Windows Forms

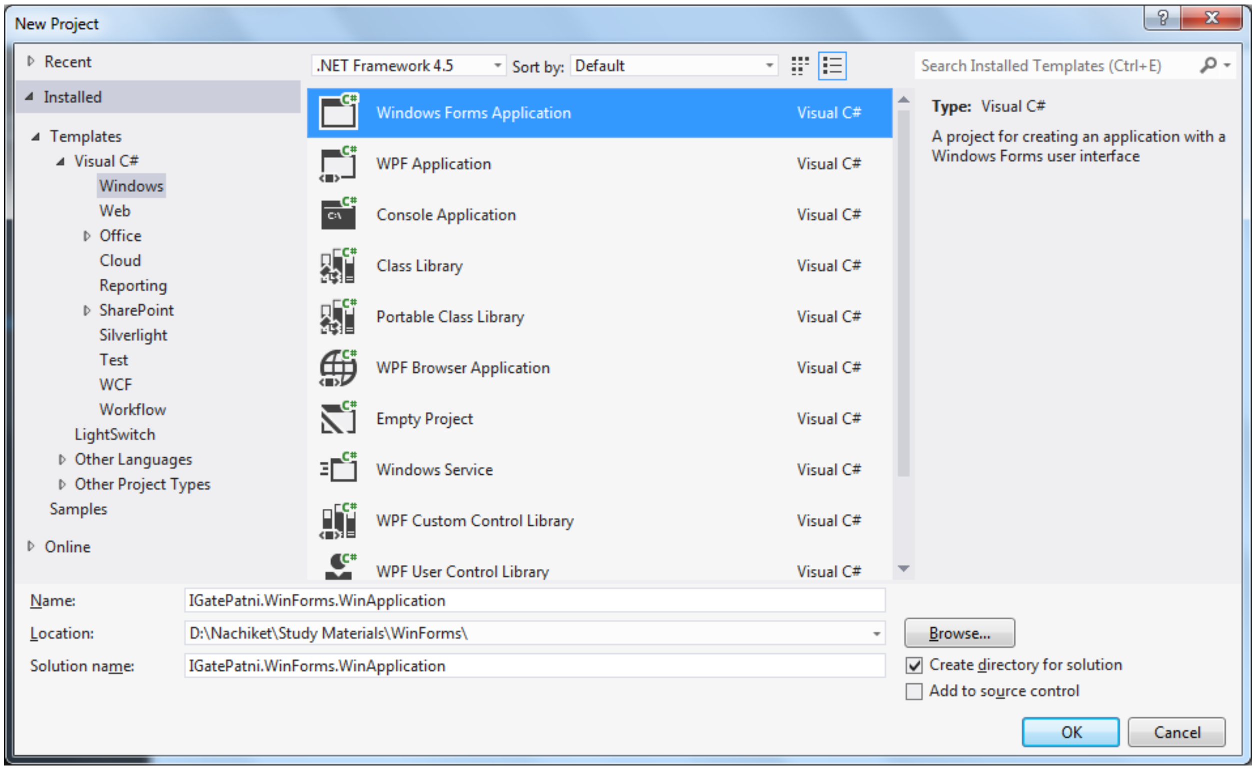
Lab Book

Table of Contents

Lab 1.Getting familiar with Windows Application development environment

Description	We will be creating a simple application which welcomes the user.
Objective	To Learn: • How to set properties • Generate event handlers and handle event • Setting frequently used common properties.
Time	90 Mins

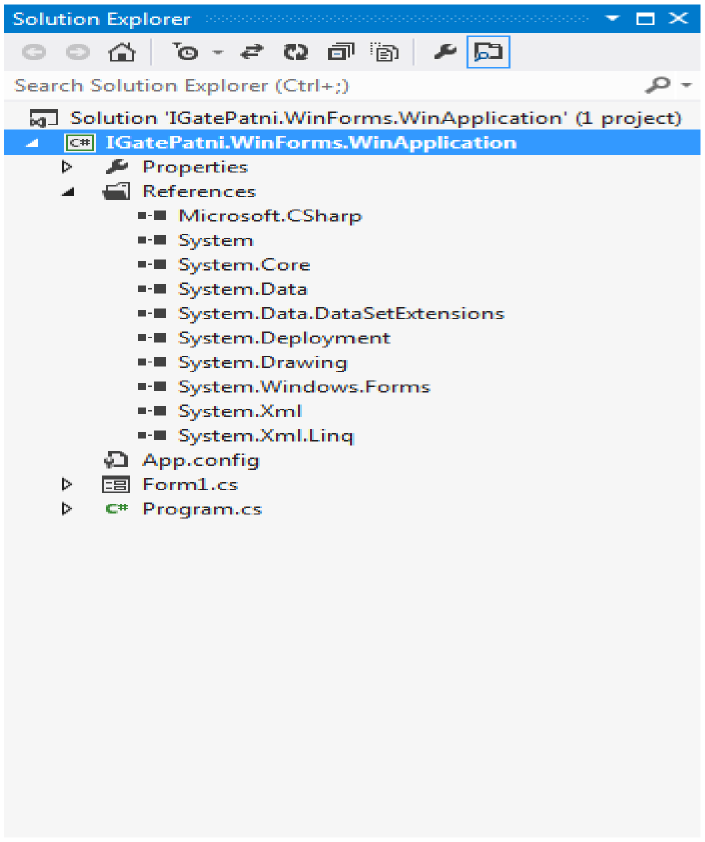
1. Start a new windows form application project
    a. File => New Project, Select "Windows Forms Application" template
    Name it IGatePatni.WinForms.WinApplication


