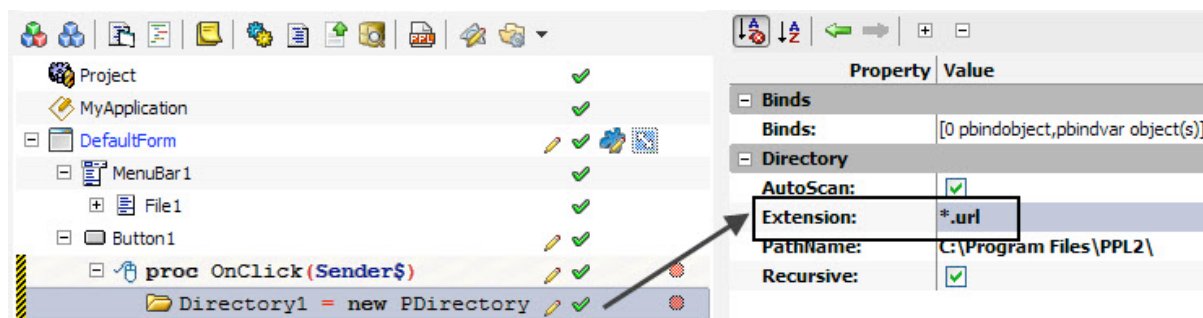


Figure 333: Change Extension property

- While the **PDirectory** object is still selected, press **Ctrl+Space** to bring code complete window and select **ShowMessage** from it.

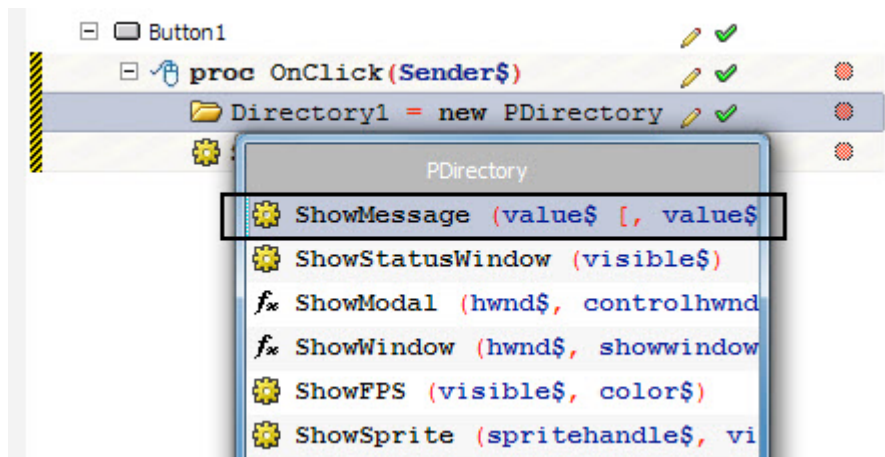


Figure 334: ShowMessage

- While **ShowMessage** object is still selected, drag the **PDirectory** object to its **Value** property. This would initiate a **code complete** window.

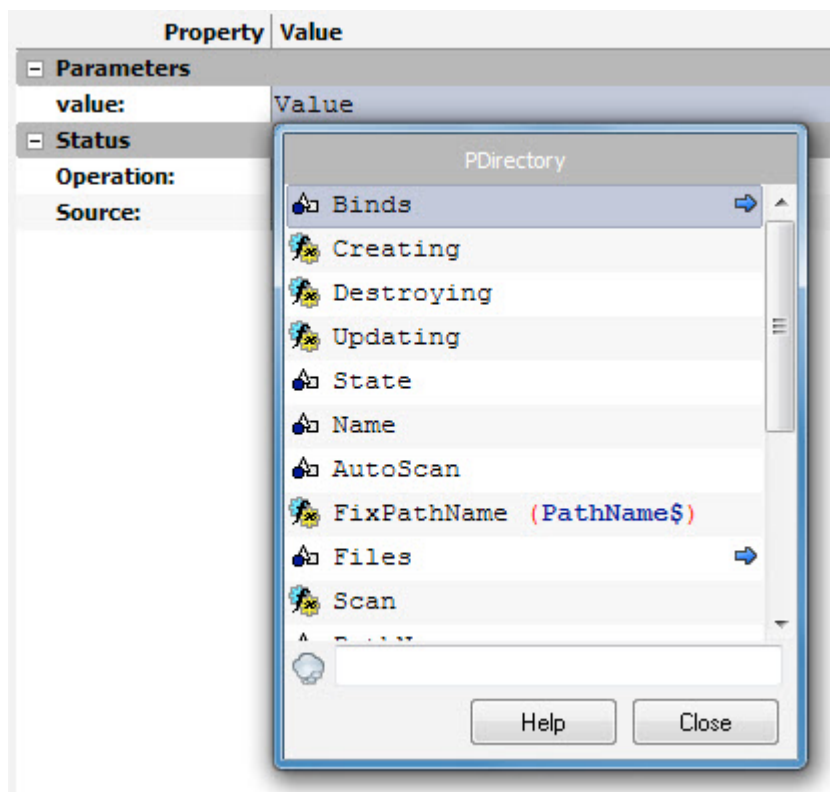


Figure 335: Code Complete window

- In the **code complete** window, go to **Files** and click the arrow beside it.

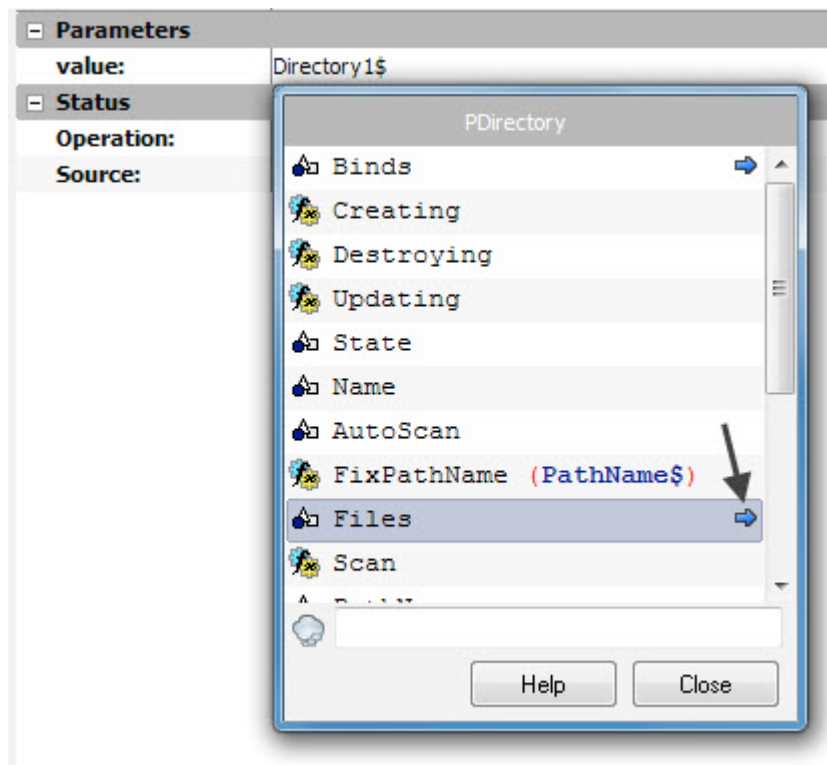


Figure 336: Click the arrow

- Now in the code complete window, select **Count** property. After you have selected **Count** property, the Value property of **ShowMessage** should look like **Directory1.Files.Count**

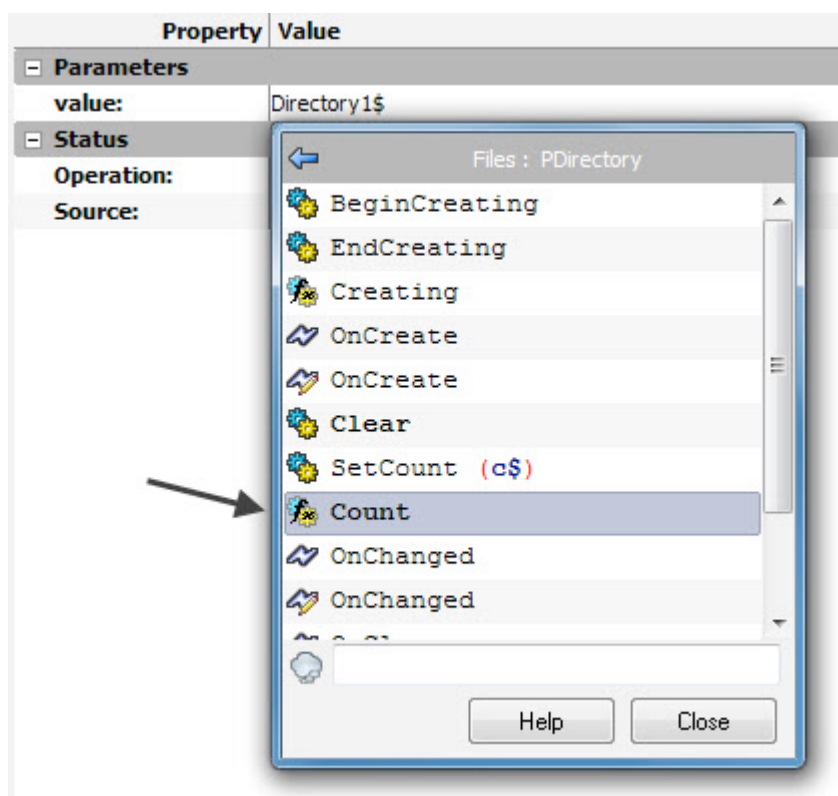


Figure 337: Select Count Property



- That's it! The application is complete. Now you can save the project by going to **File>Save Project** and save it anywhere on the computer.

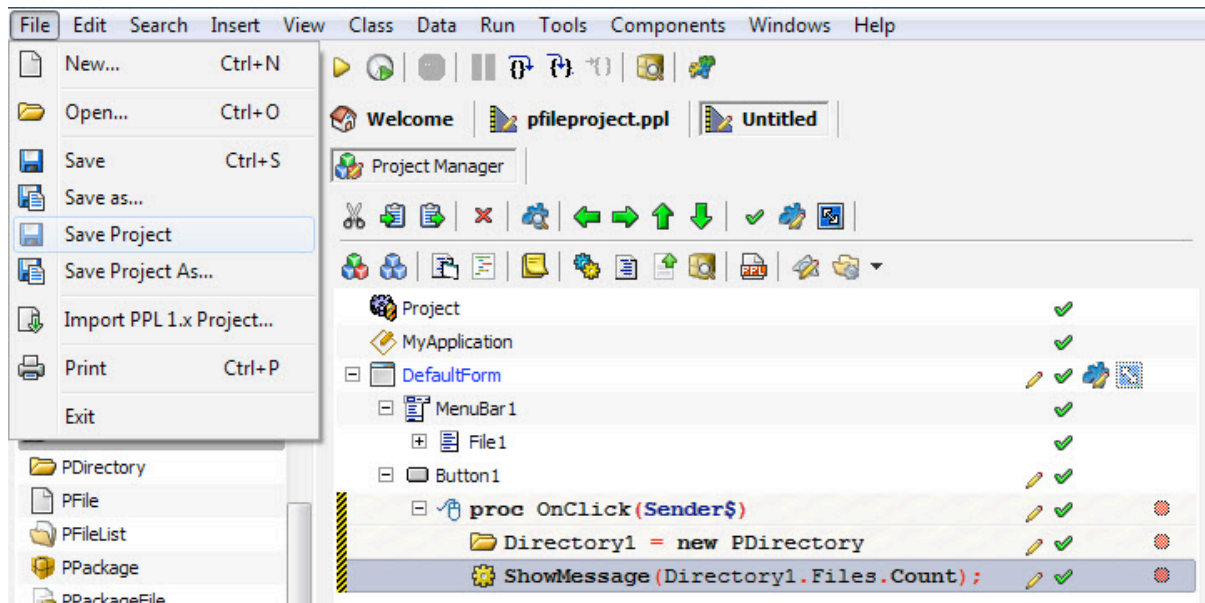


Figure 338: Save Project

- Press **F5** to see the result!

