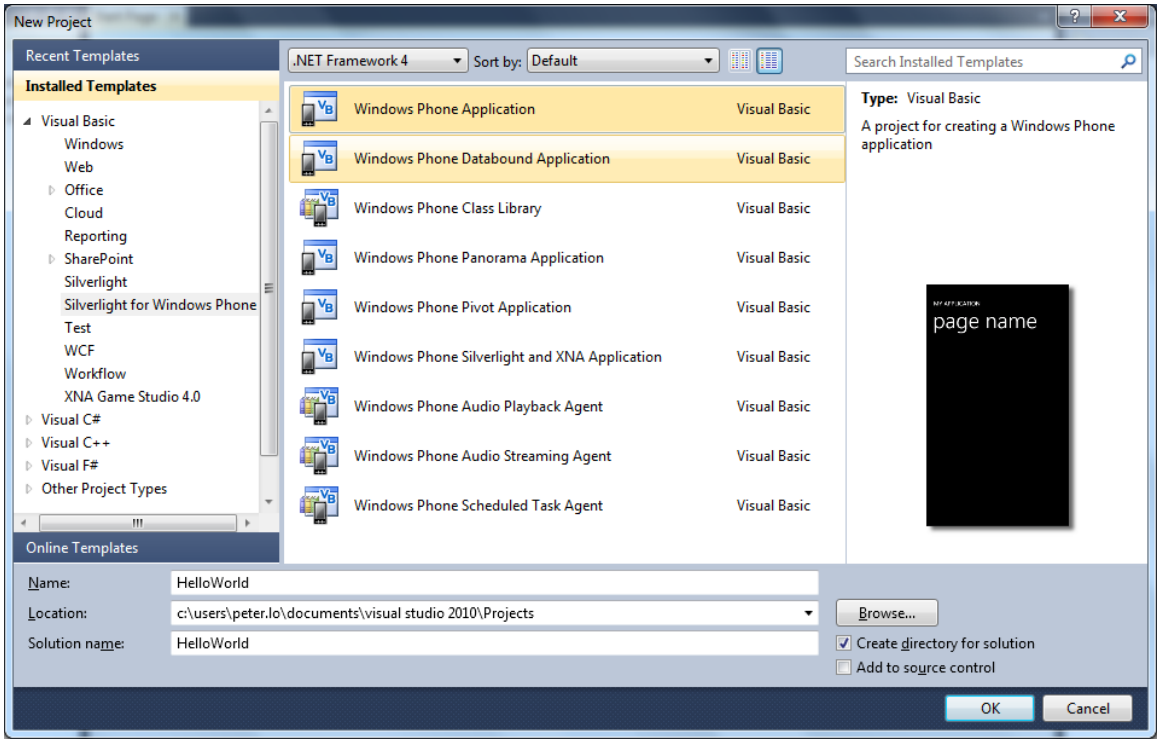
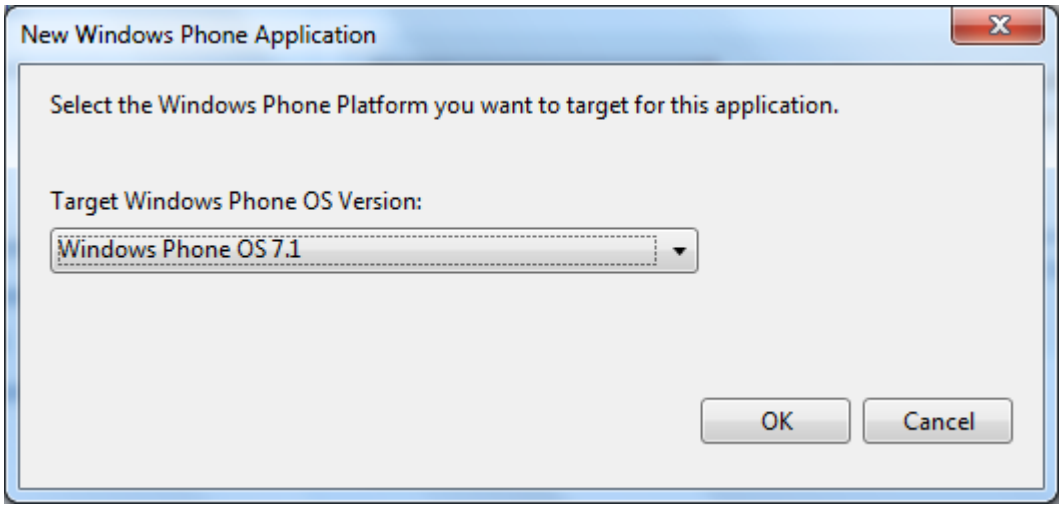
1. Windows Phone Development – Hello World

1. Open the Microsoft Visual Studio and start a new Visual Basic Project. Then create a new **Sliverlight for Windows Phone** Project **Hello World** by selecting **Windows Phone Application** Project on the **Installed Template** panel in the **New Project** dialog

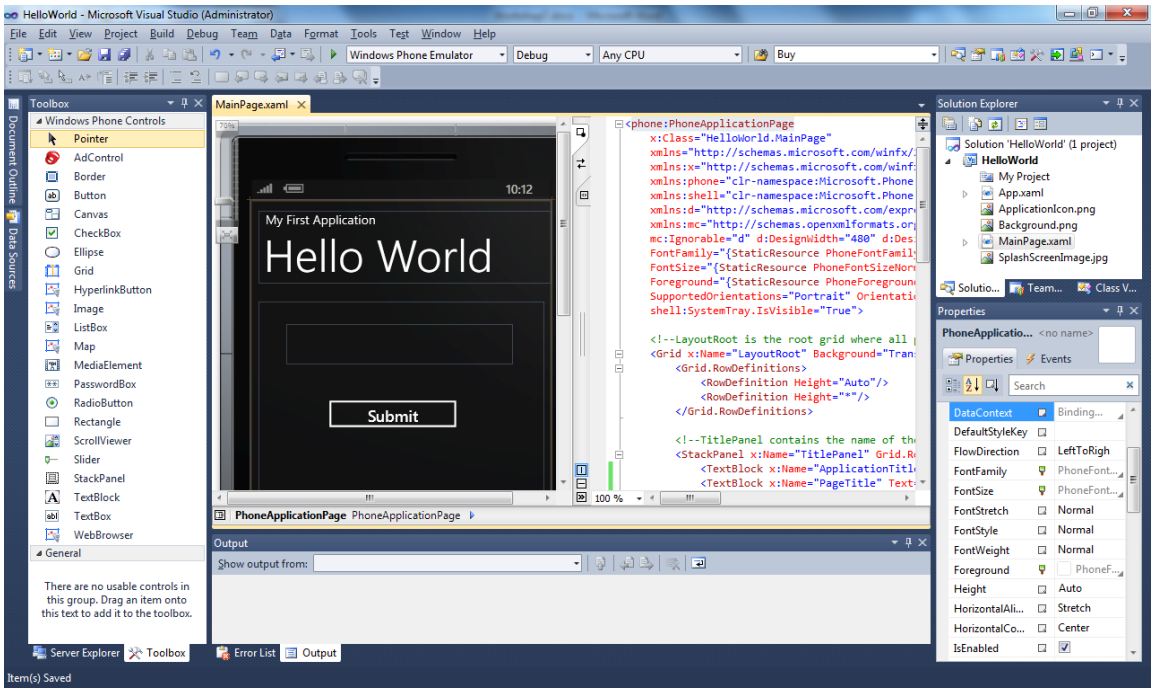


2. Select **Windows Phone OS 7.1** as the target Windows Phone OS Version.



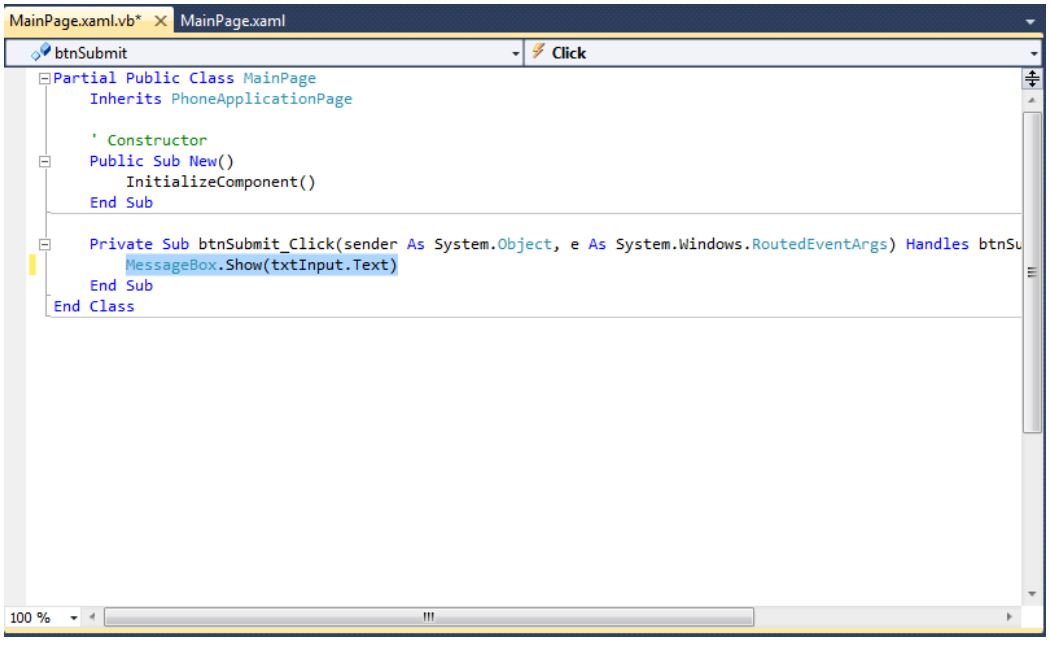
3. From the Toolbox, drag an **Image** control, a **Textbox** control and a **Button** control onto the form and customize the properties.