New Project
Recent
Installed
Templates
Visual C#
Windows
Web
Office
Cloud
Reporting
SharePoint
Silverlight
Test
WCF
Workflow
LightSwitch
Other Languages
Other Project Types
Samples
Online
.NET Framework 4.5
Sort by: Default
Search Installed Templates (Ctrl+E)
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
Class Library Visual C#
Portable Class Library Visual C#
WPF Browser Application Visual C#
Empty Project Visual C#
Windows Service Visual C#
WPF Custom Control Library Visual C#
WPF User Control Library Visual C#
Browse...
OK
Cancel

Type: Visual C#
A project for creating an application with a Windows Forms user interface

Name: IGatePatni.WinForms.WinApplication
Location: D:\Nachiket\Study Materials\WinForms\
Solution name: IGatePatni.WinForms.WinApplication
[✓] Create directory for solution
[ ] Add to source control

2. Take a look at solution explorer:

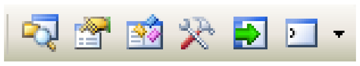


Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Solution 'IGatePatni.WinForms.WinApplication' (1 project)
◢ IGatePatni.WinForms.WinApplication
    ▷ Properties
    ◢ References
        Microsoft.CSharp
        System
        System.Core
        System.Data
        System.Data.DataSetExtensions
        System.Deployment
        System.Drawing
        System.Windows.Forms
        System.Xml
        System.Xml.Linq
    App.config
    ▷ Form1.cs
    ▷ Program.cs
```

- The References node shows assemblies referred
- Form1.cs is the default Windows Form
- Form1.designer.cs file has code; which is auto generated when we drag drop controls on the designer surface and set their properties.
- If required you can add additional “windows form” with Project => Add Window
- You can shift between “form code behind window” and “form designer surface” by using “F7” and “Shift + F7”.
- The Program.cs same as console has Main() function, that launches the startup form using Application.Run() function.
- You would use the following buttons shown in the toolbar frequently.



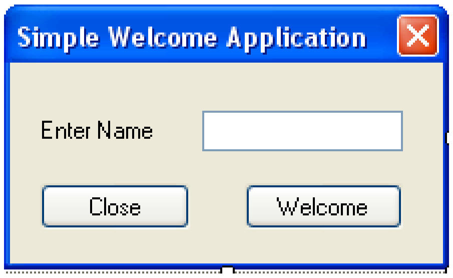
  - Button1: To launch Solution Explorer.
  - Button2: To launch Properties Window
  - Button3: To launch Object Browser to take a look at various assemblies and types in it as well as type metadata.
  - Button4: To launch Toolbox to drag controls.

3. Open Form1 designer and drag 3 controls:

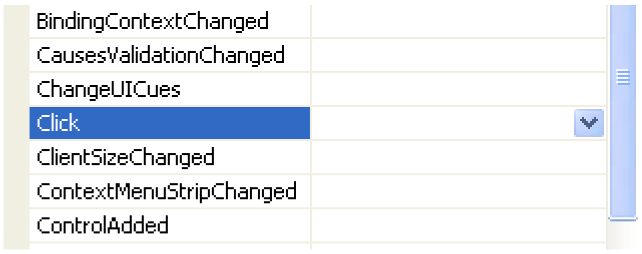
a. Label, TextBox and 2 Buttons.

Select the control and set the properties. You can use F4 to get properties window or click the 2nd button shown in the toolbar above.

Button 1	Name: btnclose Text: Close
Button 2	Name: btnwelcome Text: Welcome
Label	Name: lblname Text: Enter Name
TextBox	Name: txtname
Form	Text: Simple Welcome Application FormBorderStyle: FixedDialog StartPosition: CenterScreen MinimizeBox: False MaximizeBox: False



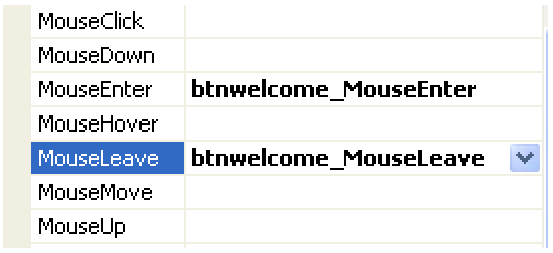
4. To add a click event handler for button; double click welcome button OR select welcome button and go to properties window click the event's small button in property window to get events list
   a. Find click event and double click.



5. Write the following event handling code:

```
private void btnwelcome_Click(object sender, EventArgs e)
{
        MessageBox.Show("Welcome " + txtname.Text + "  !!!");
}
```

6. Select the button and from property window go to MouseEnter event and double click it5 to generate event handler. Similarly generate event handler for MouseLeave.



MouseClick	
MouseDown	
MouseEnter	btnwelcome_MouseEnter
MouseHover	
MouseLeave	btnwelcome_MouseLeave
MouseMove	
MouseUp	

7. The event handler code is given below:

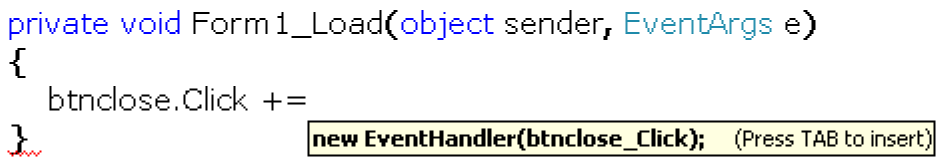
```
private void btnwelcome_MouseEnter(object sender, EventArgs e)
 {
        //sender is source of event ie: btnwelcome button.
     Button b = sender as Button;
          b.BackColor = Color.Yellow;
     b.ForeColor = Color.OrangeRed;
 }

 private void btnwelcome_MouseLeave(object sender, EventArgs e)
 {
          //sender is source of event ie: btnwelcome button.
          Button b = sender as Button;
          b.BackColor = SystemColors.Control;
          b.ForeColor = Color.Black;
  }
```

8. Double click form in free space to generate load event handler and write the following code

```
    private void Form1_Load(object sender, EventArgs e)
 {
            btnclose.Click += new EventHandler(btnclose_Click);
 }
```

The moment you type “btnclose.Click +=” you will get a option to generate event handler automatically as shown below: press TAB twice at this point.



```
private void Form1_Load(object sender, EventArgs e)
{
   btnclose.Click +=
}                    new EventHandler(btnclose_Click);   (Press TAB to insert)
```

9. The close button event handler code is