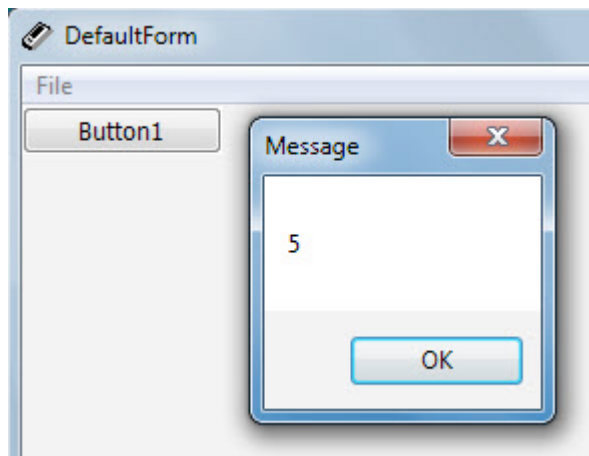


Figure 339: Output

## Getting the file size of a file by using PFile object

- Start by creating a new **Desktop Form** project. For doing this, press **Ctrl+N** or Go to **File>New** and select **Desktop Form** from **Select New Project Type...** window.

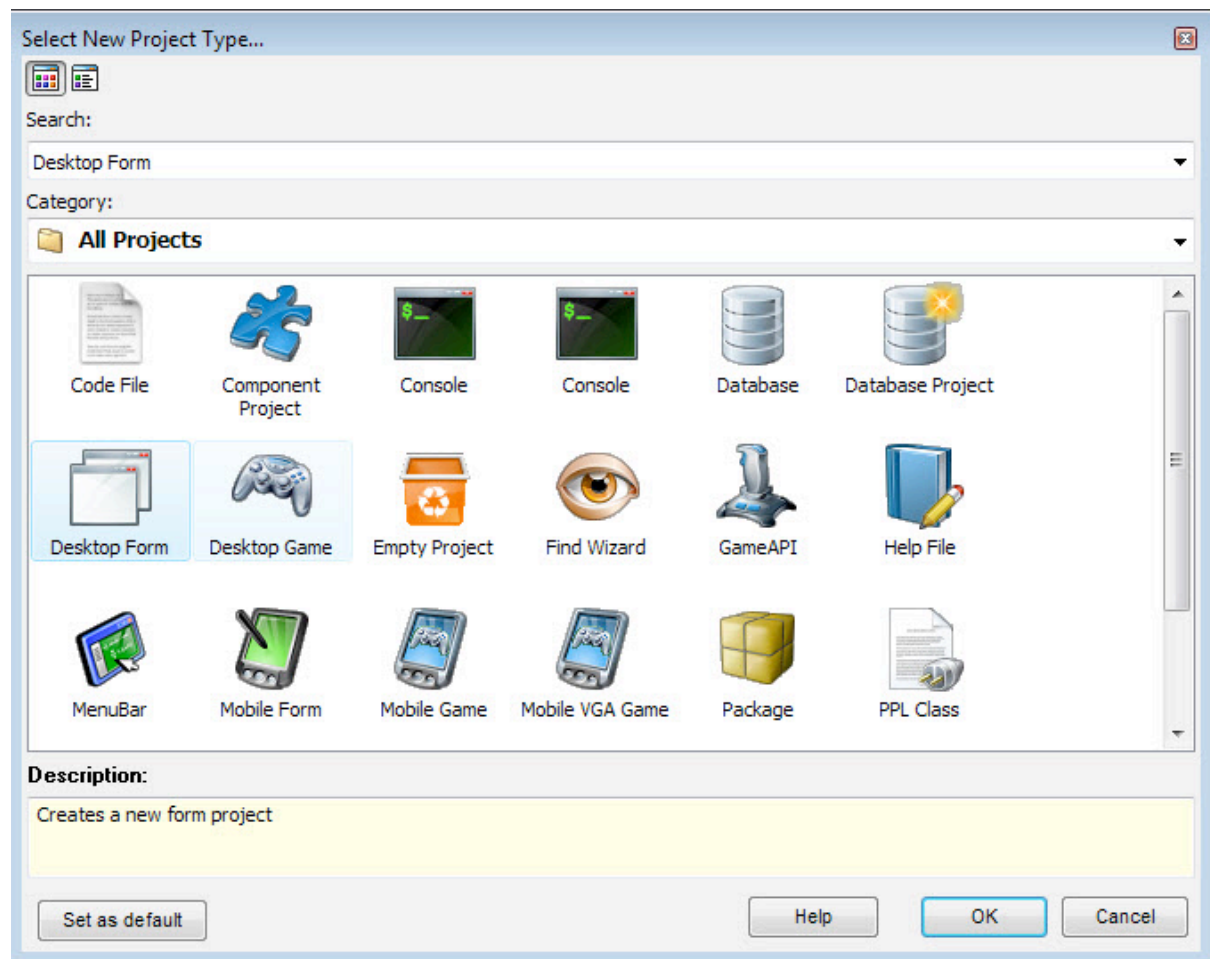


Figure 340: Create New Project

- Drag a **PButton** object on the **Default Form** and double click it to create an **OnClick** event which would be used to initiate all the actions on our application. Alternatively, for creating an **OnClick** event you can also right click the **PButton** object and select **OnClick** from the **Events** sub menu.

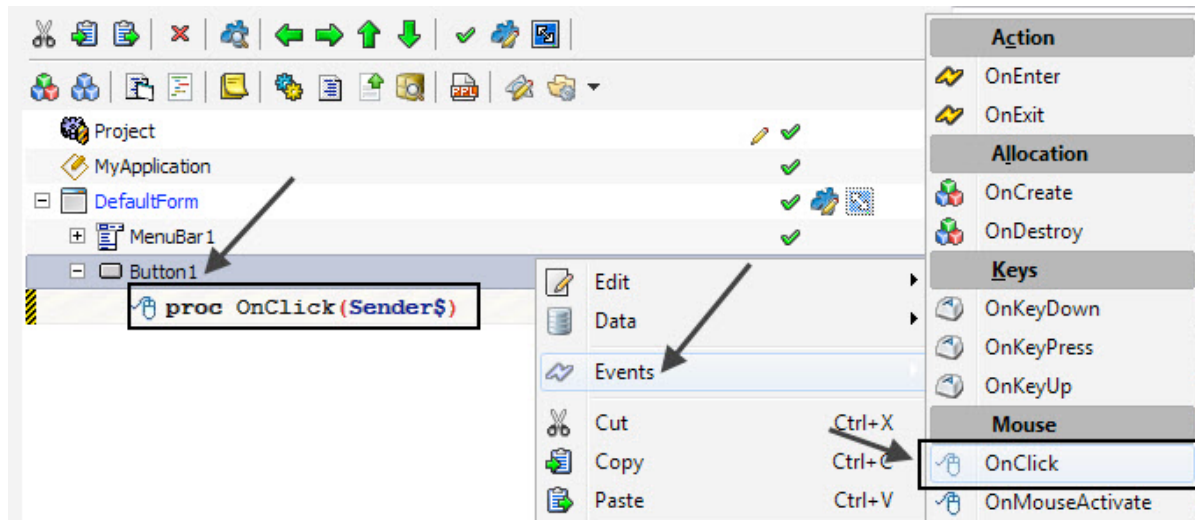


Figure 341: Create OnClick event

- Drag an **PEdit** object to the project above the **PButton** object you just created. This **PEdit** box will help us point to the path of the file we want to the size of.

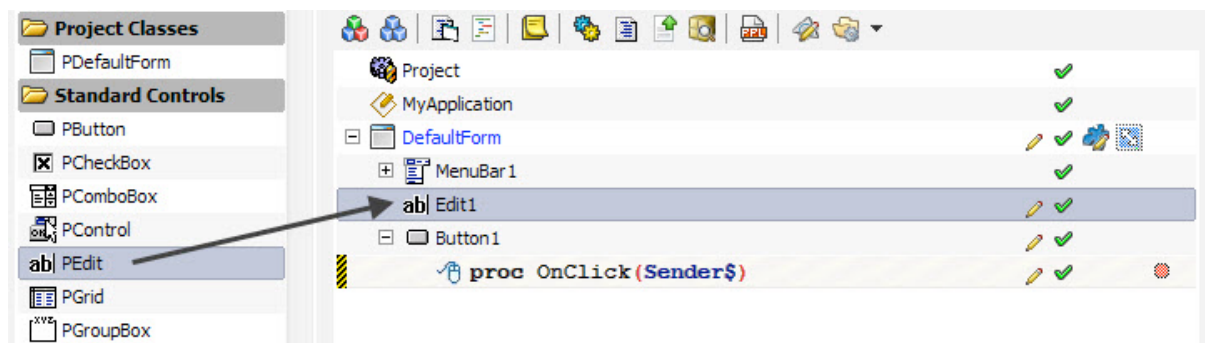


Figure 342: Drag PEdit

- Now we would need to store the address of the file input by the user on the **PEdit** object in a variable so that it can be used. For doing this, drag a **PVariable** object to the **OnClick** event. This would create the definition of a new variable.

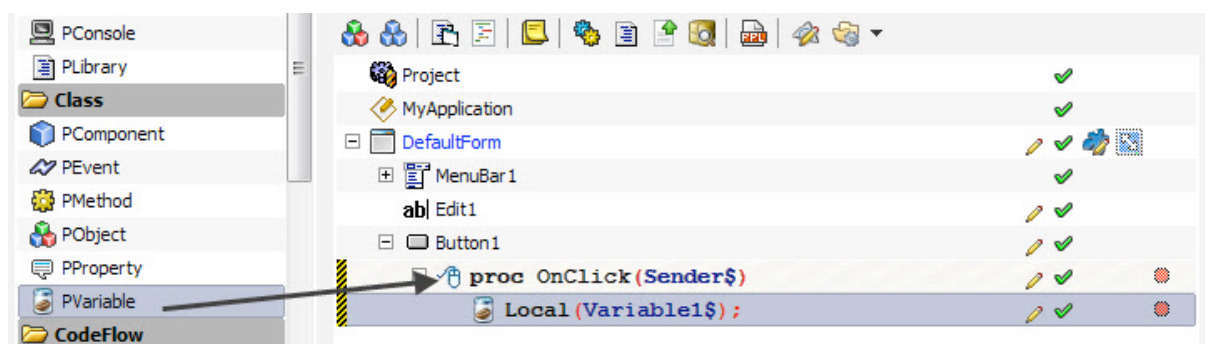


Figure 343: Drag PVariable

- Drag the newly created Variable definition back to the **OnClick** procedure while holding the **ALT** key on the keyboard to create the variable.