Object	Name	Property	Property Value
TextBlock	ApplicationTitle	Text	My First Application
TextBlock	PageTitle	Text	Hello World
TextBlock	txtInput	Text	*(Blank)*
Button	btnSubmit	Content	Submit

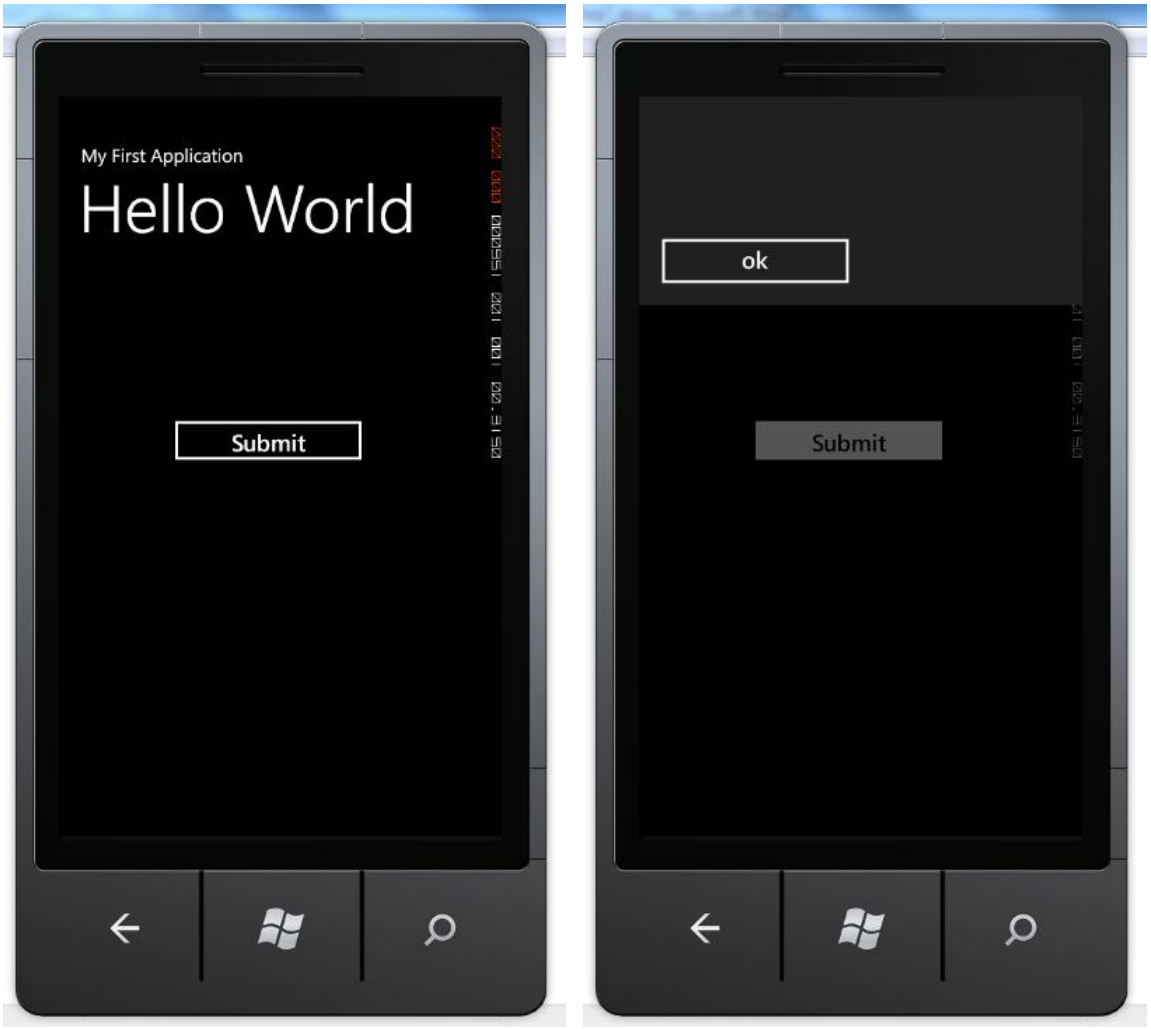


4. In the **Click** event procedure of the **btnSubmit** control, add the following code.

```
' Show the input from textblock
MessageBox.Show(txtInput.Text)
```



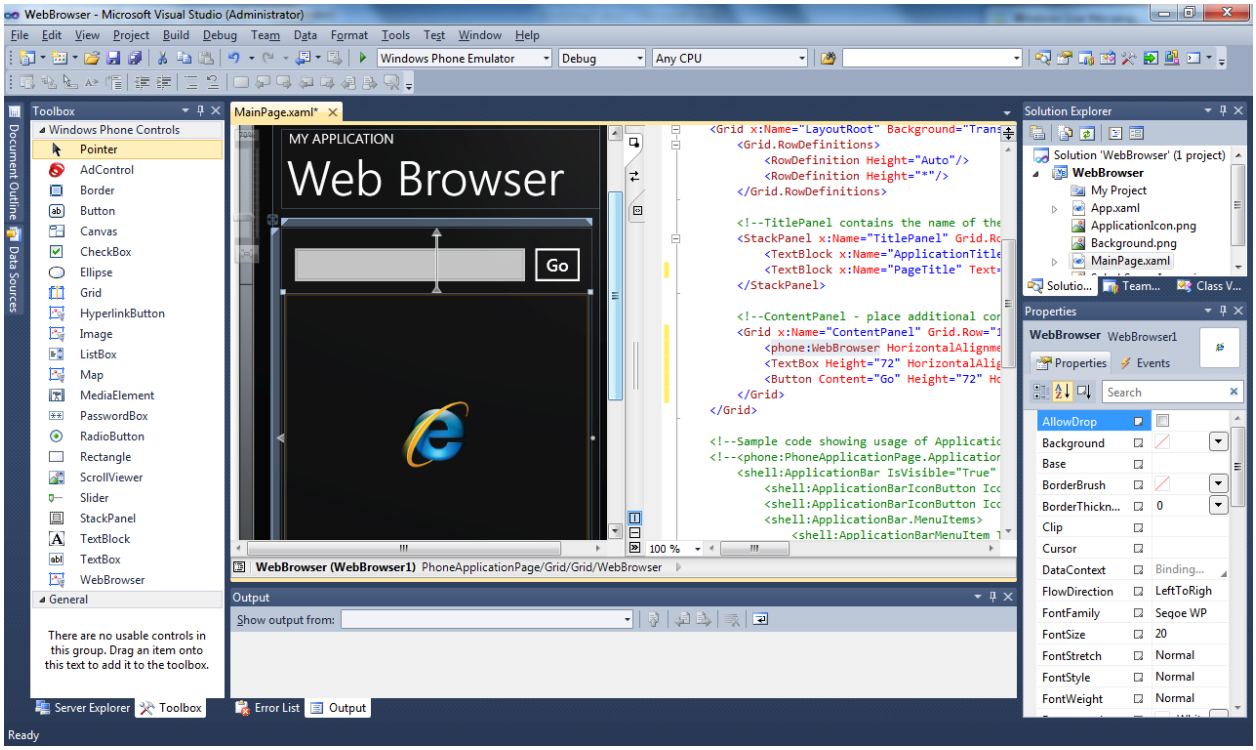
5. Compile and execute your application in the Windows Phone Emulator.



2. Windows Phone Development – Web Browser

1. Open the Microsoft Visual Studio and start a **Windows Phone Application** Project **WebBrowser**. From the Toolbox, drag a **TextBox** control a WebBrowser control and a **Button** control onto the form and customize the properties.

Object	Name	Property	Property Value
TextBlock	PageTitle	Text	Web Browser
WebBrowser	WebBrowser1		
TextBox	txtInput	Text	*(Blank)*
Button	btnGo	Content	Go



2. In the **Click** event procedure of the **btnGo** control, add the following code.

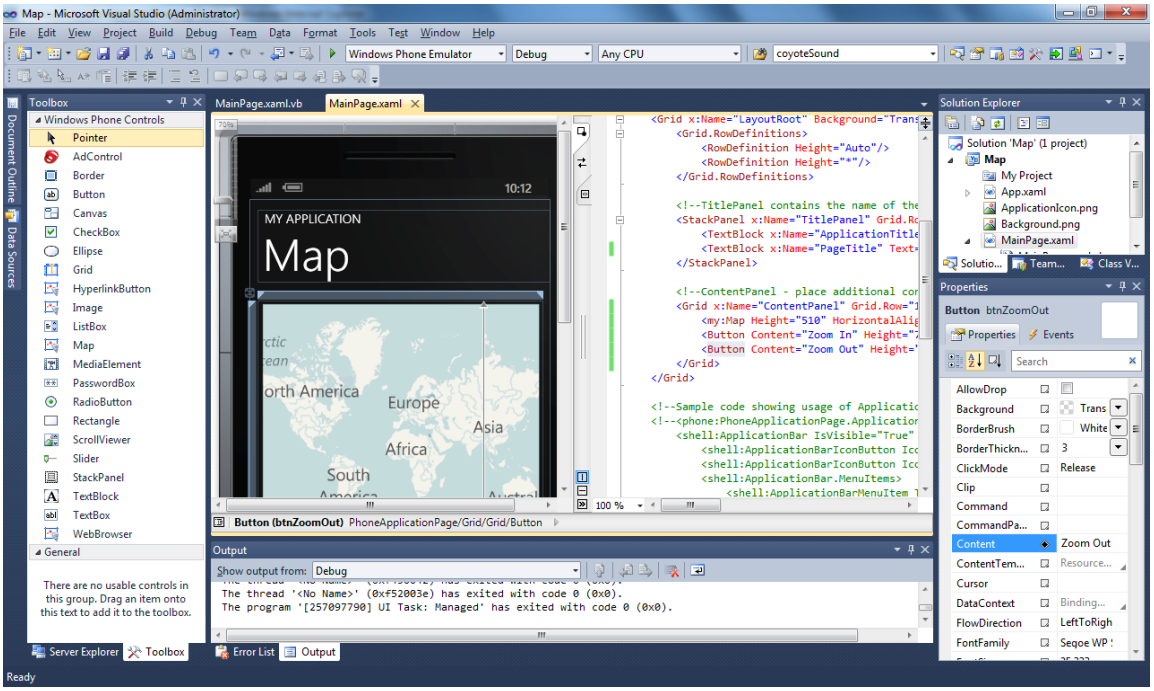
```
' Browse the website
WebBrowser1.Navigate(New Uri(txtInput.Text, UriKind.Absolute))
```

3. Compile and execute your application in the Windows Phone Emulator

3. Windows Phone Development – Map

1. Open the Microsoft Visual Studio and start a **Windows Phone Application** Project **Map**. From the Toolbox, drag two **Button** controls onto the form and customize the properties.