```
   void btnclose_Click(object sender, EventArgs e)
{
              this.Close();
 }
```

10. Press F5 to run and test the form.

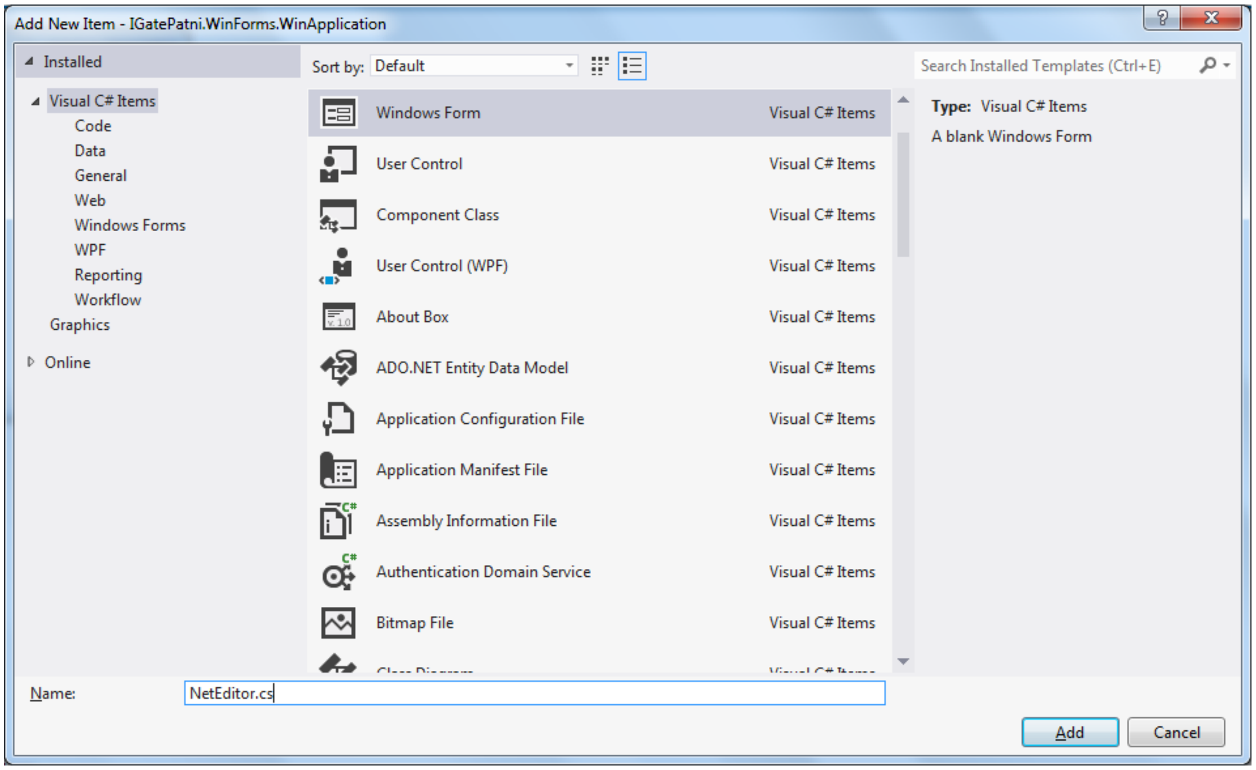
Assignment

To Do:
Write a code so that the textbox gets a yellow background when text cursor comes in.

Lab 2. Understand using MDI application and Menus

Description	We will be creating a MDI Text Editor, which will allow user to open text files as save new ones.
Objective	To Learn: • How to create MDI application • Use Menu Control • Docking controls
Time	90 Mins

1. Add a new windows form and name it NetEditor
    a. Project => Windows Form


Add New Item - IGatePatni.WinForms.WinApplication

Installed
Visual C# Items
Code
Data
General
Web
Windows Forms
WPF
Reporting
Workflow
Graphics
Online

Sort by: Default

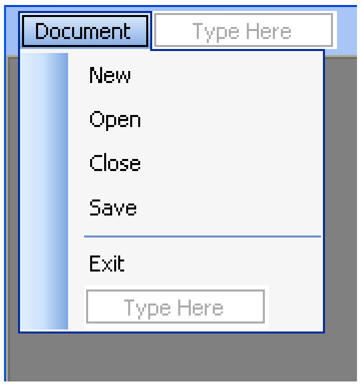
Search Installed Templates (Ctrl+E)

Windows Form	Visual C# Items
User Control	Visual C# Items
Component Class	Visual C# Items
User Control (WPF)	Visual C# Items
About Box	Visual C# Items
ADO.NET Entity Data Model	Visual C# Items
Application Configuration File	Visual C# Items
Application Manifest File	Visual C# Items
Assembly Information File	Visual C# Items
Authentication Domain Service	Visual C# Items
Bitmap File	Visual C# Items

Type: Visual C# Items
A blank Windows Form

Name: NetEditor.cs

2. Set the Form property IsMdiContainer: True
3. Drag a menu strip onto this form and configure menu strip as below:



4. Drag a OpenFileDialog control
5. Add one more “windows form”
    a. Project => Add Windows Form; name it NetEditorChild
6. Drag a textbox onto this form, Set textbox properties
    a. multiline property: True
    b. Dock: Fill.
7. Set Form properties
    a. MinimizeBox: False
    b. MaximizeBox: False
8. Double click the menu items from NetEditor form one by one to generate click event handlers
9. The code is given below:

```
    private void newToolStripMenuItem_Click(object sender, EventArgs e)
{
   NetEditorChild nec = new NetEditorChild();
   nec.Text = "New Document"; nec.MdiParent = this;
   nec.Show();
}

private void closeToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (this.ActiveMdiChild != null)
      this.ActiveMdiChild.Close();
}

private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
   this.Close();
}

    private void openToolStripMenuItem_Click(object sender, EventArgs e)
{
   openFileDialog1.Filter = "Text Files|*.txt|XML Files|*.xml";
   if (openFileDialog1.ShowDialog() == DialogResult.OK)
   {
      StreamReader sr= new StreamReader(File.OpenRead(openFileDialog1.FileName));
```