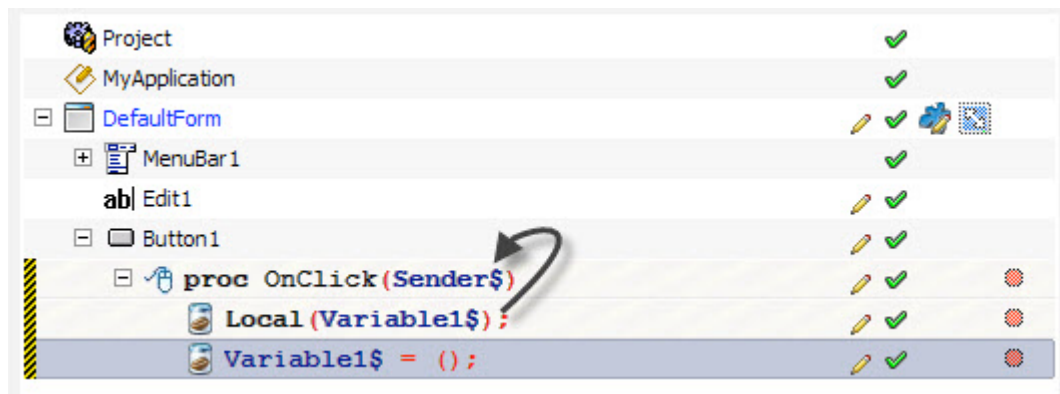


Figure 344: Create Variable

- Drag the **PEdit** object to the **Expr** property of the newly created variable. This would initiate a **Code Completion** window.

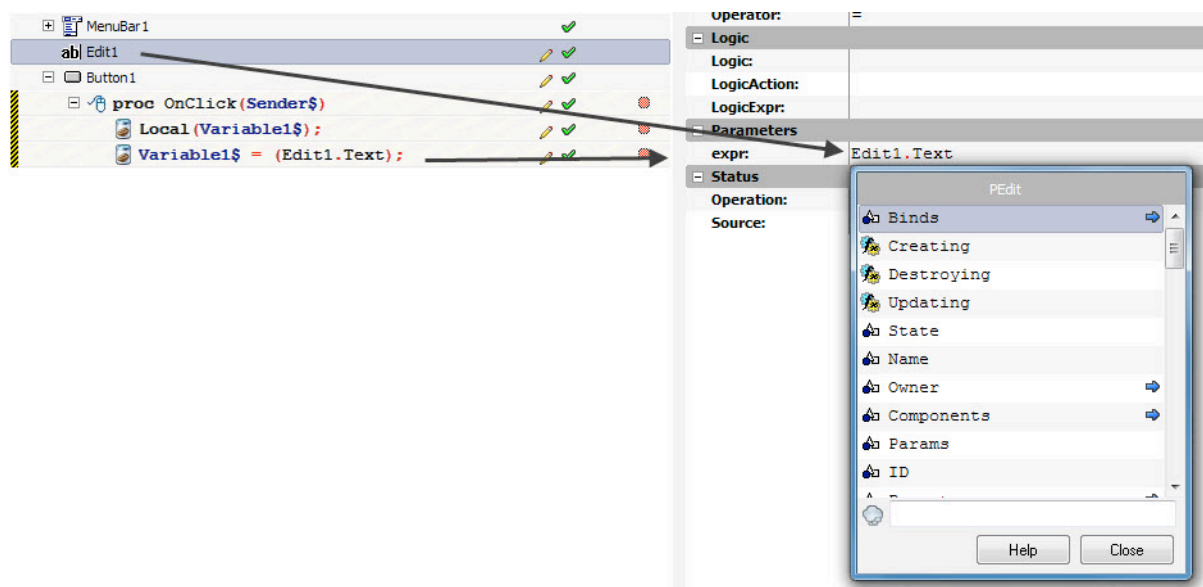


Figure 345: Drag PEdit1 to expr

- Select the **Text** property in the code complete window. This would send the text entered in the **PEdit** box to the variable.

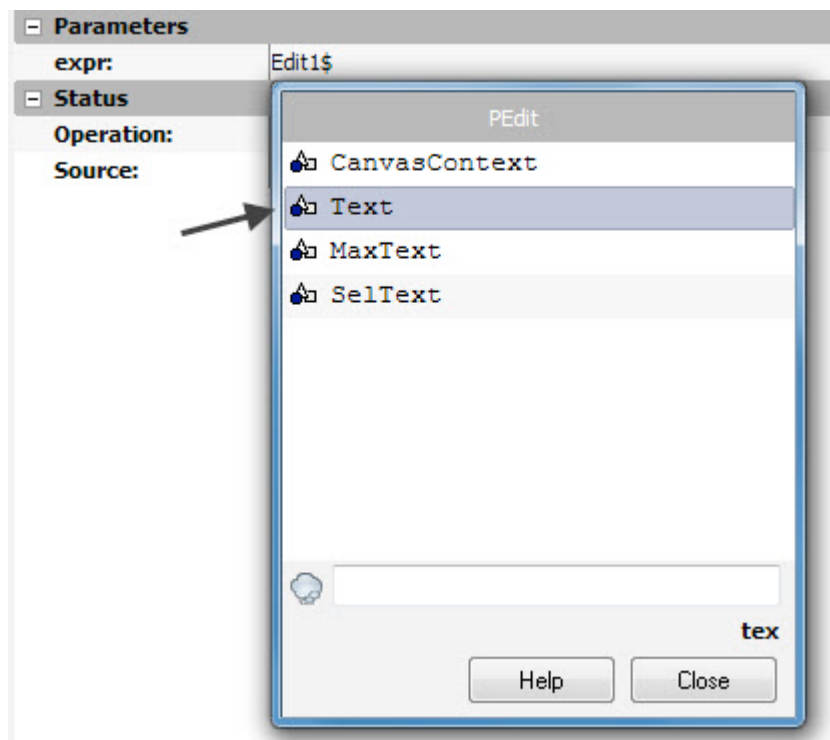


Figure 346: Select Text

- Go to the **Components Panel** and drag **PFile** from the File section to the **OnClick** object.

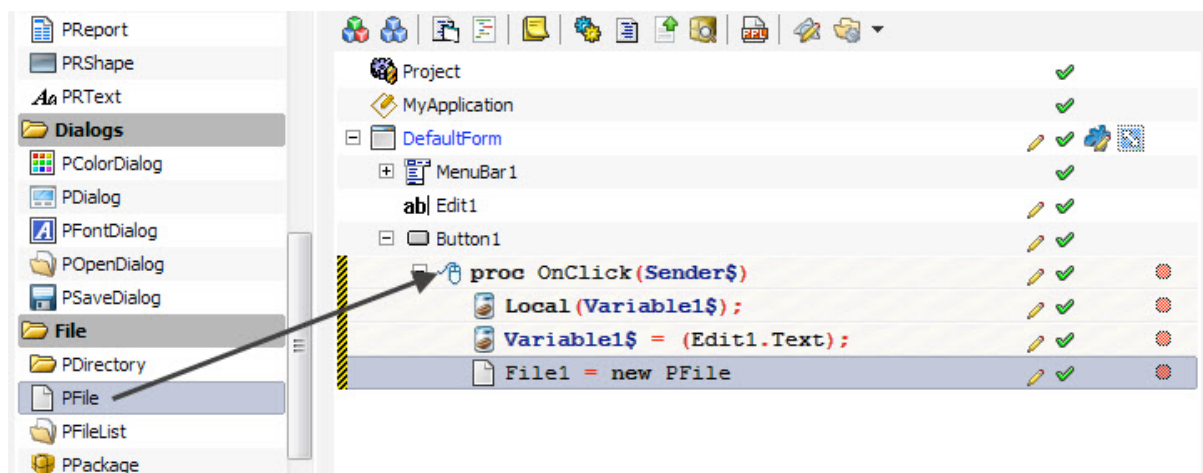


Figure 347: Drag PFile to OnClick

- Click on the **PFile** object and drag **Variable1\$** to its filename property. This would provide the path entered in the **PEdit** box to the **PFile** object.

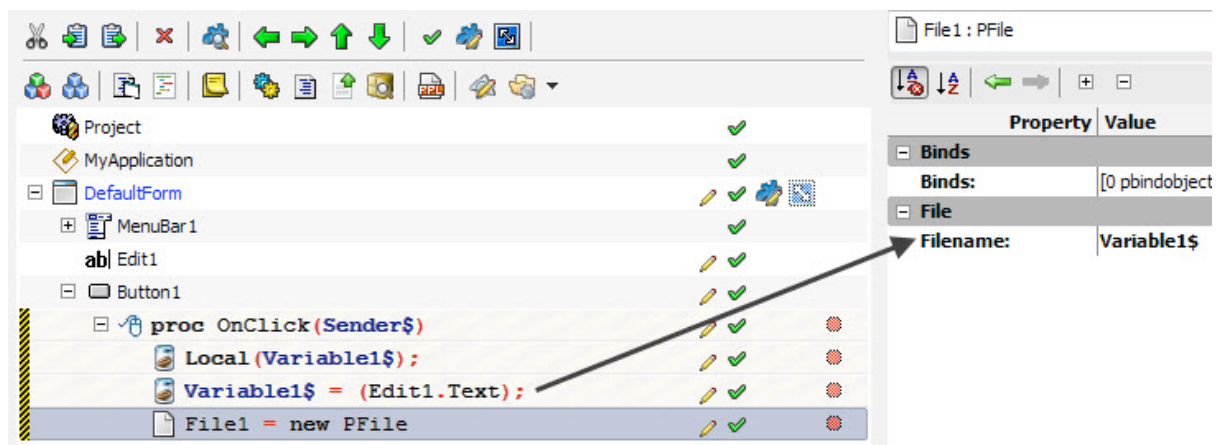
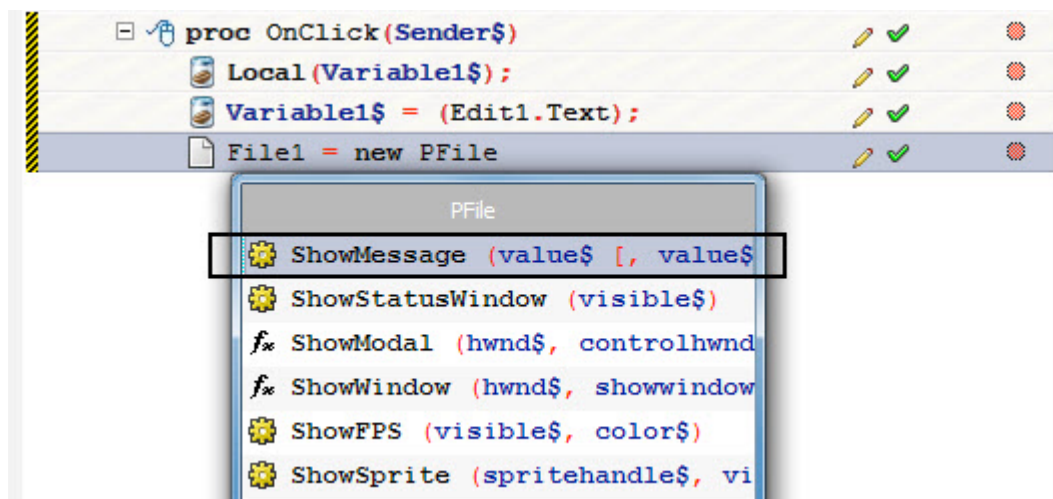


Figure 348: Drag variable to Filename

- Now, for displaying the file size, Press **Ctrl+Space** on the keyboard while **PFile** object is still selected. This would generate a code complete window. Select **ShowMessage** in it.