Object	Name	Property	Property Value
TextBlock	PageTitle	Text	Web Browser
Map	Map1		
Button	btnZoomIn	Text	Zoom In
Button	btnZoomOut	Text	Zoom Out



2. In the **Click** event procedure of the **btnZoomIn** control, add the following code.

```
Dim zoom As Double

zoom = Map1.ZoomLevel + 1
Map1.ZoomLevel = zoom
```

3. In the **Click** event procedure of the **btnZoomOut** control, add the following code.

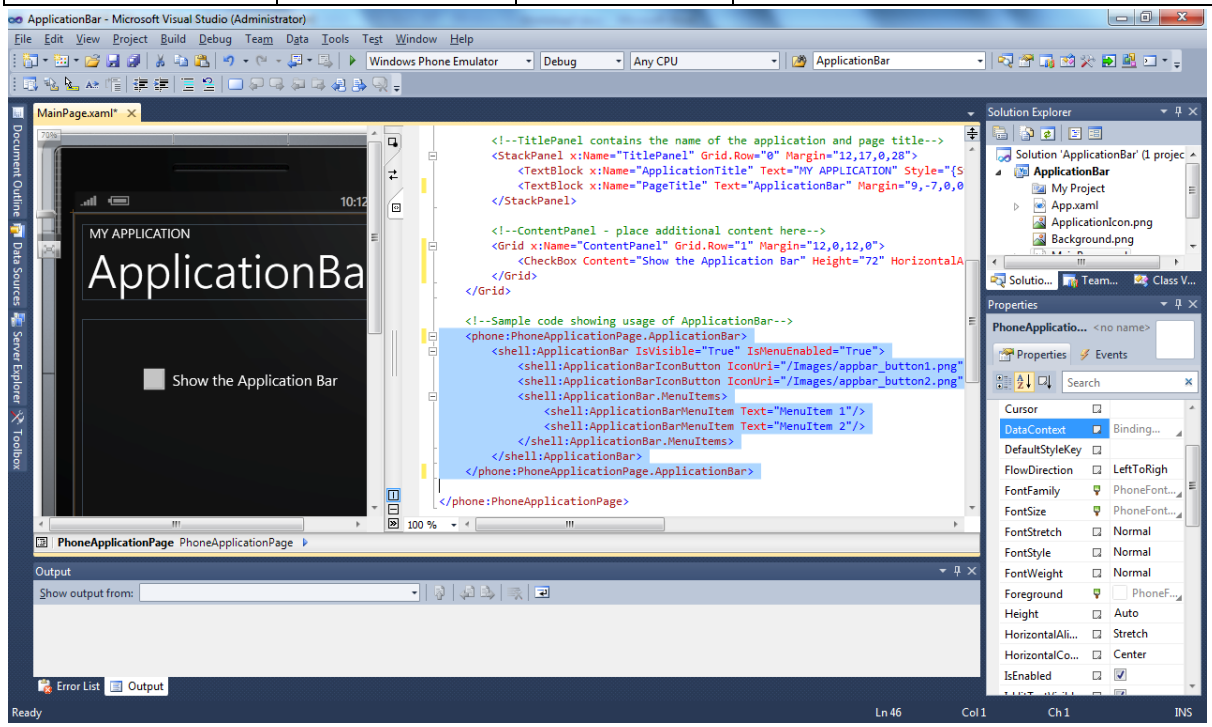
```
Dim zoom As Double

zoom = Map1.ZoomLevel - 1
Map1.ZoomLevel = zoom
```

4. Compile and execute your application in the Windows Phone Emulator

4. Windows Phone Development – Application Bar

1. Open the Microsoft Visual Studio and start a **Windows Phone Application** Project **ApplicationBar**. From the Toolbox, drag a **Checkbox** controls onto the form and customize the properties.

Object	Name	Property	Property Value
TextBlock	PageTitle	Text	Application Bar
Checkbox	CheckBox1	Content	Show the Application Bar
		IsChecked	True

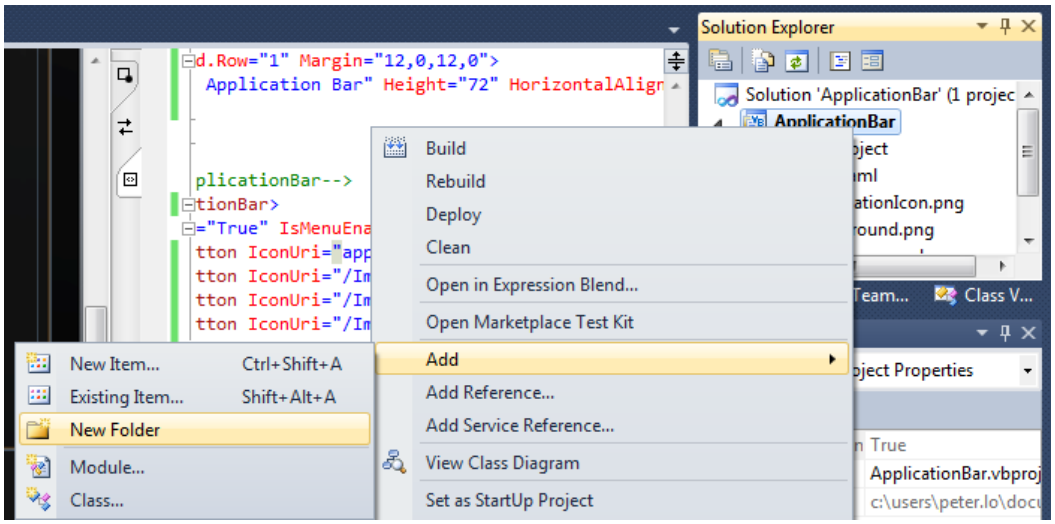


2. Add the Application bar by edit the XAML as follow:

```
<phone:PhoneApplicationPage.ApplicationBar>
    <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
        <shell:ApplicationBarIconButton IconUri="/Images/appbar.edit.rest.png"
                                        Text="Edit" Click="IconBarButton_Click"
        <shell:ApplicationBarIconButton IconUri="/Images/appbar.download.rest.png"
                                        Text="Download"/>
        <shell:ApplicationBarIconButton IconUri="/Images/appbar.favs.rest.png"
                                        Text="Favourite"/>
        <shell:ApplicationBarIconButton IconUri="/Images/appbar.folder.rest.png"
                                        Text="Folder"/>
        <shell:ApplicationBar.MenuItems>
```

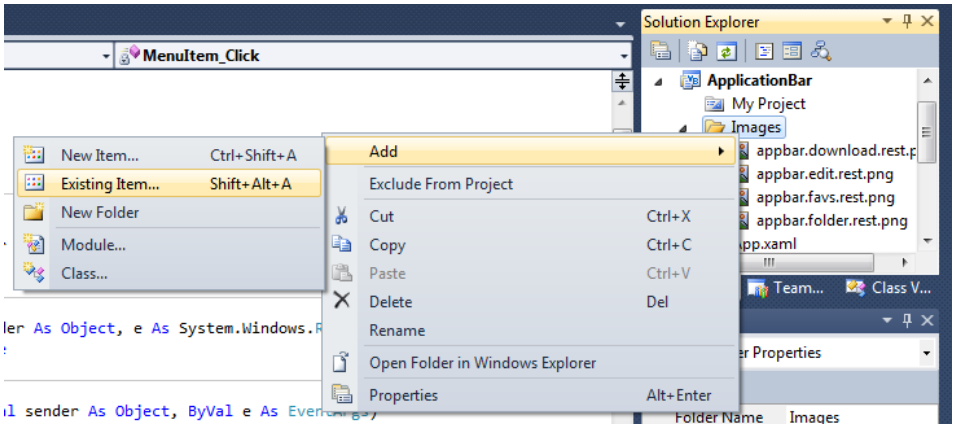
```
            <shell:ApplicationBarMenuItem Text="Item1" Click="MenuItem_Click"/>
            <shell:ApplicationBarMenuItem Text="Item2"/>
            <shell:ApplicationBarMenuItem Text="Item3"/>
        </shell:ApplicationBar.MenuItems>
    </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar>
```

3. In Solution Explorer, create a new folder called **images**.

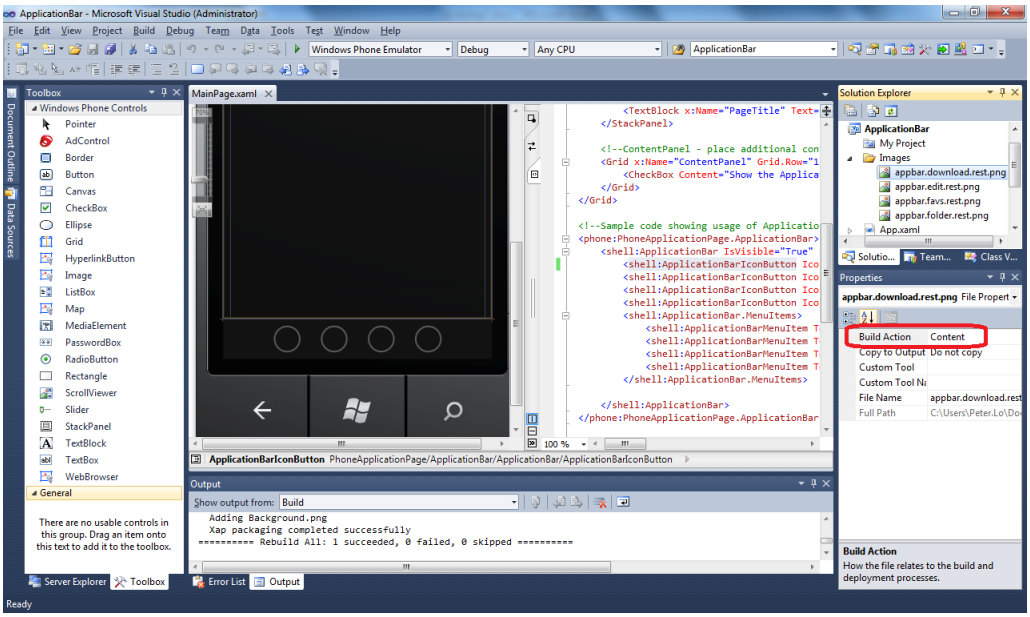


4. Icons used for ApplicationBar command buttons should be of 48 x 48 pixels. Some common icon are stored in the folder **C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.1\Icons**.