```
            NetEditorChild nec = new NetEditorChild();
            nec.Text = openFileDialog1.FileName;
            nec.textBox1.Text = sr.ReadToEnd();
            sr.Close();
            nec.MdiParent = this;
            nec.Show();
        }
    }
```

10. Make the NetEditor Form as startup. Open Program.cs and change main function to:

```
    [STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new NetEditor());
 }
```

11. Run the application
    a. Click Open menu item, Locate any .txt file and click open
    b. Similarly you can locate and open any .xml file.
    c. click exit

Assignment

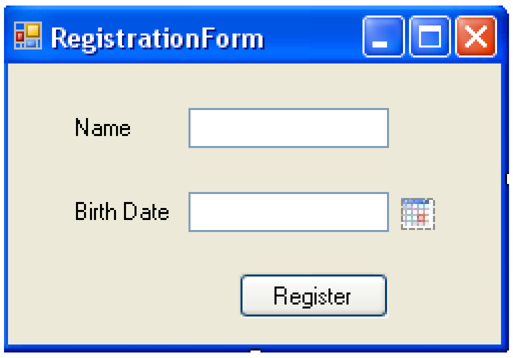
To Do:

Write the code for savemenuitem which should open save as dialog box if the data is not already saved and then save the entire contents of the textbox into the file. If the contents were already saved earlier it should not popup save as dialog box and save contents within same file.

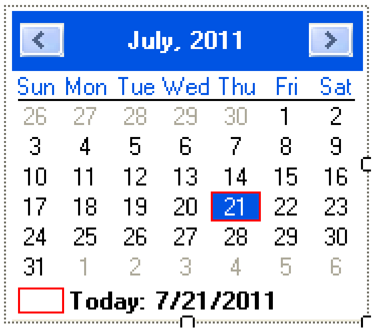
Lab 3. Showing Custom Dialog

Description	We will be having a small registration which uses custom calendar dialog box.
Objective	To Learn: • How to show custom dialog box and get the required input from the user.
Time	60 Mins

1. Add a new Windows Form, Name it RegistrationForm.
2. Drag 2 labels, 2 textboxes (txtname, txtbirthdate), 1 picturebox (imgcal), 1 button (btnregistration).



3. For picturebox control associate a miniature calendar image.
4. Add a new Windows Form, Name it CalendarDialog
5. Drag a MonthCalendar control. Rename the control to “DateSelection”
6. Set BorderStyle property of this form to none.



7. set modifier property of calendar control to “Public”
8. In the keyup event of this calendar write following code:

```
 private void DateSelection_KeyUp(object sender, KeyEventArgs e)
{
        if (e.KeyCode == Keys.Escape)
       {
               this.DialogResult = DialogResult.Cancel;
               this.Close();
     }
else if (e.KeyCode == Keys.Enter)
     {
         this.DialogResult = DialogResult.OK;
               this.Close();
   }
}
```

9. In the registration form write the following code for click event on picturebox control.