- Select the **ShowMessage** statement and drag **PFile** statement to its value property. In the **code complete** window that initiates, select the **Size** property. This would allocate the size of the File to the message box.

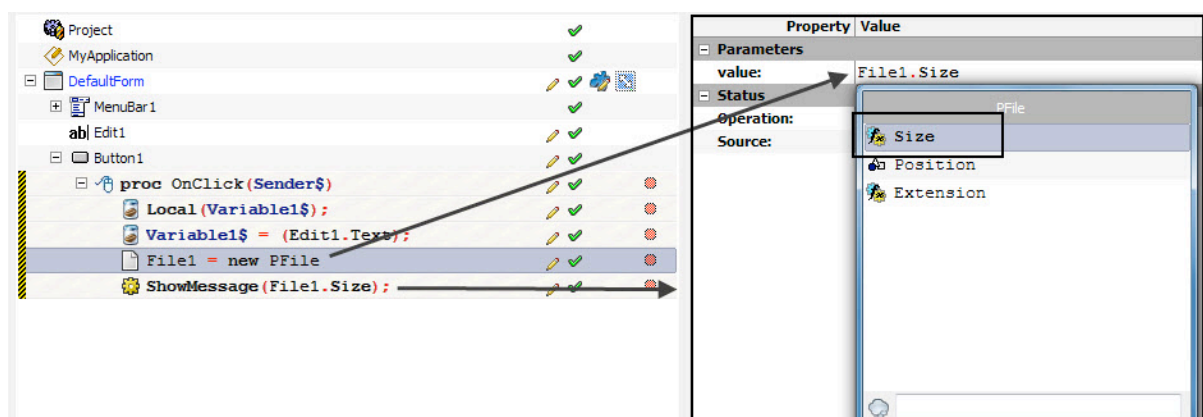


Figure 349: Change Value property



- That's it! The application is complete. Now you can save the project by going to File>Save Project and save it anywhere on the computer.

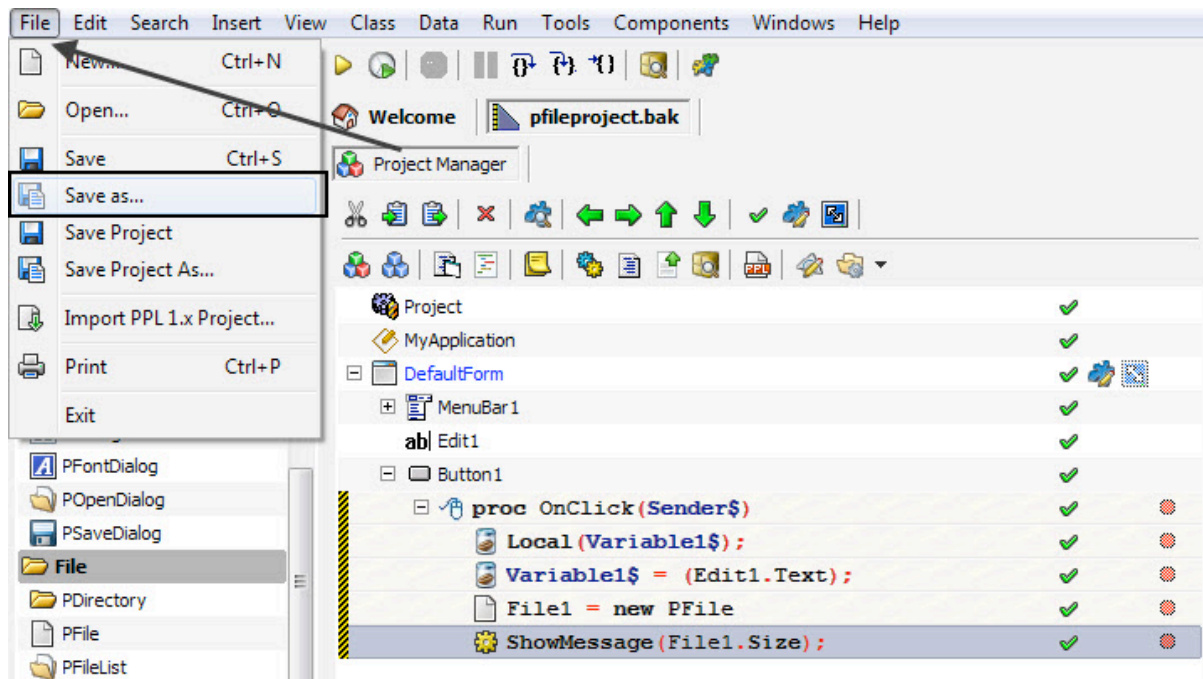


Figure 350: Save Project

- Press **F5** to see the result!

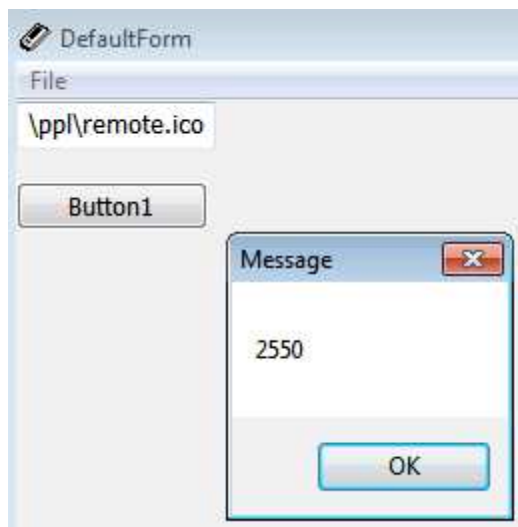


Figure 351: Output

## Using **PFileList** to display the constituents of a **Folder**

**PFileList** is an object that is used inside the **PFolder** and contains a list of **PFile** items that the **PFolder** object contains. With the help of **PFileList**, actions can be performed on the while list of items. Given below is an example that would demonstrate the use of **PFileList** object to delete all the objects present in the **PFolder**. Using a **ForEach** object along with the file property of **PDirectory** we can get the **PFileList** object and work with various methods on the files; the example given below will demonstrate how to delete all the files in a directory:

- Create a new **Desktop Form** project in PIDE. For doing this, go to **File>New** or press **Ctrl+N** to get the **Select New Project Type...** window. In the **Select New Project Type** window, select **Desktop Form** project and press **Ok**.

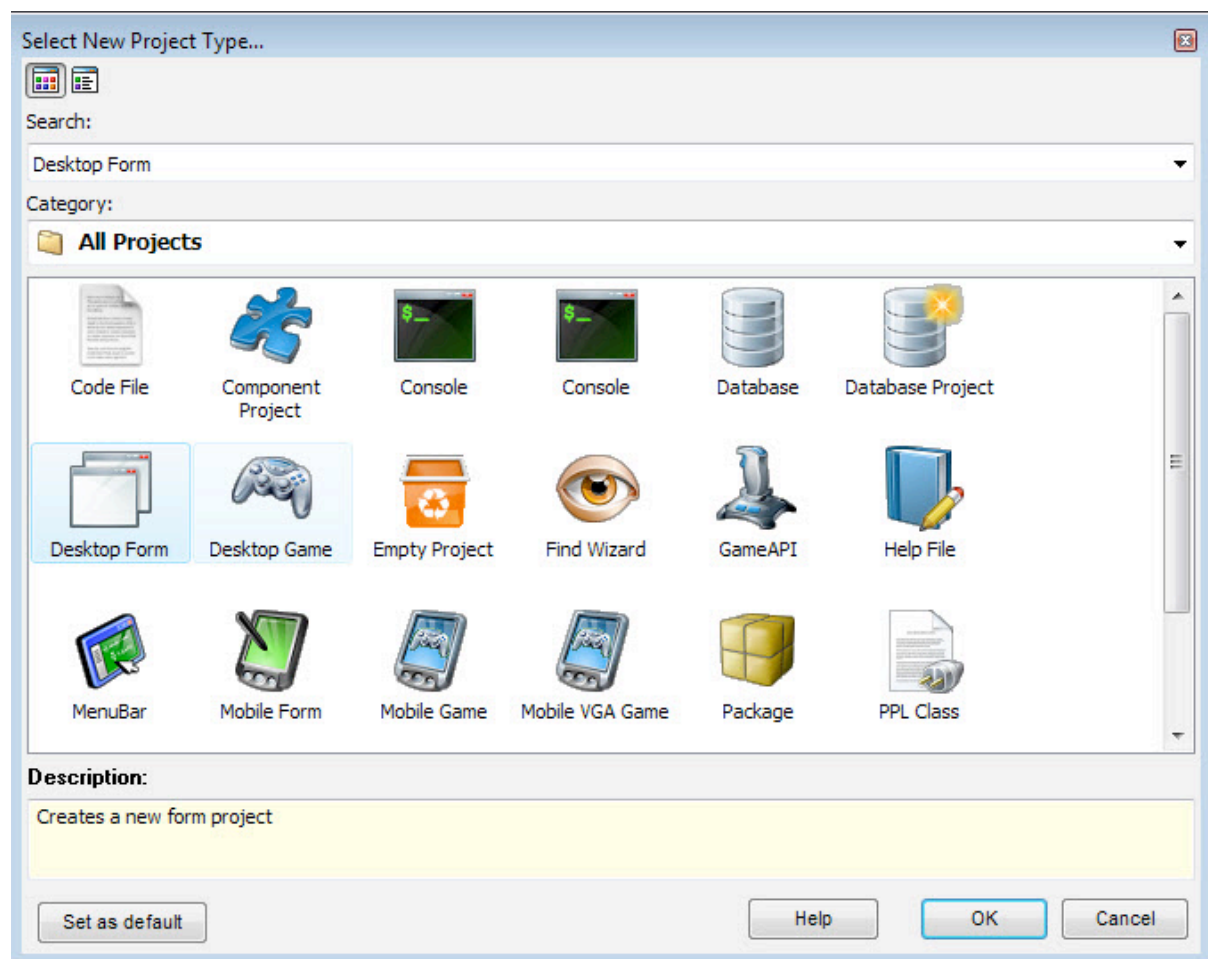


Figure 352: Create New Project

- After selecting the **Desktop Form**, Drag a **PButton** to the **Project Manager**, this will allow us to trigger the actions on the click on the button.