5. Right click the **Image** folder, and select **Add ➔ Existing Items** to add the selected images. You can also drag and drop the selected image to the **Image** folder in the **Solution Manager**.



6. The application bar will not display the right images there if only add the images to the project. To set them properly, right click on the images and go to the properties panel. Now from the **Build Action** dropdown chose **Content** instead of **Resource**.



7. In the **Checked** event procedure of the **CheckBox1** control, add the following code.

```
' Show the Application Bar
ApplicationBar.IsVisible = True
```

8. In the **Unchecked** event procedure of the **CheckBox1** control, add the following code.

```
' Hide the Application Bar
ApplicationBar.IsVisible = False
```

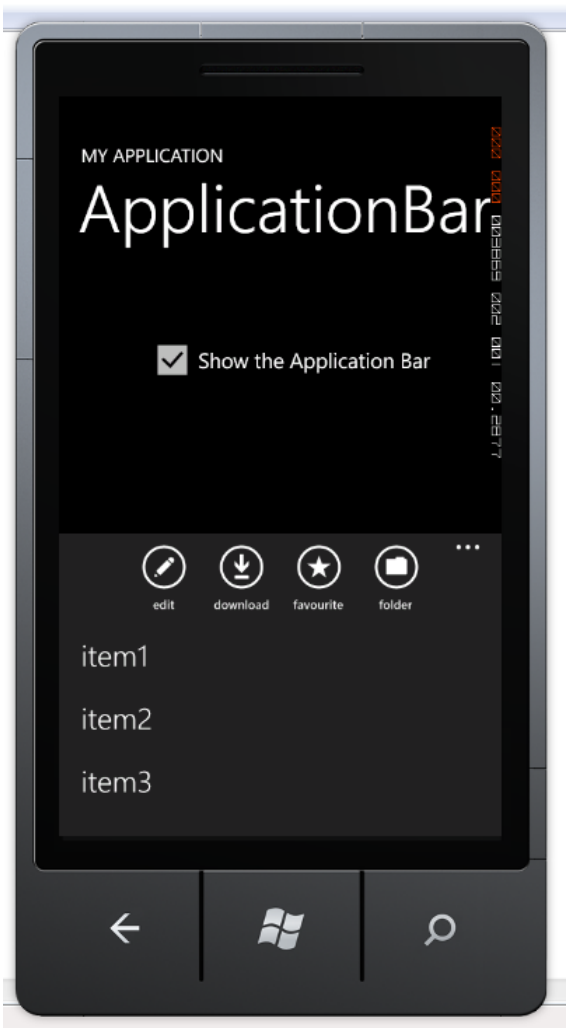
9. Create the click event handlers for the icon bar button by adding the following code.

```
' Create the click event handler for the Icon bar button
Private Sub IconBarButton_Click(ByVal sender As Object, ByVal e As EventArgs)
        MessageBox.Show("Icon button is clicked")
End Sub
```

10. Create the click event handlers for the menu item by adding the following code.

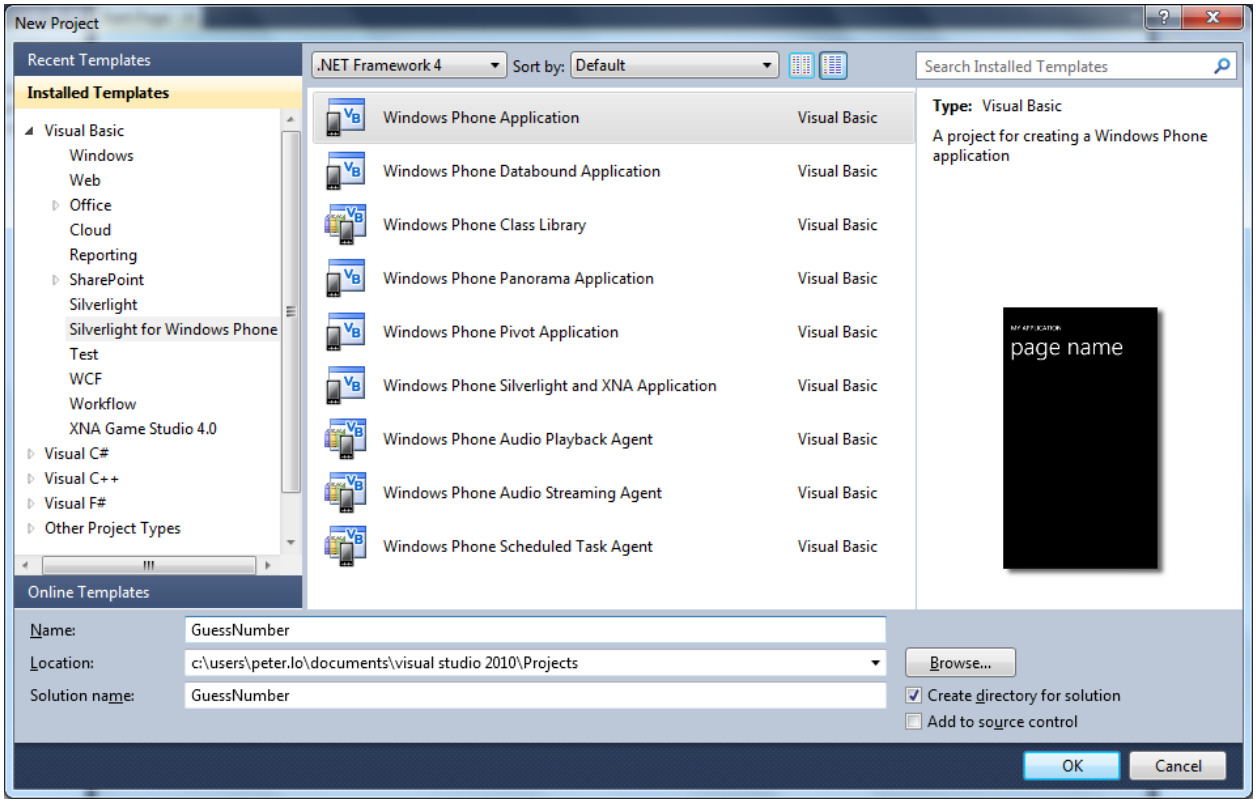
```
' Create the click event handler for the menu item
Private Sub MenuItem_Click(ByVal sender As Object, ByVal e As EventArgs)
        MessageBox.Show("Menu button is clicked")
End Sub
```

11. Compile and execute your application in the Windows Phone Emulator

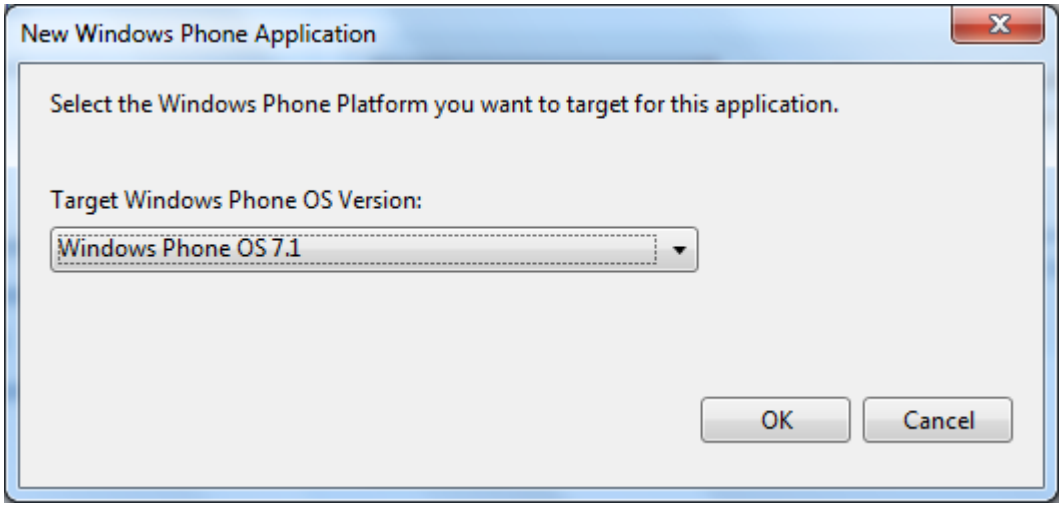


5. Windows Phone Development – Guess Number

1. Open the Microsoft Visual Studio and start a new Visual Basic Project. Then create a new **Sliverlight for Windows Phone** Project **GuessNumber** by selecting **Windows Phone Application** Project on the **Installed Template** panel in the **New Project** dialog

