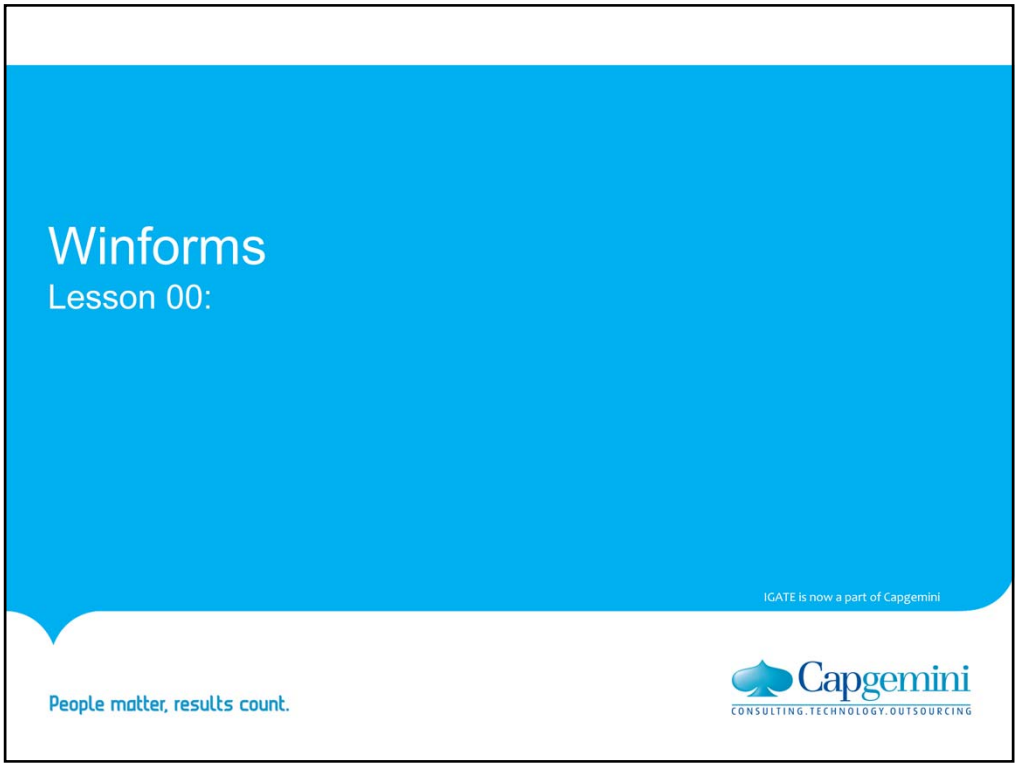


Document History

Date	Course Version No.	Software Version No.	Developer / SME	Change Record Remarks
		Microsoft Visual C# 2012 Express Edition	Ganesh Desai	Draft version
16-Oct-2009	1.0	10-Oct-2009	1.0	Review
10-July-2011	2.0	Visual Studio 2008	Ganesh Desai	Changes in material made based on integration process
19-July-2013	3.0		Vaishali Kasture	Changes done and added new contents

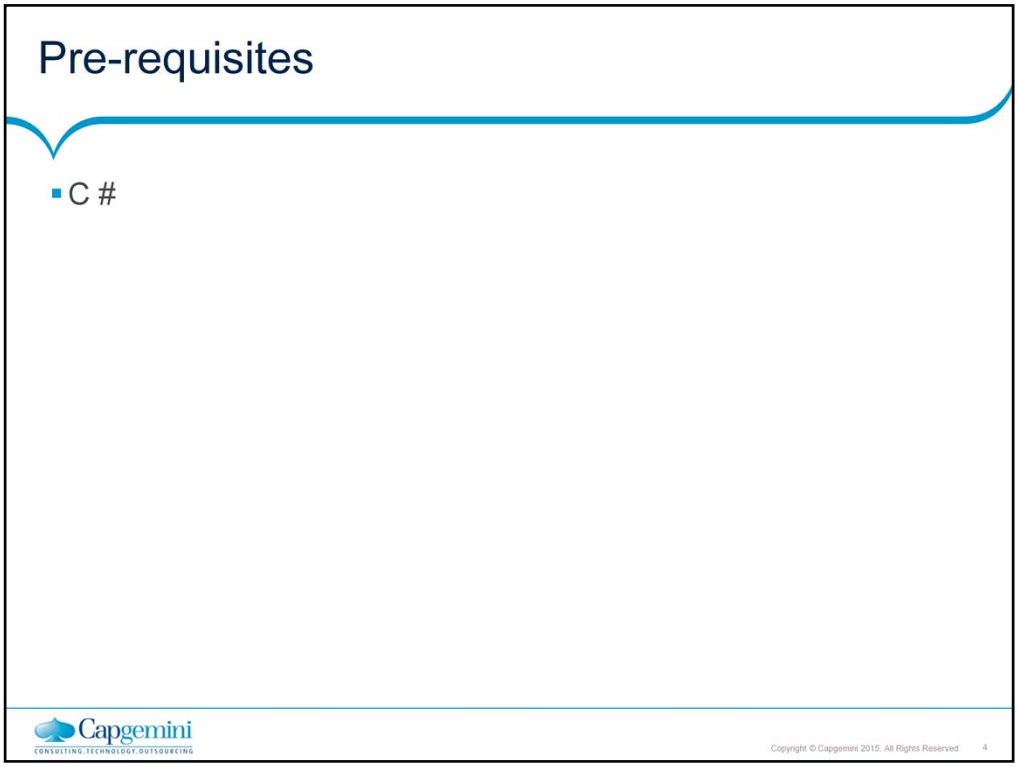
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Course Goals and Non Goals