```
  private void imgcal_Click(object sender, EventArgs e)
{
             CalendarDialog cal = new CalendarDialog();
             cal.StartPosition = FormStartPosition.CenterScreen;
             if (cal.ShowDialog() == DialogResult.OK)
                txtbirthdate.Text = cal.DateSelection.SelectionStart.
                                                          ToString("dd-
                                                 MMM-yyyy");
}
```

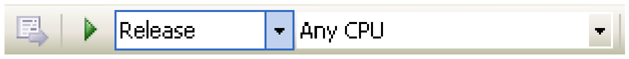
10. Make the RegistrationForm as startup and execute.

Lab 4. Deployment using ClickOnce

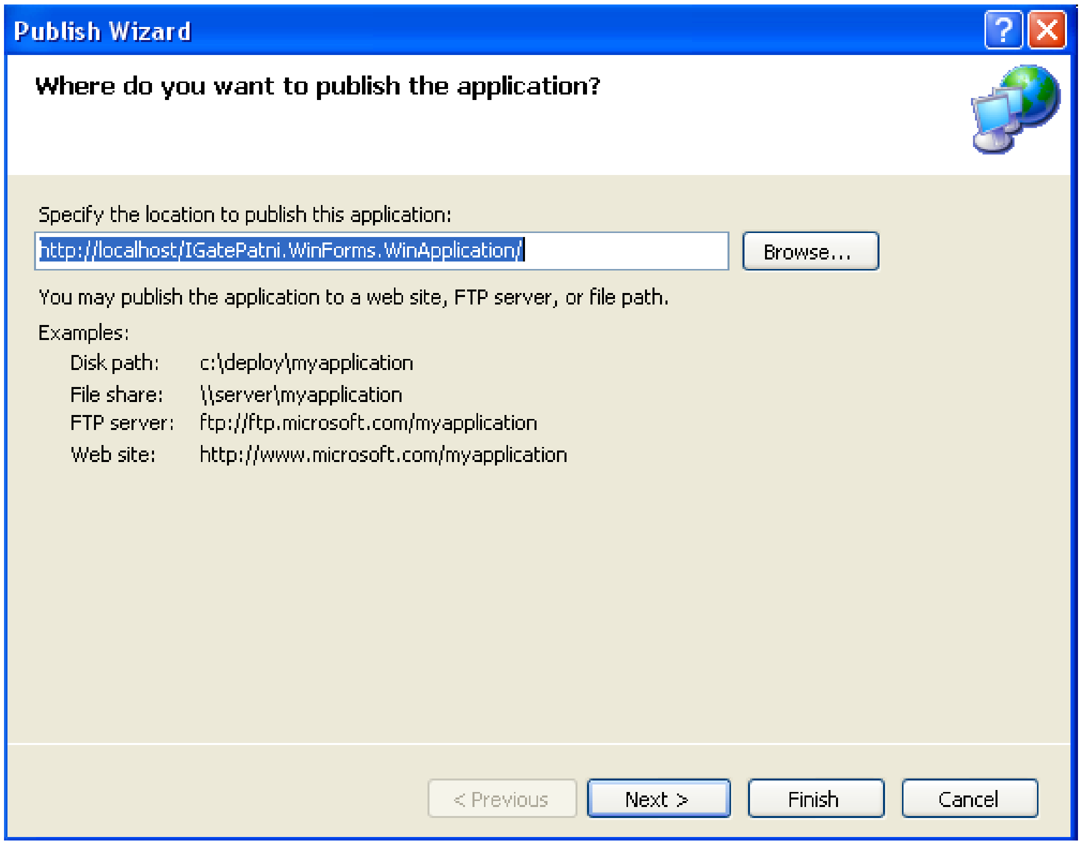
Description	We will be deploying our current windows application using click once deployment feature.
Objective	To Learn: • How to deploy a windows application using ClickOnce • Check the auto-updates feature of ClickOnce
Time	60 Mins

To perform this lab admin rights on local system and IIS installation is required

1. Through Main() function in Program.cs make the NetEditor Form as Startup form
2. Set the Build Configuration of the application to “Release” from the drop down above in the toolbar.



3. Build the project.
4. Publish the application Setup on the local IIS web server
    a. Build => Publish OR Right Click the project in solution explorer and select Publish