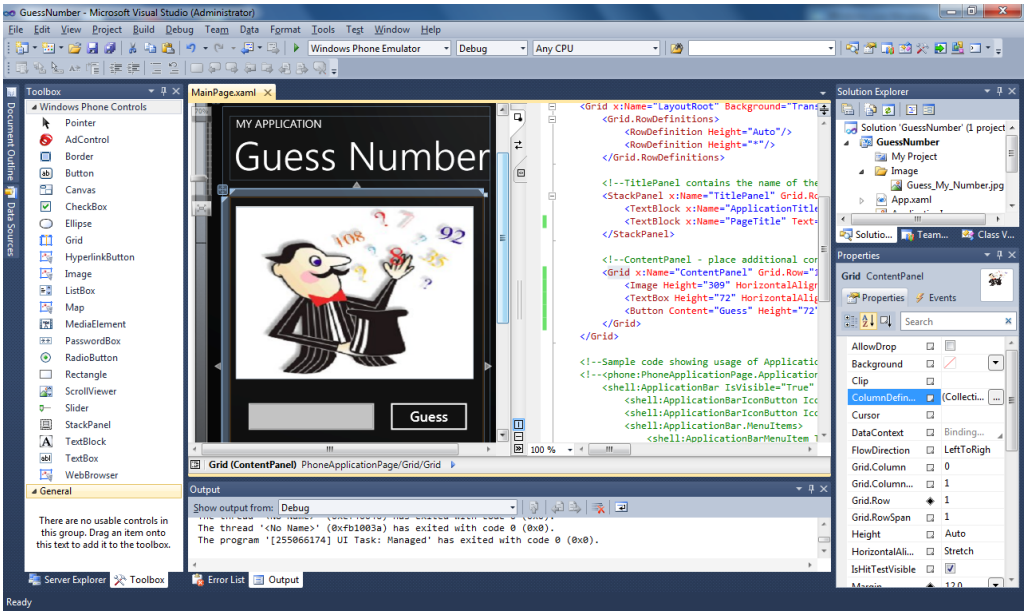


6. Select **Windows Phone OS 7.1** as the target Windows Phone OS Version.



7. From the Toolbox, drag an **Image** control, a **Textbox** control and a **Button** control onto the form and customize the properties.

Object	Name	Property	Property Value
TextBlock	PageTitle	Text	Guess Number
Image	Image1	Source	*(Guess_My_Number.jpg)*
TextBox	txtInput	Text	*(Blank)*
Button	btnGuess	Content	Guess



8. Declare the module level variables within the class.

```
Private RandomNumber As Integer
```

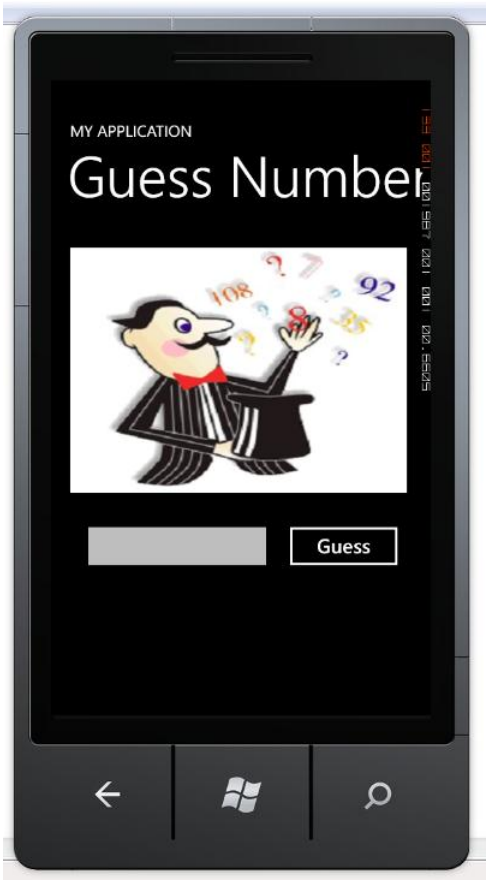
9. In the **Loaded** event procedure of the **MainPage** control, add the following code.

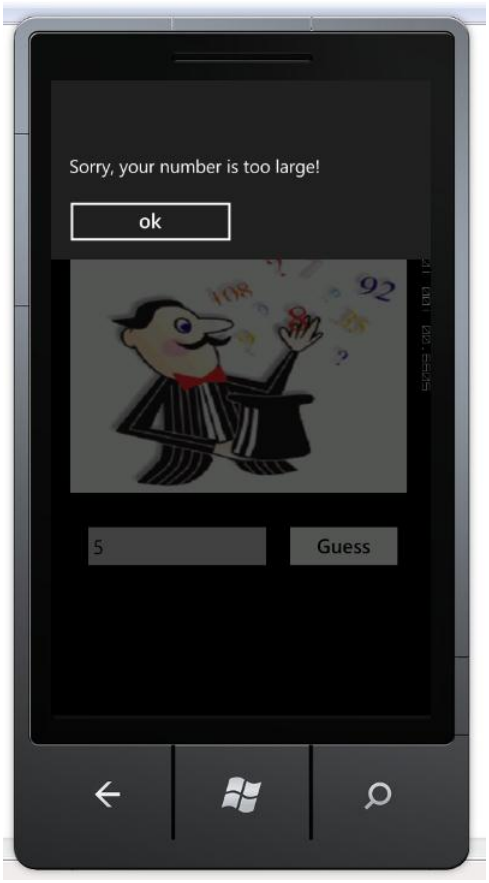
```
' Declare a random generator
Dim RandomGenerator As New Random

' Generate a random number from 1 to 9
RandomNumber = RandomGenerator.Next(1, 9)
```

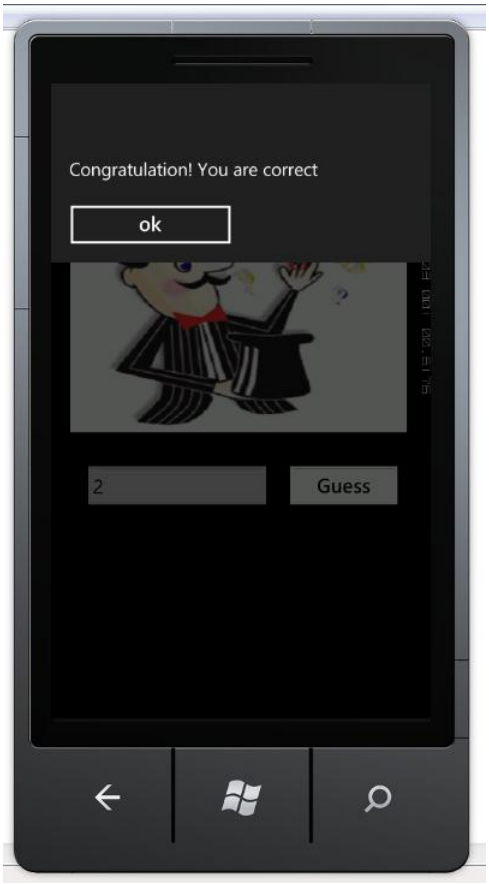
10. In the **Click** event procedure of the **btnGuess** control, add the following code.

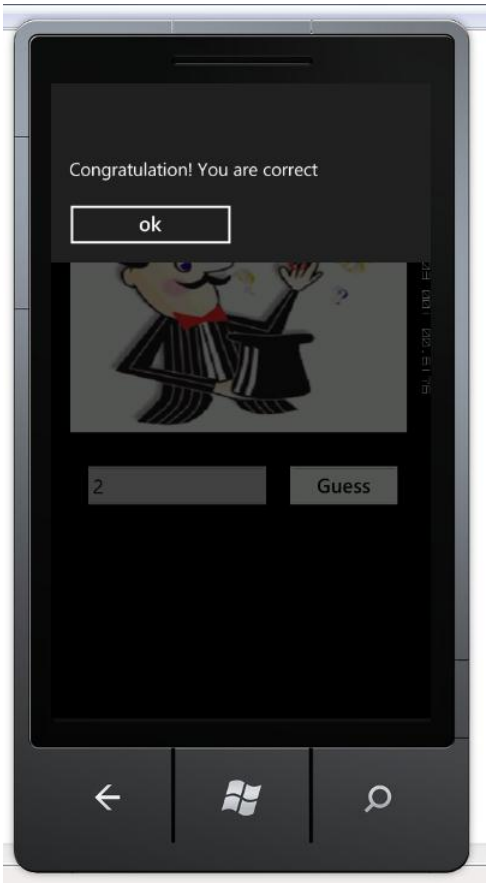
```
' Validate the result
If txtInput.Text = RandomNumber Then
    MessageBox.Show("Congratulation! You are correct")
ElseIf txtInput.Text > RandomNumber Then
    MessageBox.Show("Sorry, your number is too large!")
ElseIf txtInput.Text < RandomNumber Then
    MessageBox.Show("Sorry, your number is too small!")
End If
```