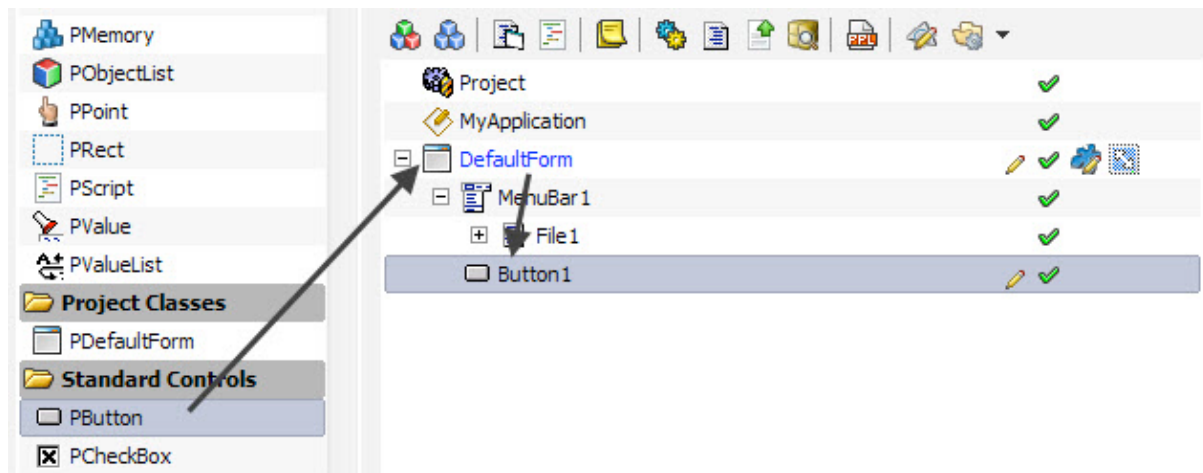


Figure 353: Drag PButton to DefaultForm

- Double click the **PButton** to create an **OnClick** event. This event will allow us to raise all the actions we want.

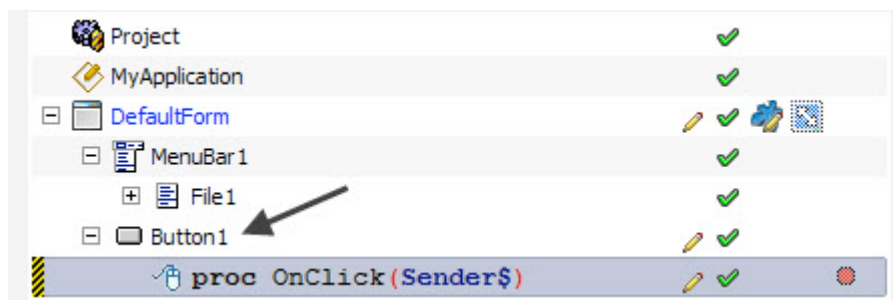


Figure 354: Create OnClick event

- Drag a **PDirectory** object from the File section of **Components Pane** to the **OnClick** event. This would declare a new **PDirectory** object.

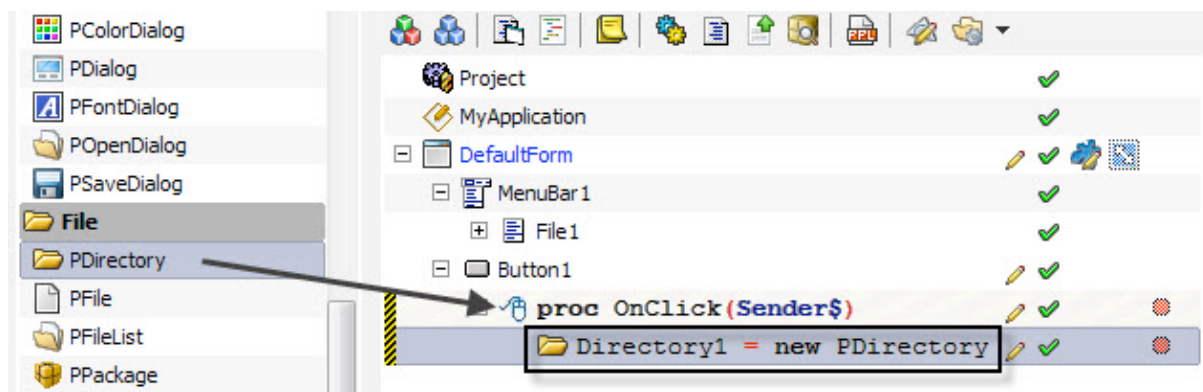


Figure 355: Drag PDirectory

- Change the **PathName** property of **PDirectory** to point to the directory you want. For our example, we will point it to a dummy folder that contains various files.

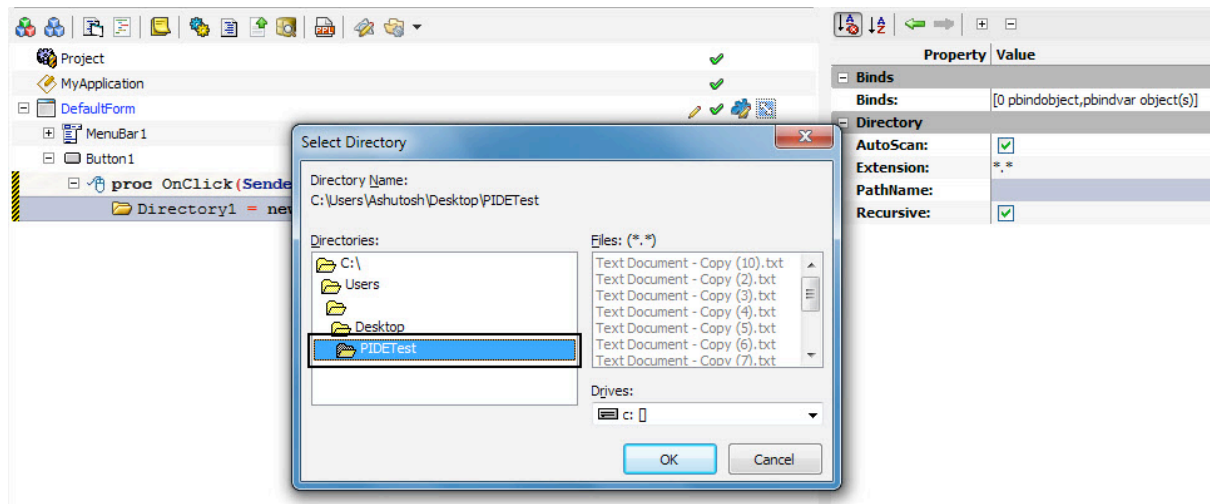


Figure 356: Change PathName

- Since we are going to delete files, it would be clever to filter the files that we want to delete. In the Extension property of **PDirectory** object, set \*.txt so that all the text files are targeted.

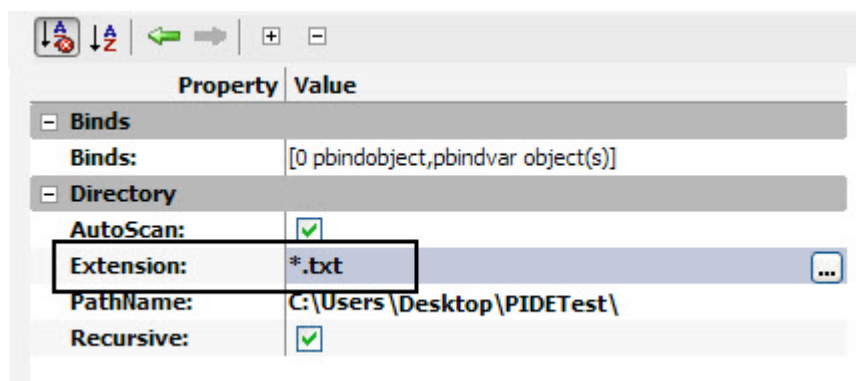


Figure 357: Change Extension Property

- Now that the **PDirectory** object is configured, we can continue with the **PFileList** object that is represented by the **Files** method. For deleting all the files in a directory specified in the **PDirectory** object all we need to do is to call the **PFileList** object that will scan all the files in the **PDirectory** and delete all the files in there. For doing this, select the **PDirectory** object and press **Ctrl+Space** bar on the keyboard, this will bring the **Code Completion** menu.

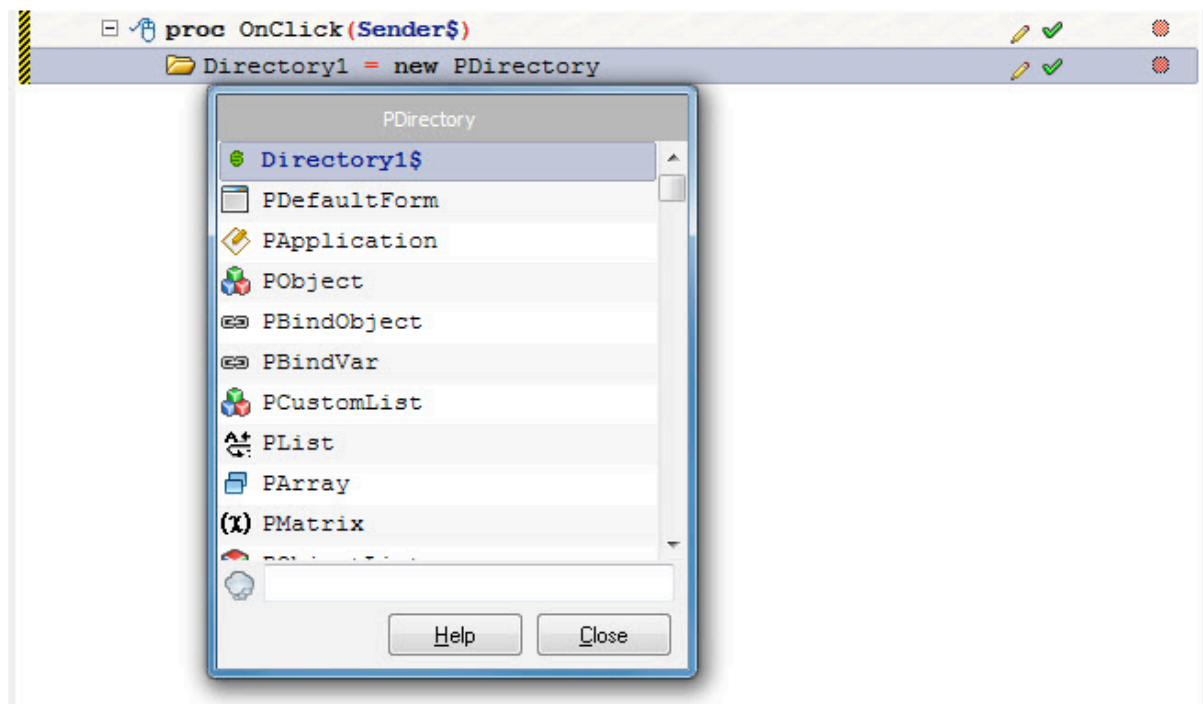


Figure 358: Hit Ctrl+Space for Code Complete window

- Select **Files** menu item or write Files in the text box to go straight to it. Once on it, press the arrow beside it to go to a sub method.