5. A publish popup dialog will come up:



Where do you want to publish the application?

Specify the location to publish this application:

http://localhost/IGatePatni.WinForms.WinApplication/

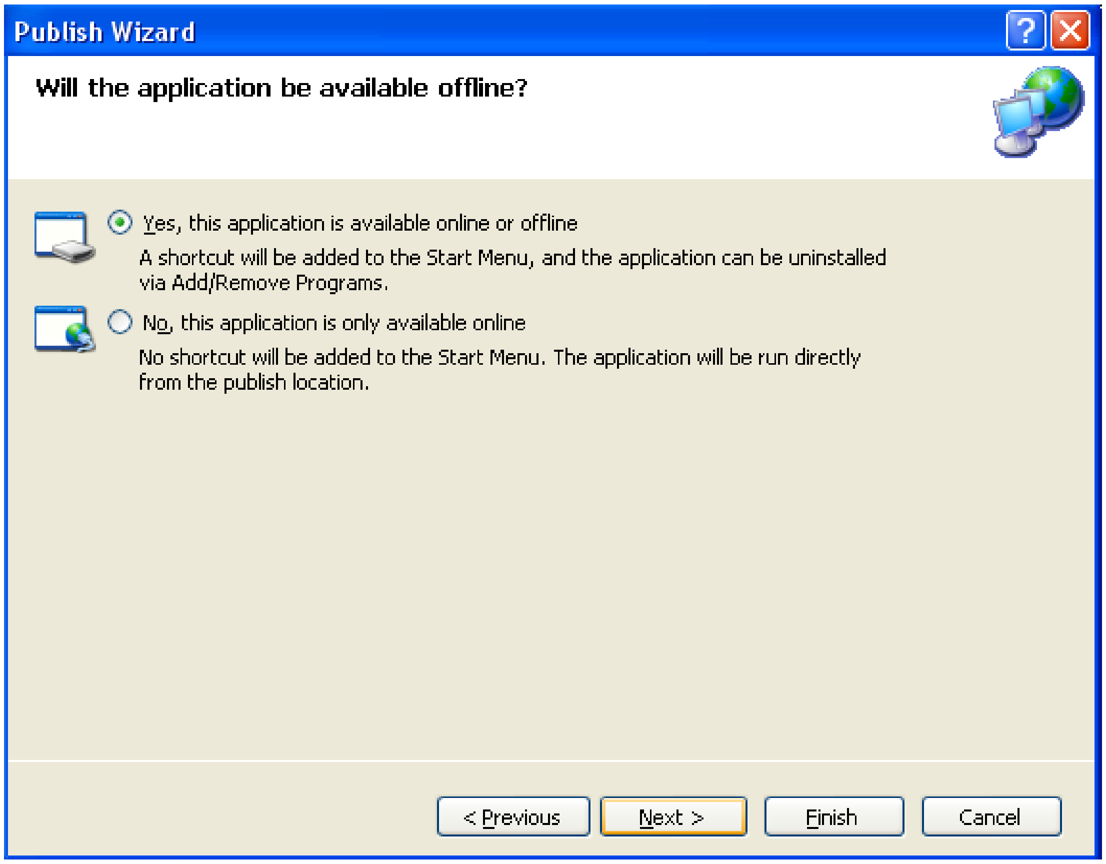
You may publish the application to a web site, FTP server, or file path.

Examples:

Disk path:	c:\deploy\myapplication
File share:	\\server\myapplication
FTP server:	ftp://ftp.microsoft.com/myapplication
Web site:	http://www.microsoft.com/myapplication

The popup will ask for the URL for publishing the Application Setup.

6. Let the URL be default, Click Next
7. In the next screen select the “application will be available offline or online” so that the application will get installed locally.

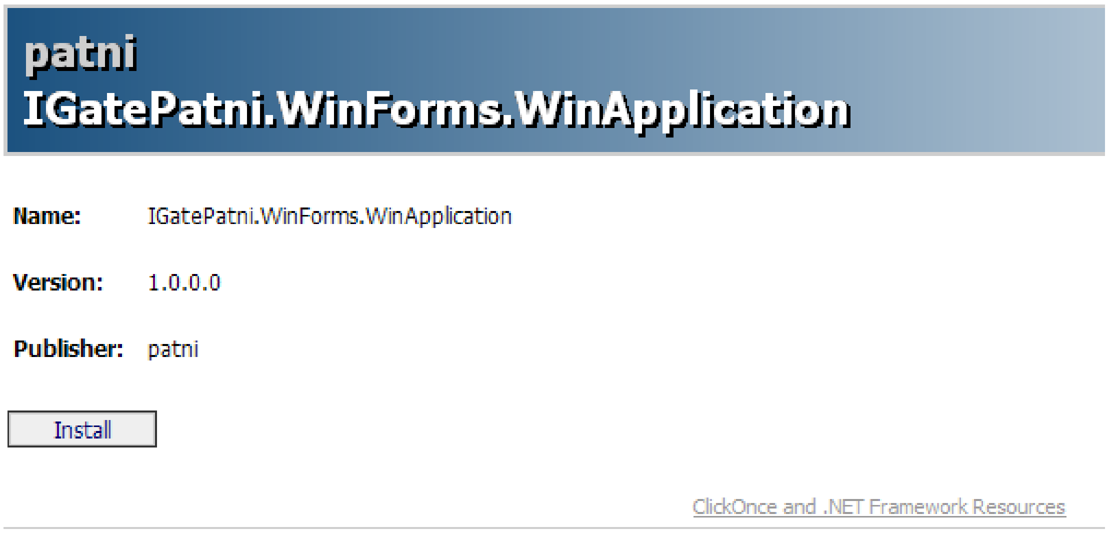

Publish Wizard

Will the application be available offline?

[✓ Yes, this application is available online or offline]
A shortcut will be added to the Start Menu, and the application can be uninstalled via Add/Remove Programs.

No, this application is only available online
No shortcut will be added to the Start Menu. The application will be run directly from the publish location.

8. Click Next.
9. Click Finish on last screen.
10. The setup will be published and browser will show you the URL.


patni
IGatePatni.WinForms.WinApplication

Name: IGatePatni.WinForms.WinApplication