11. Compile and execute your application in the Windows Phone Emulator.



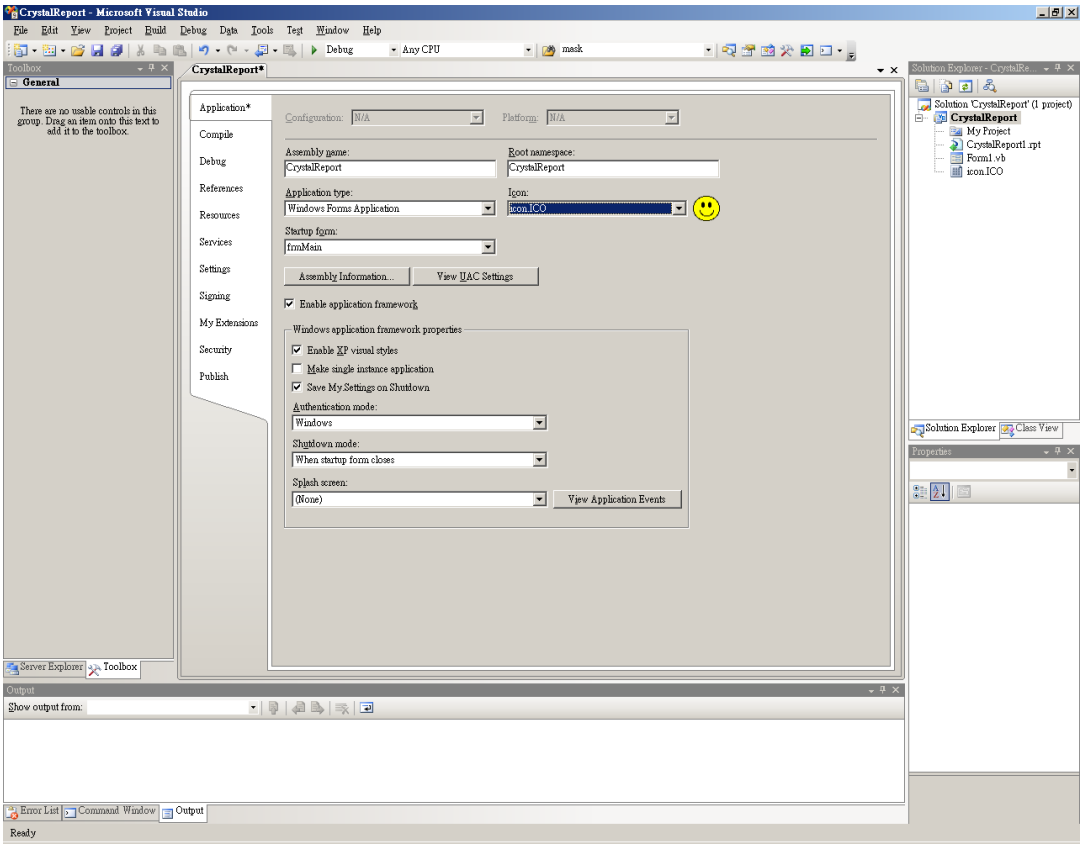




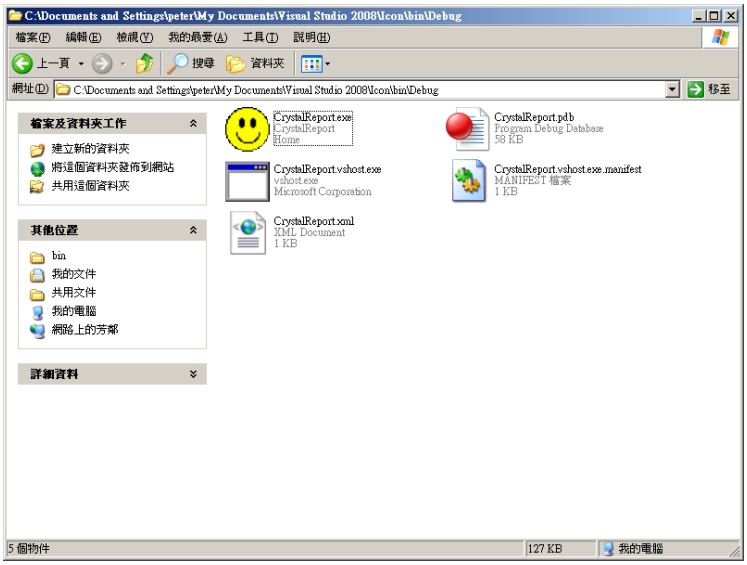


6. Adding Icon to Windows Application

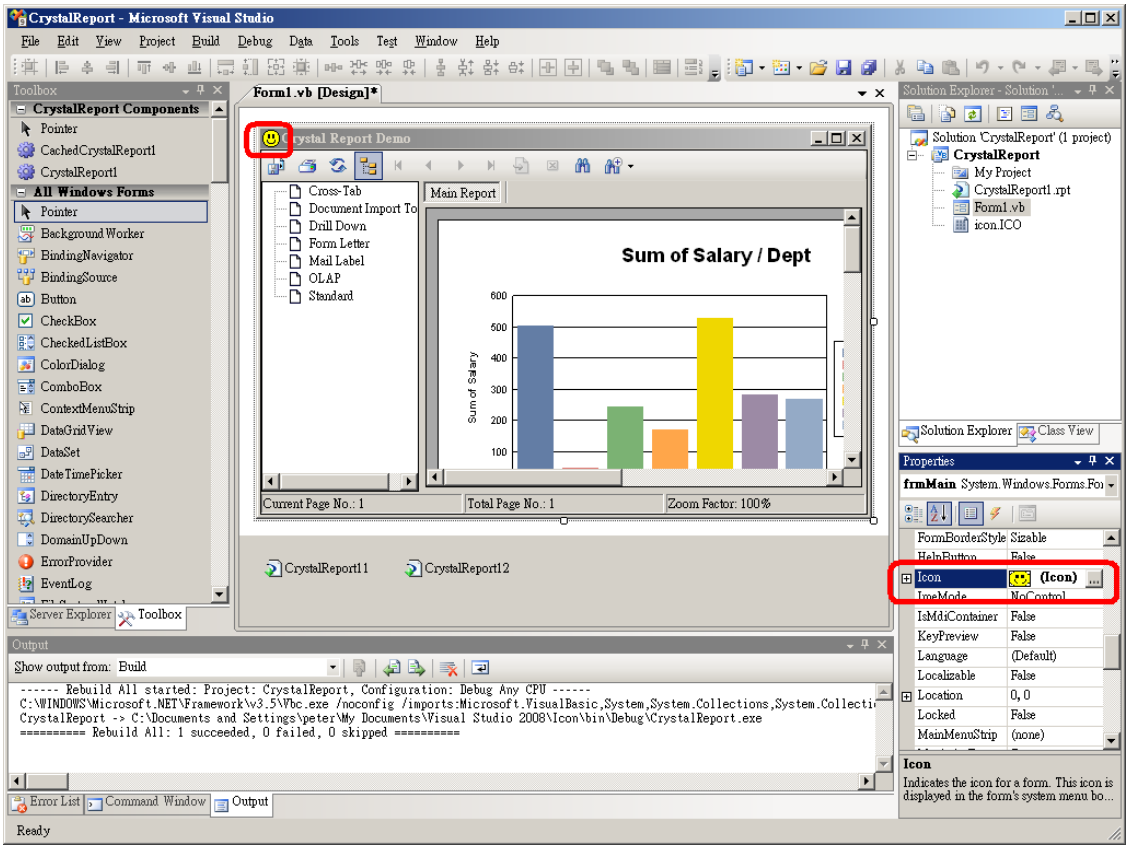
1. Start the Microsoft Visual Studio, in the **Solution Explorer** Windows, select the **Properties** for the **Project**, and then add the icon file to the **Common Properties**.



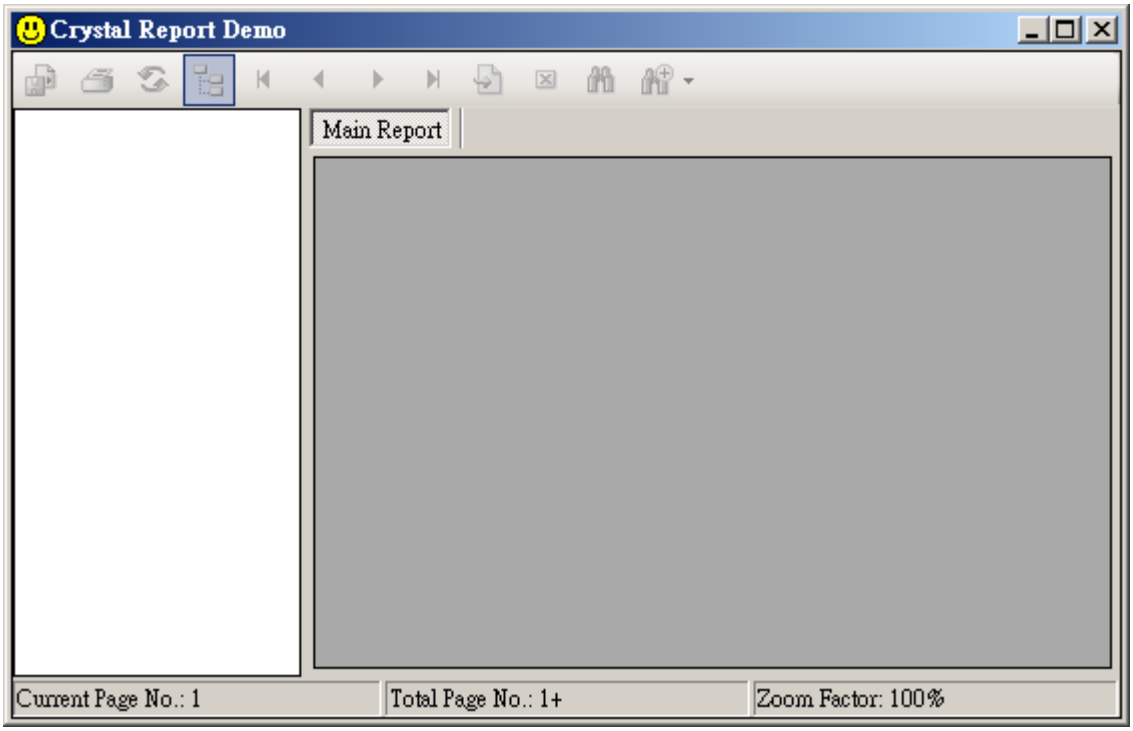
2. The application icon will change after the rebuild of the project.



3. In the **Solution Explorer** Windows, select the **Properties** for the **Project**, and then add the icon file to the **Common Properties**.