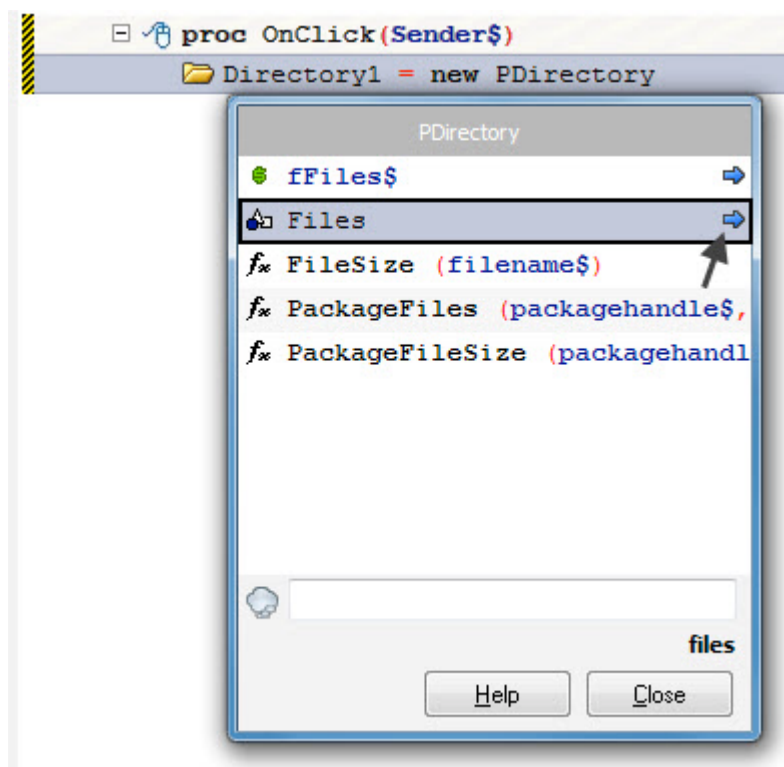


Figure 359: Select Files

- In the **PDirectory.Files** menu, select **DeleteFiles** method. This method will delete all the files available in the directory that is specified in the **PDirectory**.

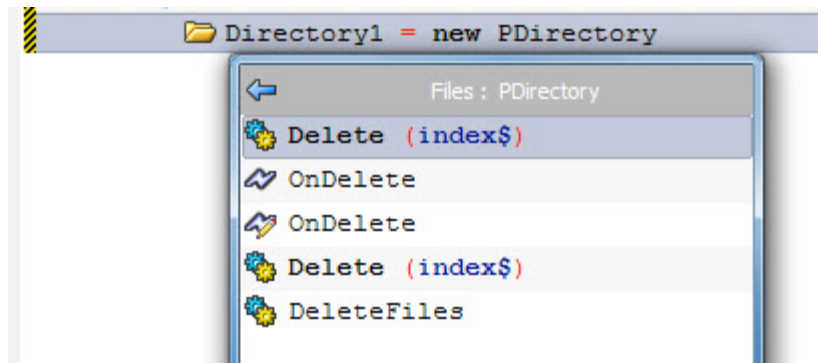


Figure 360: Select DeleteFiles

- After you have selected the **DeleteFiles** method, the **Project Manager** should look like the screenshot given below.

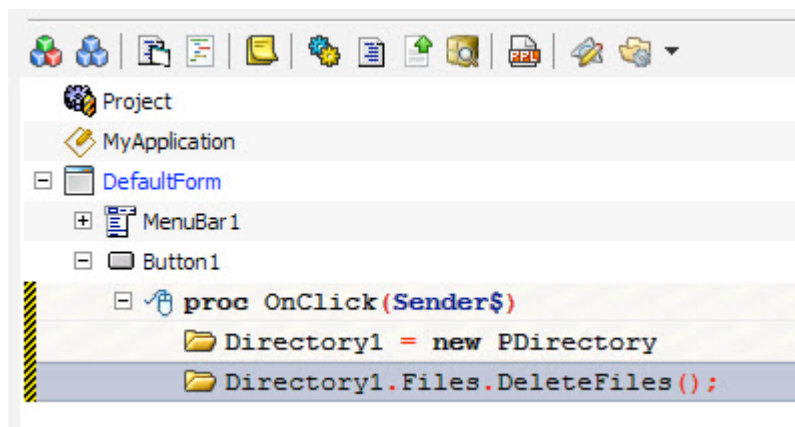


Figure 361: Project screenshot

- The application that can delete the text files from a directory is complete. Now you can save the project by going to **File>Save Project** and save it anywhere on the computer.



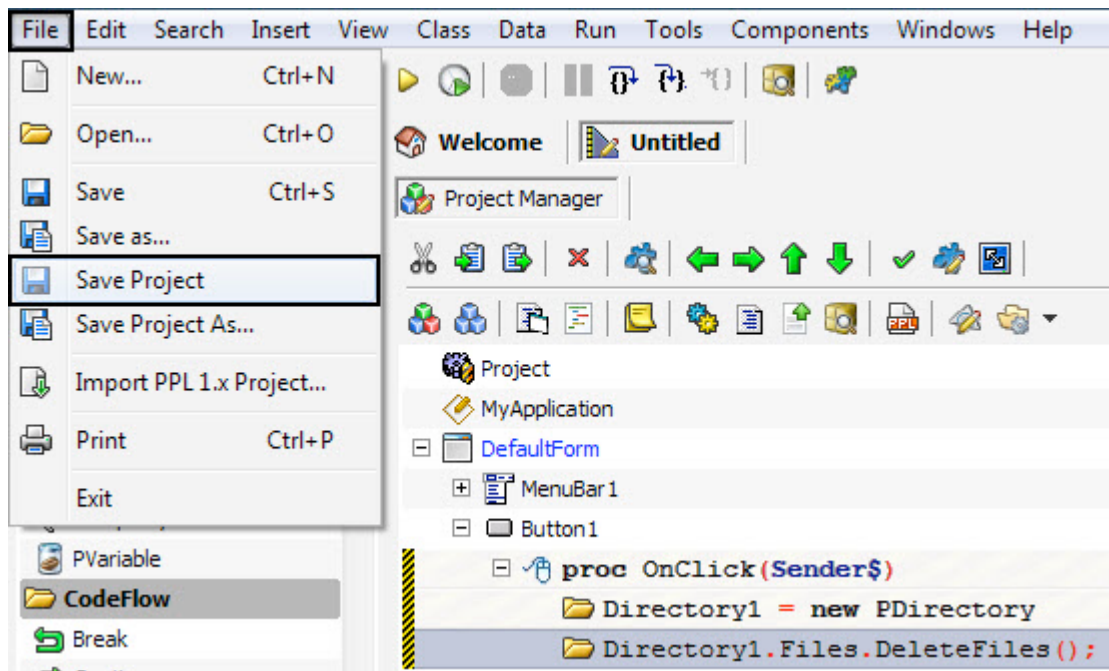


Figure 362: Save Project

- Press **F5** to see the result! Below screenshot shows that the folder that had many text files before became empty when we clicked the button.

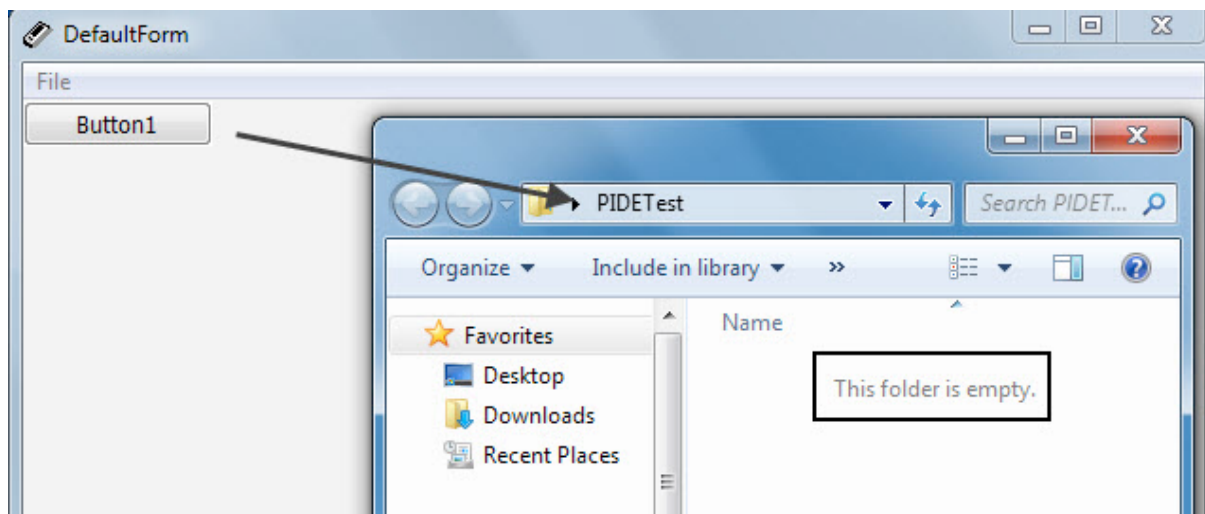


Figure 363: Output

## Presource

**Presource** object can be used to store information from a file directly in the PIDE code. Dragging a file from your computer to the **Project Manager** shows **Presource** in the context window. By selecting **Presource** in the context window, we can mark an object as the resource that is needed for the execution of a program.

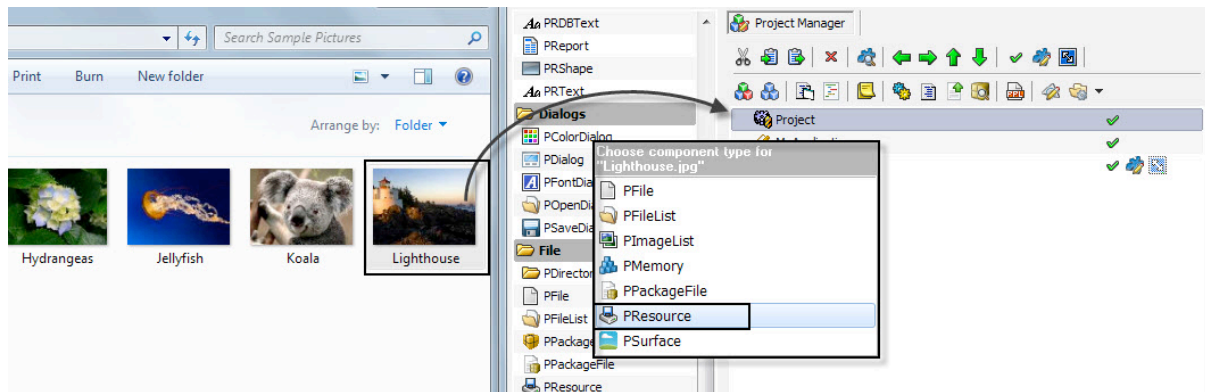


Figure 364: Drag and drop files and mark PResource

A popular example of using PResource would be of using it to store music as well as sound files that are used in applications for button clicks etc.