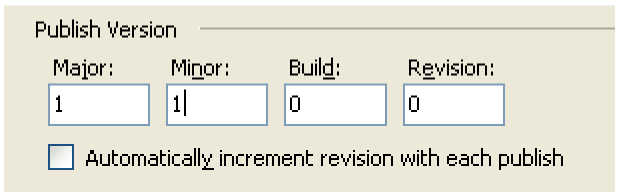
Version: 1.0.0.0

Publisher: patni

11. Click Install to install application locally.
12. Run the application from the programs menu.

Lets check auto update feature

13. Open Visual Studio and change the BackColor of TextBox on NetEditorChild Form to Yellow. Rebuild the project.
14. Go to Project => Properties OR Right Click Project in solution explorer and select properties, Move to Publish TAB.
15. Make the build version as 1.1.0.0, click save

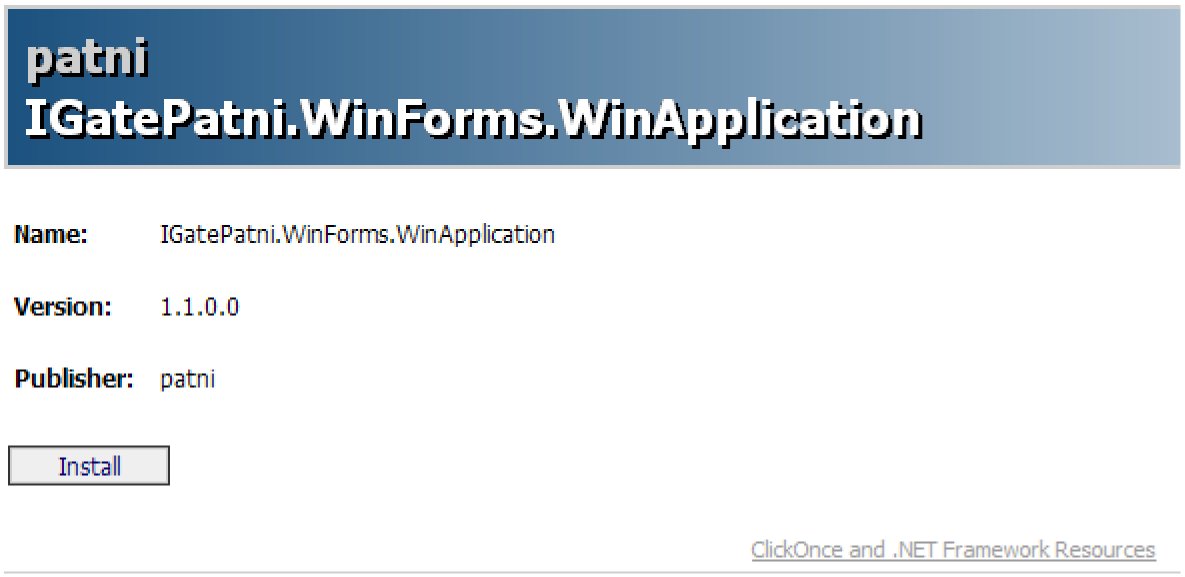

Publish Version

Major:	Minor:	Build:	Revision:
1	1	0	0

☐ Automatically increment revision with each publish

16. Click “Publish Now”.
17. New 1.1.0.0 setup will be now available

Do not click Install



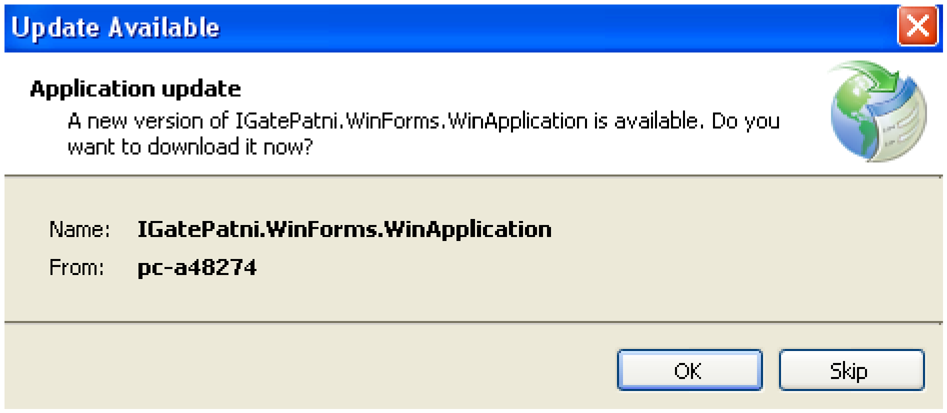
patni
IGatePatni.WinForms.WinApplication

Name: IGatePatni.WinForms.WinApplication

Version: 1.1.0.0

Publisher: patni

18. Run the application from desktop it will detect updates are available



Application update
A new version of IGatePatni.WinForms.WinApplication is available. Do you want to download it now?

Name: **IGatePatni.WinForms.WinApplication**
From: **pc-a48274**

19. Click Ok to download updates.
20. Run the application and check that textbox back color will be yellow. It means the changes were updated.