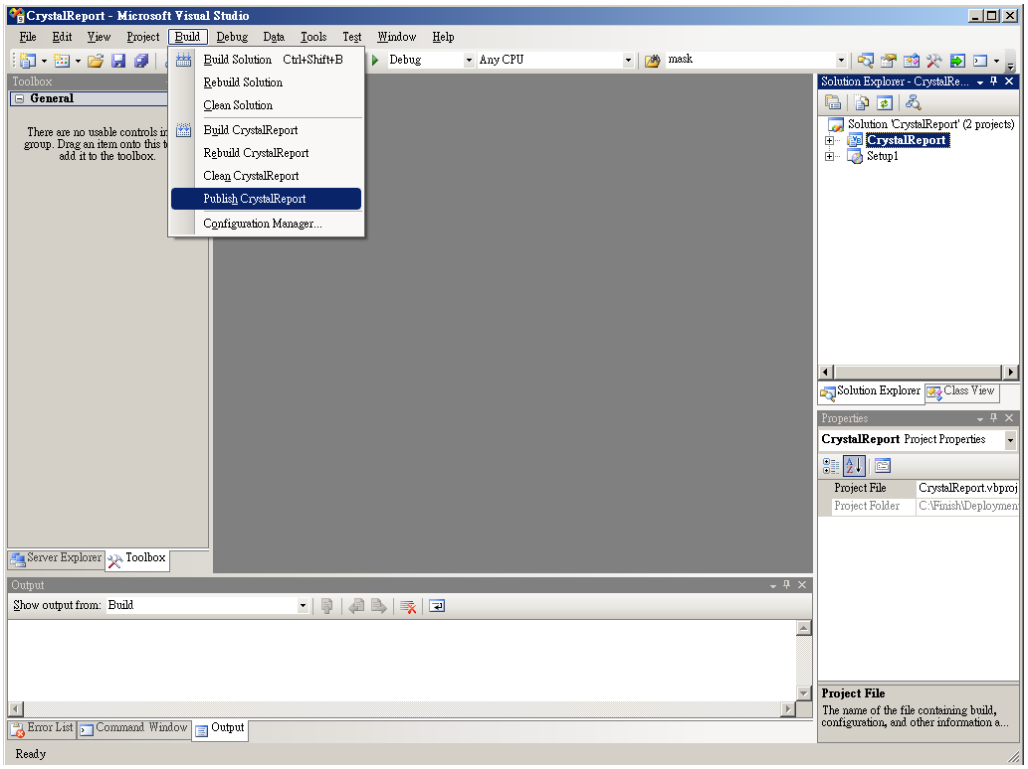


4. Save the project and build the solution, and then execute it. The icon for the form is changed.

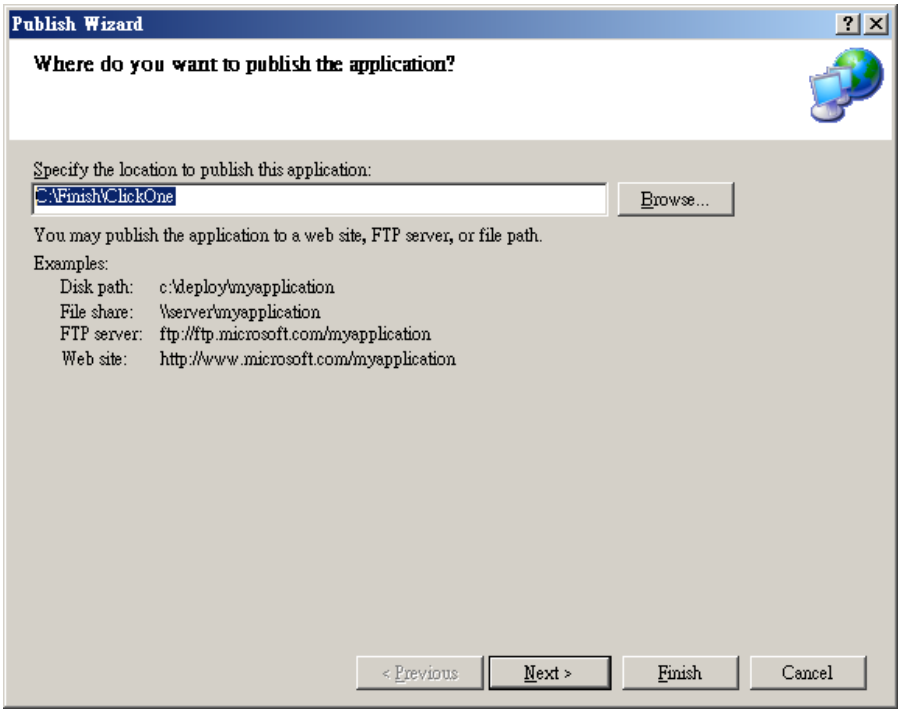


7. Deployment – ClickOne

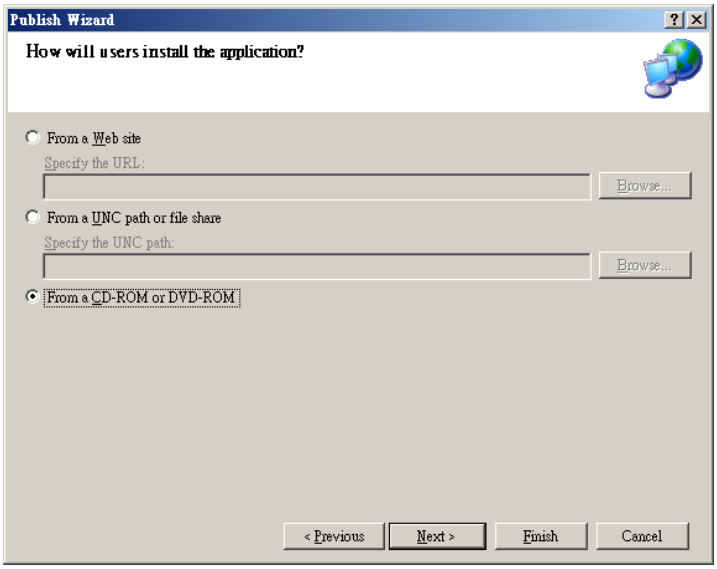
1. Start the Microsoft Visual Studio and open the previous Visual Basic Project **CrystalReport**. Select **Build ➔ Publish** for ClickOne deployment.



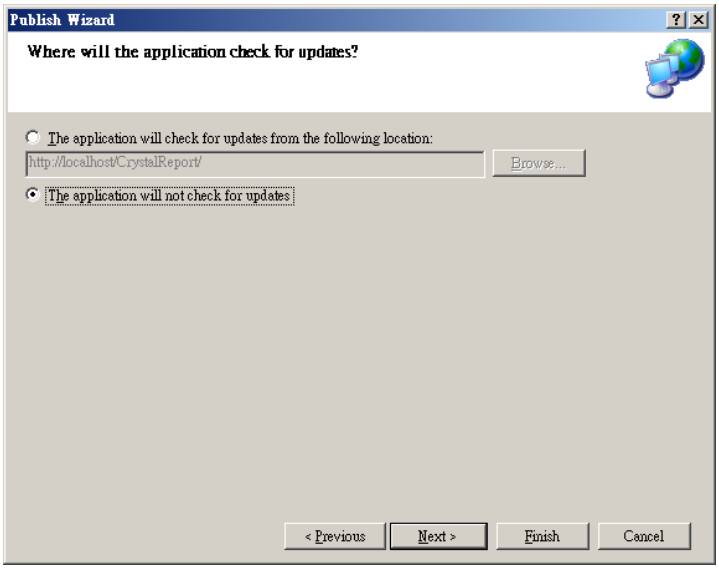
2. Input the location for the application to be published. If you define it in a URL, the deployed application can get the update from the website.

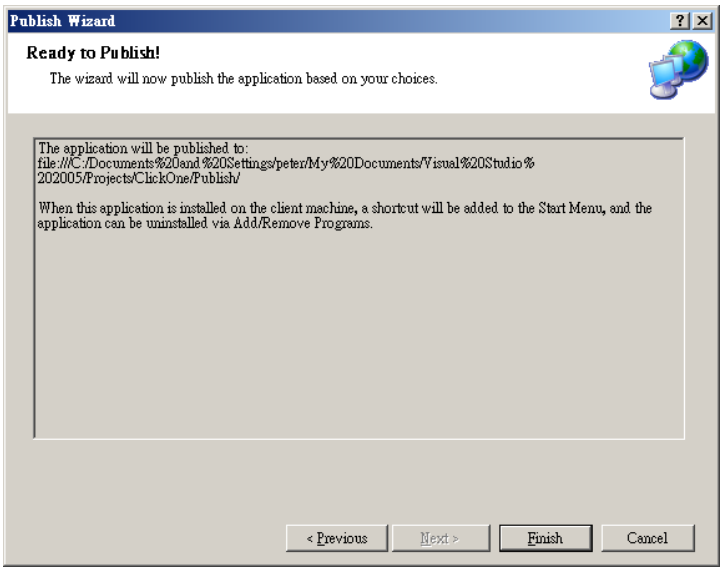


3. Define the method that user install the application, you need to specify **From a CD-ROM or DVD-ROM** if it is published in a file.



4. You can define the update option if a website is used for published.

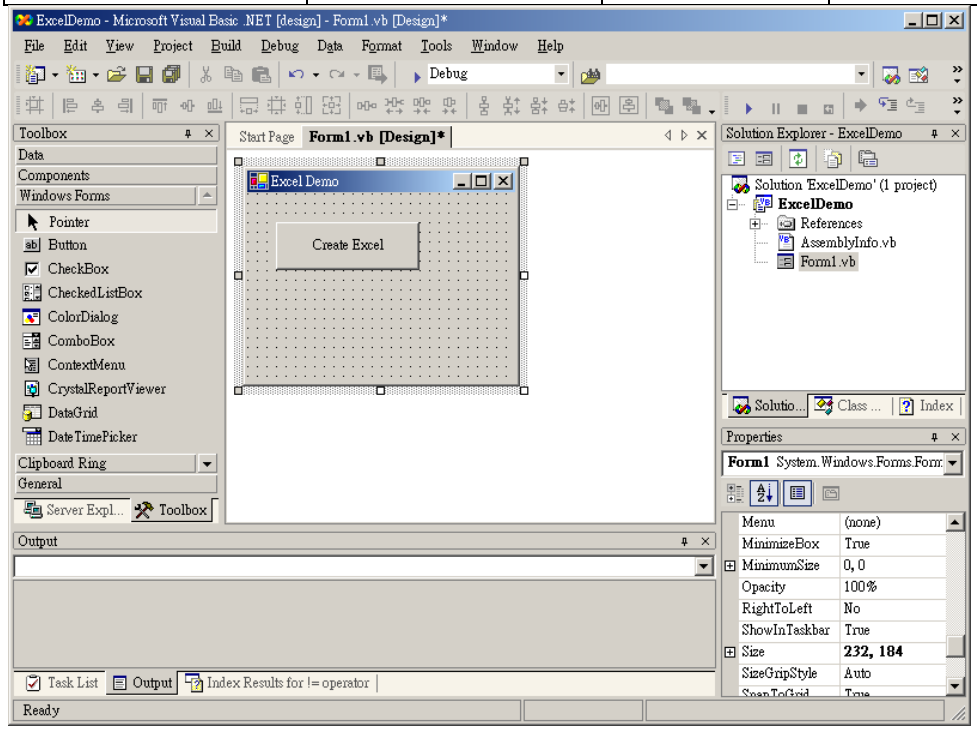




8. Export Data to Microsoft Excel

1. Open the Microsoft Visual Studio and start a new Visual Basic Project named as **ExportExcel**. From the Toolbox, drag a **Button** controls onto the form and customize the properties.