## Additional Support

For additional support, including discussion boards, frequently asked questions, and a knowledge base, please visit the <http://www.arianesoft.ca/> website.

Know more about PIDE and what you can do from it, Visit <http://ppl-lang.com/>

Get all the support you want from your peers, visit <http://forum.arianesoft.ca/>

Watch n' learn! Visit the YouTube channel at <http://www.youtube.com/arianesoft>

## Appendix A: Figures

|   |    |
|---|----|
| Figure 1: Creating a new project.....           | 5  |
| Figure 2: Code.....                             | 6  |
| Figure 3: Output.....                           | 6  |
| Figure 4: New Procedural project.....           | 7  |
| Figure 5: Main Library.....                     | 7  |
| Figure 6: Hello World Code .....                | 8  |
| Figure 7: Run Hello World .....                 | 8  |
| Figure 8: Run the project .....                 | 8  |
| Figure 9: Output.....                           | 8  |
| Figure 10: Create a new project.....            | 9  |
| Figure 11: OnStart event.....                   | 9  |
| Figure 12: Code Completion window.....          | 10 |
| Figure 13: Expr property.....                   | 10 |
| Figure 14: Output.....                          | 10 |
| Figure 15: New Project .....                    | 11 |
| Figure 16: Right click to Edit Form.....        | 12 |
| Figure 17: Caption Property .....               | 12 |
| Figure 18: Right click to open DefaultForm..... | 12 |
| Figure 19: Code.....                            | 13 |
| Figure 20: New Project .....                    | 14 |
| Figure 21: Right click to Edit Form.....        | 15 |
| Figure 22: Place components .....               | 16 |
| Figure 23: Change Caption property.....         | 16 |
| Figure 24: Final placement .....                | 17 |
| Figure 25: PButton.....                         | 17 |
| Figure 26: Project Manager .....                | 18 |

|   |    |
|---|----|
| Figure 27: OnClick event.....                         | 18 |
| Figure 28: Drag a PVariable .....                     | 18 |
| Figure 29: Drag PEdit to expr.....                    | 19 |
| Figure 30: Steps from 1 to 4 .....                    | 20 |
| Figure 31: Drag operation.....                        | 20 |
| Figure 32: Change property .....                      | 21 |
| Figure 33: Change Expr property .....                 | 21 |
| Figure 34: Condition .....                            | 22 |
| Figure 35: ShowMessage in Code Completion window..... | 22 |
| Figure 36: Change Value property.....                 | 22 |
| Figure 37: Output.....                                | 23 |
| Figure 38: Create New Project.....                    | 24 |
| Figure 39: Drag and Drop PPackage .....               | 25 |
| Figure 40: Change Property .....                      | 25 |
| Figure 41: Drop Files.....                            | 26 |
| Figure 42: Expand for constituents .....              | 26 |
| Figure 43: Run the Project.....                       | 27 |
| Figure 44: Creating new project .....                 | 28 |
| Figure 45: Drag a database .....                      | 29 |
| Figure 46: Drag a PTable.....                         | 29 |
| Figure 47: Specify a filename.....                    | 30 |
| Figure 48: Give TableName.....                        | 30 |
| Figure 49: Add PField object.....                     | 31 |
| Figure 50: Specify Property .....                     | 31 |
| Figure 51: Creating Database.....                     | 32 |
| Figure 52: Viewing Table .....                        | 32 |
| Figure 53: Drag PQuery.....                           | 33 |
| Figure 54: Change Database property.....              | 33 |
| Figure 55: Visual query editor .....                  | 34 |

|  |    |
|--|----|
| Figure 56: Insert PDBGird.....                           | 34 |
| Figure 57: Drag table to form .....                      | 35 |
| Figure 58: Smart move.....                               | 35 |
| Figure 59: Drag PQuery.....                              | 36 |
| Figure 60: Select appropriate database name.....         | 36 |
| Figure 61: Double click PQuery .....                     | 37 |
| Figure 62: The four buttons of visual query editor ..... | 37 |
| Figure 63: Save file button.....                         | 38 |
| Figure 64: The SQL Tab .....                             | 38 |
| Figure 65: Results Tab .....                             | 39 |
| Figure 66: Selecting Query Type.....                     | 39 |
| Figure 67: Creating Advanced Query .....                 | 40 |
| Figure 68: New Project .....                             | 41 |
| Figure 69: Drag a PForm .....                            | 42 |
| Figure 70: Create new project.....                       | 42 |
| Figure 71: Get database .....                            | 43 |
| Figure 72: Create new project.....                       | 44 |
| Figure 73: Database Project.....                         | 44 |
| Figure 74: Create new project.....                       | 46 |
| Figure 75: Form Editor .....                             | 46 |
| Figure 76: Drag PGrid .....                              | 47 |
| Figure 77: Text Property.....                            | 47 |
| Figure 78: Project Manager .....                         | 48 |
| Figure 79: Events Pane .....                             | 48 |
| Figure 80: Create event.....                             | 48 |
| Figure 81: Drag PVariable .....                          | 49 |
| Figure 82: Drag For Object.....                          | 49 |
| Figure 83: Set For properties .....                      | 50 |
| Figure 84: Change Property .....                         | 50 |



|  |    |
|--|----|
| Figure 85: Alt + Drop .....                        | 51 |
| Figure 86: Change Expr .....                       | 51 |
| Figure 87: Drop a For loop .....                   | 51 |
| Figure 88: Change Property .....                   | 52 |
| Figure 89: Change Property .....                   | 52 |
| Figure 90: Code Completion window .....            | 53 |
| Figure 91: Select option .....                     | 53 |
| Figure 92: Change property .....                   | 54 |
| Figure 93: Drag RowTotal\$ .....                   | 54 |
| Figure 94: Drag and Drop .....                     | 55 |
| Figure 95: Drag operation .....                    | 55 |
| Figure 96: Change Row property .....               | 56 |
| Figure 97: Change Property .....                   | 56 |
| Figure 98: CHange Value property .....             | 56 |
| Figure 99: Save Project .....                      | 57 |
| Figure 100: Output .....                           | 57 |
| Figure 101: Create new project .....               | 58 |
| Figure 102: Drag Database .....                    | 59 |
| Figure 103: Database with tables and columns ..... | 59 |
| Figure 104: Drop database to Default Form .....    | 60 |
| Figure 105: Smart Move .....                       | 60 |
| Figure 106: Change properties .....                | 60 |
| Figure 107: Save project .....                     | 61 |
| Figure 108: Output .....                           | 61 |
| Figure 109: Create new project .....               | 62 |
| Figure 110: PReport .....                          | 63 |
| Figure 111: PReport in existing project .....      | 63 |
| Figure 112: PReport with Database .....            | 64 |
| Figure 113: Create new project .....               | 65 |

|  |    |
|--|----|
| Figure 114: Drag PList.....                            | 66 |
| Figure 115: Create Event .....                         | 66 |
| Figure 116: Drag PList.....                            | 67 |
| Figure 117: Select Add in Code Complete window .....   | 68 |
| Figure 118: CHange Property .....                      | 68 |
| Figure 119: Change Expr Property .....                 | 69 |
| Figure 120: Add List.....                              | 69 |
| Figure 121: Drag with ALT key .....                    | 70 |
| Figure 122: Code Completion window.....                | 70 |
| Figure 123: Sort list .....                            | 71 |
| Figure 124: Code Complete window .....                 | 72 |
| Figure 125: The value property.....                    | 72 |
| Figure 126: Change the value property.....             | 73 |
| Figure 127: Add Value.....                             | 73 |
| Figure 128Save Project.....                            | 74 |
| Figure 129: Output.....                                | 74 |
| Figure 130: Create new project.....                    | 75 |
| Figure 131: OnCreate event.....                        | 76 |
| Figure 132: PValue .....                               | 76 |
| Figure 133: Value property.....                        | 77 |
| Figure 134: Change Expr property .....                 | 77 |
| Figure 135: Select ShowMessage .....                   | 77 |
| Figure 136: Change UpperCase from Code Completion..... | 78 |
| Figure 137: Save Project.....                          | 78 |
| Figure 138: Output.....                                | 79 |
| Figure 139: Create New Project.....                    | 80 |
| Figure 140: Drag ValueList .....                       | 81 |
| Figure 141: Create OnCreate Event.....                 | 81 |
| Figure 142: Code Completion window.....                | 82 |

|   |    |
|---|----|
| Figure 143: Select Add .....                        | 83 |
| Figure 144: Change Value Property .....             | 83 |
| Figure 145: Change Value Property .....             | 84 |
| Figure 146: Project View .....                      | 84 |
| Figure 147: Drag ForEach .....                      | 85 |
| Figure 148: Drag PValueList .....                   | 85 |
| Figure 149: Select Items .....                      | 86 |
| Figure 150: Code completion window .....            | 86 |
| Figure 151: Select Items .....                      | 87 |
| Figure 152: Select Uppercase .....                  | 87 |
| Figure 153: Code Complete Window .....              | 88 |
| Figure 154: Select PrintLine .....                  | 89 |
| Figure 155: Change Value property .....             | 89 |
| Figure 156: Change the Calue property .....         | 90 |
| Figure 157: Save project .....                      | 90 |
| Figure 158: Output .....                            | 91 |
| Figure 159: Create new project .....                | 92 |
| Figure 160: Delete Object .....                     | 93 |
| Figure 161: Drag PObject .....                      | 93 |
| Figure 162: Rename .....                            | 93 |
| Figure 163: Drag PProperty to Project Manager ..... | 94 |
| Figure 164: Change PProperty Name .....             | 94 |
| Figure 165: Drag PMethod and rename it .....        | 94 |
| Figure 166: Create Class .....                      | 95 |
| Figure 167: Create New project .....                | 96 |
| Figure 168: Drag Psurface .....                     | 97 |
| Figure 169: Surface Editor .....                    | 97 |
| Figure 170: Drag PSprite .....                      | 98 |
| Figure 171: Select Surface Property .....           | 98 |

|  |     |
|--|-----|
| Figure 172: The events tab.....                                      | 99  |
| Figure 173: Code complete.....                                       | 100 |
| Figure 174: Change Property.....                                     | 100 |
| Figure 175: Create New Project.....                                  | 101 |
| Figure 176: Double click .....                                       | 102 |
| Figure 177: Place components in the form .....                       | 102 |
| Figure 178: Project manager.....                                     | 103 |
| Figure 179: Drag PVariable .....                                     | 103 |
| Figure 180: Drag with ALT key pressed .....                          | 104 |
| Figure 181: For loop .....   | 104 |
| Figure 182: Drag PVariable on OnCreate.....                          | 105 |
| Figure 183: Drag to OnClick .....                                    | 105 |
| Figure 184: Drag variable to start property.....                     | 105 |
| Figure 185: Drag Variable 1\$ to Finish .....                        | 106 |
| Figure 186: Create New Variable .....                                | 106 |
| Figure 187: Change Expr property .....                               | 106 |
| Figure 188: Drag variable on For Loop .....                          | 107 |
| Figure 189: Change Expr property .....                               | 107 |
| Figure 190: Code complete window .....                               | 108 |
| Figure 191: Change Value Property .....                              | 108 |
| Figure 192: Save Project.....  | 108 |
| Figure 193: Output.....  | 109 |
| Figure 194: Create new project.....                                  | 110 |
| Figure 195: Double click on DefaultForm.....                         | 111 |
| Figure 196: Component layout .....                                   | 111 |
| Figure 197: Project manager.....                                     | 112 |
| Figure 198: Drag PVariable to OnClick .....                          | 112 |
| Figure 199: Drop Variable1\$ to OnClick .....                        | 112 |
| Figure 200: Drag PEdit1 object to Expr property of Variable 1\$..... | 113 |

|   |     |
|---|-----|
| Figure 201: Drag While to Project Manager.....          | 113 |
| Figure 202: Change the exper property .....             | 113 |
| Figure 203: Drag PVariable to Project Manager .....     | 114 |
| Figure 204: Drag Variable .....                         | 114 |
| Figure 205: Change Expr .....                           | 114 |
| Figure 206: Drag Variable1\$ with ALT key.....          | 115 |
| Figure 207: Change Expr property .....                  | 115 |
| Figure 208: Drag PLabel.....                            | 115 |
| Figure 209: Drag Variable2\$ to expr property .....     | 116 |
| Figure 210: Save project .....                          | 116 |
| Figure 211: Output.....                                 | 117 |
| Figure 212: Create new project.....                     | 118 |
| Figure 213: Create OnCreate object .....                | 119 |
| Figure 214: Drag PValueList on OnCreate event.....      | 119 |
| Figure 215: Fill items.....                             | 119 |
| Figure 216: Drag ForEach object.....                    | 120 |
| Figure 217: Select Items .....                          | 120 |
| Figure 218: VarObject .....                             | 121 |
| Figure 219: Select PValue .....                         | 121 |
| Figure 220: Drag PValue object to Output Property ..... | 121 |
| Figure 221: Select Show .....                           | 122 |
| Figure 222: Save project.....                           | 122 |
| Figure 223: Output.....                                 | 123 |
| Figure 224: Create new project.....                     | 124 |
| Figure 225: Double click .....                          | 125 |
| Figure 226: Place PEdit .....                           | 125 |
| Figure 227: Place Components on Desktop form.....       | 126 |
| Figure 228: Place PLabel on Desktop Form.....           | 126 |
| Figure 229: Project Manager .....                       | 127 |

|   |     |
|---|-----|
| Figure 230: OnClick event.....                  | 127 |
| Figure 231: Drag PVariable .....                | 127 |
| Figure 232: Alt+drag on OnClick .....           | 128 |
| Figure 233: Drag PEdit to expr .....            | 128 |
| Figure 234: Create New variable .....           | 128 |
| Figure 235: Select Text option.....             | 129 |
| Figure 236: Press Ctrl+i for IF statement ..... | 129 |
| Figure 237: Change LogicExpr property .....     | 130 |
| Figure 238: Create new variable .....           | 130 |
| Figure 239: Change expr property .....          | 130 |
| Figure 240: Change Expr property .....          | 131 |
| Figure 241: Drag and drop above IF .....        | 131 |
| Figure 242: ShowMessage.....                    | 131 |
| Figure 243: Change Value property.....          | 132 |
| Figure 244: Project Screenshot .....            | 132 |
| Figure 245: Save your project.....              | 133 |
| Figure 246: Output.....                         | 133 |
| Figure 247: Create new project.....             | 134 |
| Figure 248: Place coponents .....               | 135 |
| Figure 249: Change Caption property.....        | 135 |
| Figure 250: Project Manager .....               | 136 |
| Figure 251: Create OnClick event .....          | 136 |
| Figure 252: Drag PVariable .....                | 136 |
| Figure 253: Create a new variable .....         | 137 |
| Figure 254: Drag PEdit to Expr property.....    | 137 |
| Figure 255: Change Expr Property .....          | 137 |
| Figure 256: Create variable.....                | 138 |
| Figure 257: Change Expr property .....          | 138 |
| Figure 258: Select Text property .....          | 139 |



|   |     |
|---|-----|
| Figure 259: Press Ctrl+I to create IF statement ..... | 139 |
| Figure 260: Change LogicExpr property .....           | 139 |
| Figure 261: ShowMessage .....                         | 140 |
| Figure 262: Change Value property .....               | 140 |
| Figure 263: Use ShowMessage .....                     | 140 |
| Figure 264: Change value property .....               | 141 |
| Figure 265: Save Project .....                        | 141 |
| Figure 266: Run Project .....                         | 142 |
| Figure 267: Create new project .....                  | 143 |
| Figure 268: Place Components .....                    | 144 |
| Figure 269: Change Caption .....                      | 144 |
| Figure 270: Create OnClick event .....                | 145 |
| Figure 271: Drag PVariable to OnClick .....           | 145 |
| Figure 272: Drag variable .....                       | 146 |
| Figure 273: Drag PEdit to expr property .....         | 146 |
| Figure 274: Drag PEdit and select Text .....          | 147 |
| Figure 275: Press Ctrl+I for IF statement .....       | 147 |
| Figure 276: Set LogicExpr .....                       | 148 |
| Figure 277: ShowMessage .....                         | 148 |
| Figure 278: Change value property .....               | 148 |
| Figure 279: Create ElseIf Object .....                | 149 |
| Figure 280: ShowMessage .....                         | 149 |
| Figure 281: Change value property .....               | 149 |
| Figure 282: Project screenshot .....                  | 150 |
| Figure 283: Save Project .....                        | 150 |
| Figure 284: Output .....                              | 151 |
| Figure 285: Create New Project .....                  | 152 |
| Figure 286: Place components .....                    | 153 |
| Figure 287: Double click for OnClick .....            | 153 |

|   |     |
|---|-----|
| Figure 288: Drag PVariable .....                            | 154 |
| Figure 289: Create Variable .....                           | 154 |
| Figure 290: select Text in source .....                     | 154 |
| Figure 291: Drag Plabel1 to OnCreate .....                  | 155 |
| Figure 292: Drag Variable to Expr property .....            | 155 |
| Figure 293: Save Project .....                              | 156 |
| Figure 294: Project Output .....                            | 156 |
| Figure 295: Create New Project .....                        | 157 |
| Figure 296: Place components .....                          | 158 |
| Figure 297: Drag PMethod .....                              | 158 |
| Figure 298: Change Parameters property .....                | 159 |
| Figure 299: ShowMessage .....                               | 159 |
| Figure 300: ShowMessage .....                               | 160 |
| Figure 301: Change Expr property .....                      | 160 |
| Figure 302: Double Click for OnClick event .....            | 160 |
| Figure 303: Drag PMethod .....                              | 161 |
| Figure 304: Drag PEdit objects to x,y property .....        | 161 |
| Figure 305: Save Project .....                              | 162 |
| Figure 306: Output .....                                    | 162 |
| Figure 307: Drag a Pcontrol to the Project Manager .....    | 164 |
| Figure 308: Drag PProperty to DbNavigator .....             | 164 |
| Figure 309: Set the Type property .....                     | 165 |
| Figure 310: Set the ClassName property .....                | 165 |
| Figure 311: Drag four PToolButton Objects .....             | 166 |
| Figure 312: Double click the buttons to create events ..... | 167 |
| Figure 313: Select a message .....                          | 167 |
| Figure 314: Select messages for all buttons .....           | 168 |
| Figure 315: The property panel .....                        | 169 |
| Figure 316: Components Panel .....                          | 169 |

|   |     |
|---|-----|
| Figure 317: creating a new component package.....       | 170 |
| Figure 318: Create New Project.....                     | 171 |
| Figure 319: Place components.....                       | 172 |
| Figure 320: View Binds option.....                      | 172 |
| Figure 321: choosing components class .....             | 173 |
| Figure 322: Source and Target sections.....             | 173 |
| Figure 323: SourceProperty .....                        | 174 |
| Figure 324: Selectiong a TargetObject .....             | 175 |
| Figure 325: Target options.....                         | 175 |
| Figure 326: Target property .....                       | 176 |
| Figure 327: PEdit text binded with PButton object ..... | 176 |
| Figure 328: Create new project.....                     | 177 |
| Figure 329: Drag PButton to Form .....                  | 178 |
| Figure 330: Create OnClick event .....                  | 178 |
| Figure 331: Drag PDirectory .....                       | 178 |
| Figure 332: Change PathName .....                       | 179 |
| Figure 333: Change Extension property .....             | 179 |
| Figure 334: ShowMessage.....                            | 180 |
| Figure 335: Code Complete window .....                  | 180 |
| Figure 336: Click the arrow .....                       | 181 |
| Figure 337: Select Count Property.....                  | 181 |
| Figure 338: Save Project.....                           | 182 |
| Figure 339: Output.....                                 | 182 |
| Figure 340: Create New Project.....                     | 183 |
| Figure 341: Create OnClick event .....                  | 184 |
| Figure 342: Drag PEdit .....                            | 184 |
| Figure 343: Drag PVariable .....                        | 184 |
| Figure 344: Create Variable.....                        | 185 |
| Figure 345: Drag PEdit1 to expr.....                    | 185 |

|   |     |
|---|-----|
| Figure 346: Select Text.....                              | 186 |
| Figure 347: Drag PFile to OnClick.....                    | 186 |
| Figure 348: Drag variable to Filename .....               | 187 |
| Figure 349: Change Value property.....                    | 187 |
| Figure 350: Save Project.....                             | 188 |
| Figure 351: Output.....                                   | 188 |
| Figure 352: Create New Project.....                       | 189 |
| Figure 353: Drag PButton to DefaultForm .....             | 190 |
| Figure 354: Create OnClick event .....                    | 190 |
| Figure 355: Drag PDirectory .....                         | 190 |
| Figure 356: Change PathName .....                         | 191 |
| Figure 357: Change Extension Property .....               | 191 |
| Figure 358: Hit Ctrl+Space for Code Complete window ..... | 192 |
| Figure 359: Select Files .....                            | 192 |
| Figure 360: Select DeleteFiles.....                       | 193 |
| Figure 361: Project screenshot.....                       | 193 |
| Figure 362: Save Project.....                             | 194 |
| Figure 363: Output.....                                   | 194 |
| Figure 364: Drag and drop files and mark PResource.....   | 195 |