Object	Name	Property	Property Value
Form	frmMain	Text	Excel Export Demo
Button	btnExcel	Text	Create Excel

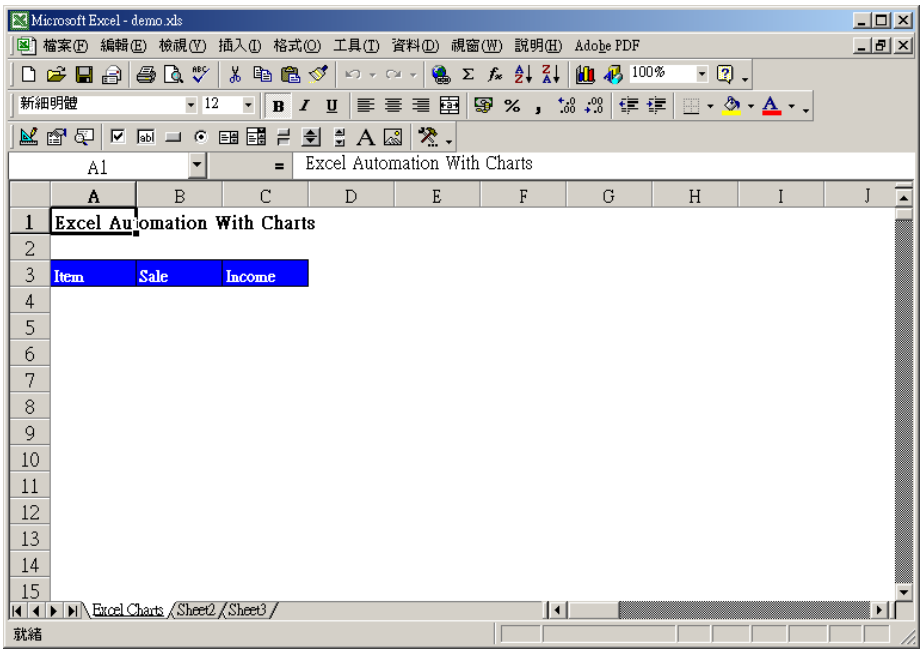


2. In the **Click** event procedure of the Create Excel button control (**btnExcel**), add the following code.

```
'File name and path
Dim Filename = AppDomain.CurrentDomain.BaseDirectory & "demo.xls"

' Create new excel application
Dim objExcel = CreateObject("Excel.Application")

' Add a new workbook
Dim objBook = objExcel.Workbooks.Add

' Set the application alerts not to be displayed for confirmation
objExcel.Application.DisplayAlerts = True

' This procedure populate the sheet
objExcel.Visible = True

' Add a new sheet
```

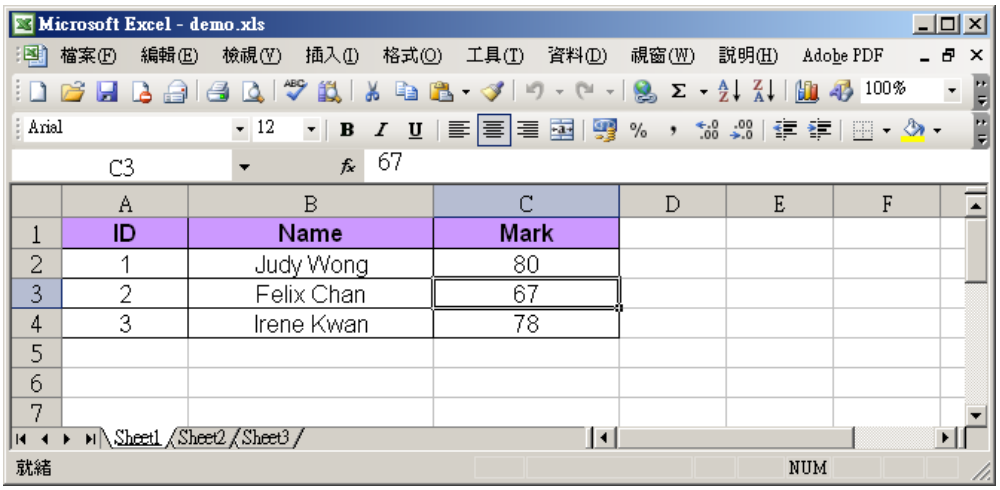
```
Dim objSheet = objExcel.Worksheets(1)

' Rename the sheet
objSheet.Name = "Excel Charts"

' Format Title
objSheet.Range("A1:AZ400").Interior.ColorIndex = 2
objSheet.Range("A1").Font.Size = 12
objSheet.Range("A1").Font.Bold = True
objSheet.Range("A1:I1").Merge()
objSheet.Range("A1").Value = "Excel Automation With Charts"
objSheet.Range("A1").EntireColumn.AutoFit()

' Format headings
objSheet.Range("A3:C3").Font.Color = RGB(255, 255, 255)
objSheet.Range("A3:C3").Interior.ColorIndex = 5
objSheet.Range("A3:C3").Font.Bold = True
objSheet.Range("A3:C3").Font.Size = 10

' Columns heading
objSheet.Range("A3").Value = "Item"
objSheet.Range("A3").BorderAround(8)
objSheet.Range("B3").Value = "Sale"
objSheet.Range("B3").BorderAround(8)
objSheet.Range("C3").Value = "Income"
objSheet.Range("C3").BorderAround(8)

' Save the Excel
objBook.SaveAs(Filename)
```

3. Save and build the project, you can press the **[Create Excel]** button to create and open a new Microsoft Excel file.



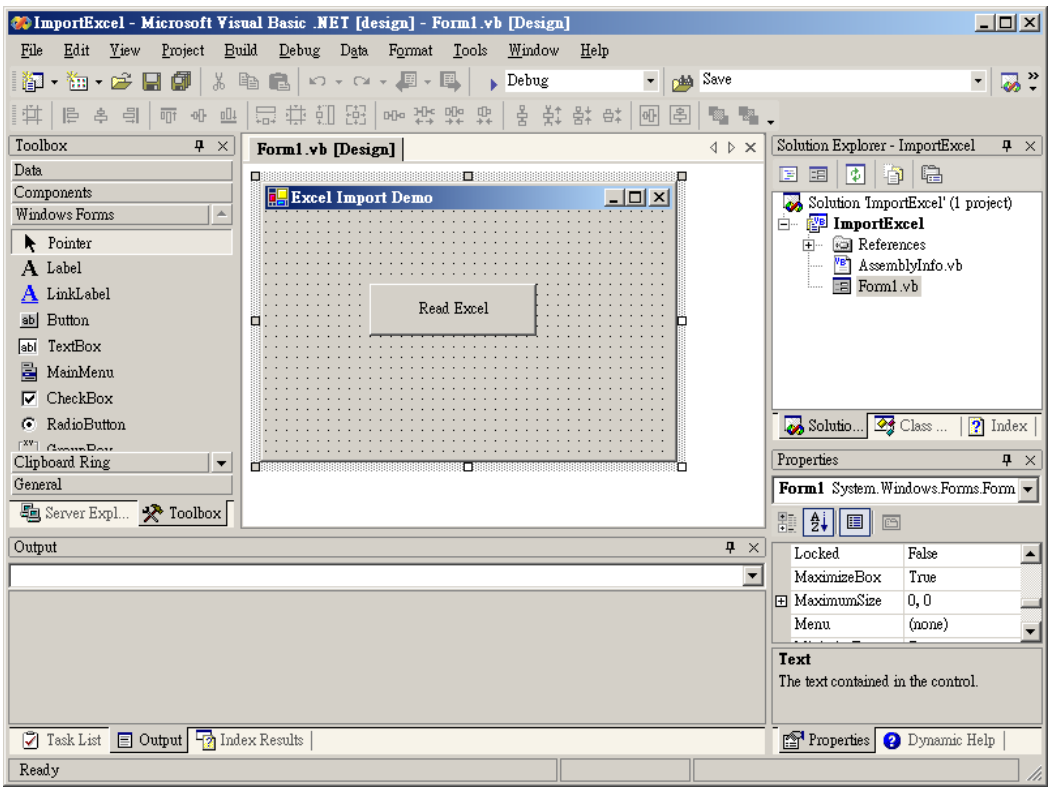
9. Read Data to Microsoft Excel

1. Create an Excel file with the following content and save to “C:\Temp\demo.xls”.



2. Open the Microsoft Visual Studio and start a new Visual Basic Project named as **ImportExcel**. From the Toolbox, drag a **Button** controls onto the form and customize the properties.

Object	Name	Property	Property Value
Form	frmMain	Text	Excel Import Demo
Button	btnExcel	Text	Read Excel



3. In the **Click** event procedure of the Read Excel button control (**btnExcel**), add the following code.

```
' Declare a string
Dim var As String

' Define the File name and path
Dim Filename = "C:\Temp\demo.xls"

' Open the Excel
Dim objExcel = GetObject(Filename)

' Get the value from Excel
var = objExcel.Worksheets(1).Cells(2, 2).Value

' Display the message
MsgBox(var)

' Close the Excel file
objExcel.close()
```

4. Save and build the project, you can press the **[Read Excel]** button to read the Excel content.