- Course Goals
  - To develop Console and Windows based Desktop applications using C# 5.0
- Course Non Goals
  - Developing Distributed multi-tier Desktop Applications using C#

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

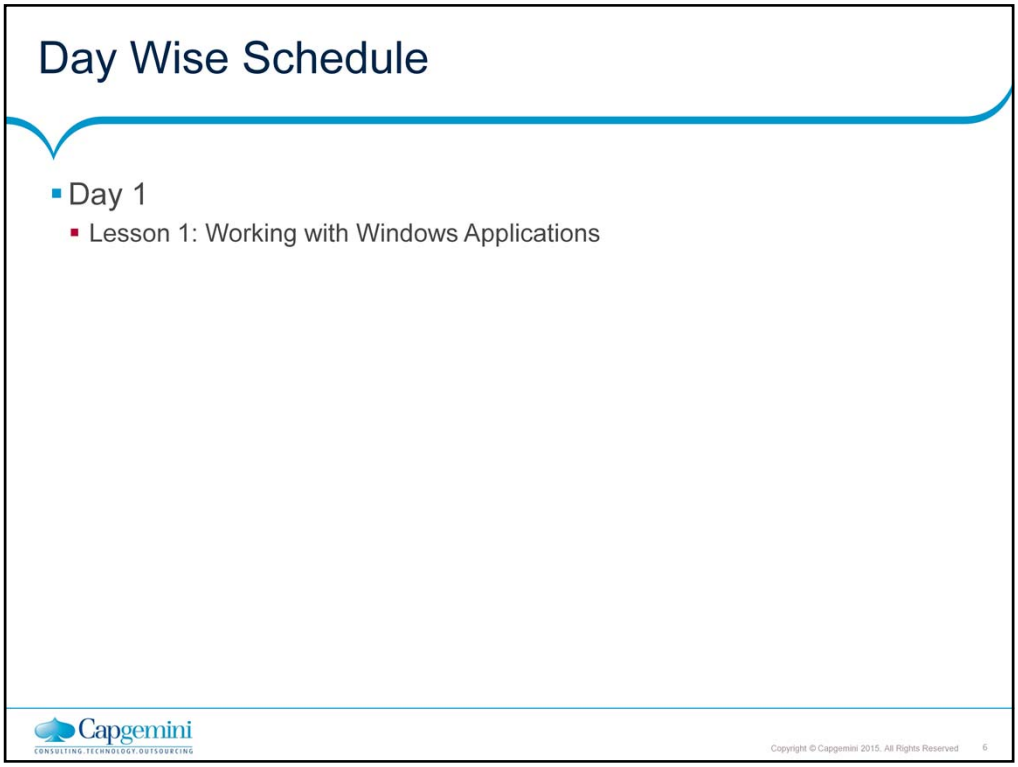

Pre-requisites
C #
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
4

Intended Audience

- Developers
- Test Engineers

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

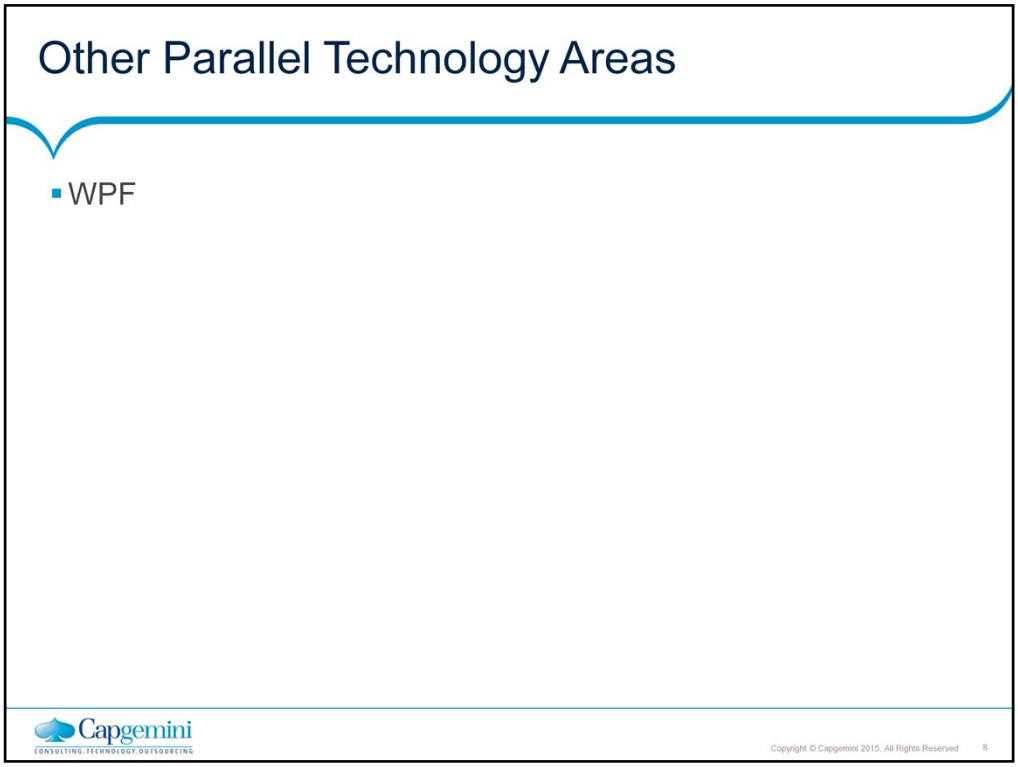
 5


Day Wise Schedule
Day 1
Lesson 1: Working with Windows Applications
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
6

References

- References (books, URL)
  - http://www.microsoft.com/msdn

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING


Other Parallel Technology Areas
WPF
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
8

Winforms

Lesson 01:Working with Windows Applications

References

- Microsoft.com/msdn
- MCTS Self Paced Training Kit (70-526)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

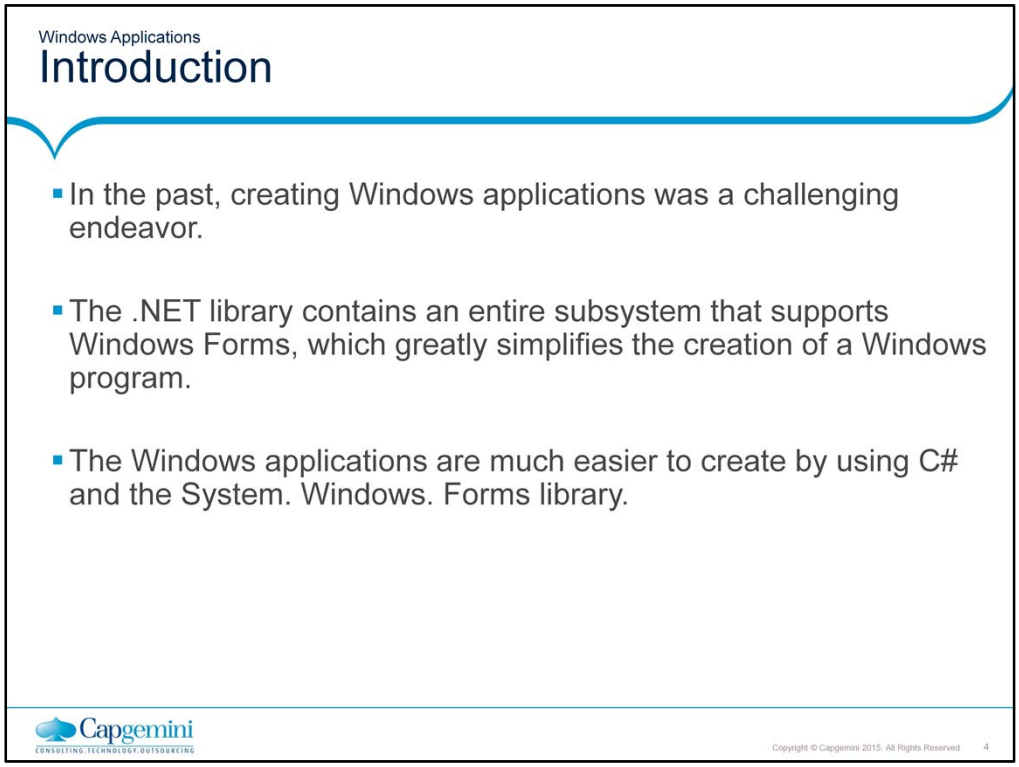
 2

Lesson Objectives

- In this lesson, you will learn:
  - The difference between Console and Windows Application
  - The challenges for Windows Applications
  - Event Handling Model in .NET
  - Different controls and their properties
  - The creation of MDI Applications and Menus in C#

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3



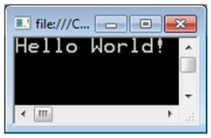
Windows Applications:
C# and the .NET Framework's Forms library offer a fully object-oriented way to approach Windows programming.
Instead of providing just a wrapper around the API, the Forms library defines a streamlined, integrated, logically consistent way of managing the development of a Windows application. This level of integration is made possible by the unique features of the C# language, such as delegates and events. Furthermore, because of C#'s use of garbage collection, the troubling problem of "memory leaks" especially has been nearly eliminated.

C# Console Application
Explanation

- Library imports
- Class and namespace definitions

```
using System;
class HelloWorld
{
   static void Main(string[] args)

   {
      Console.WriteLine("Hello World!");
   }
}
```

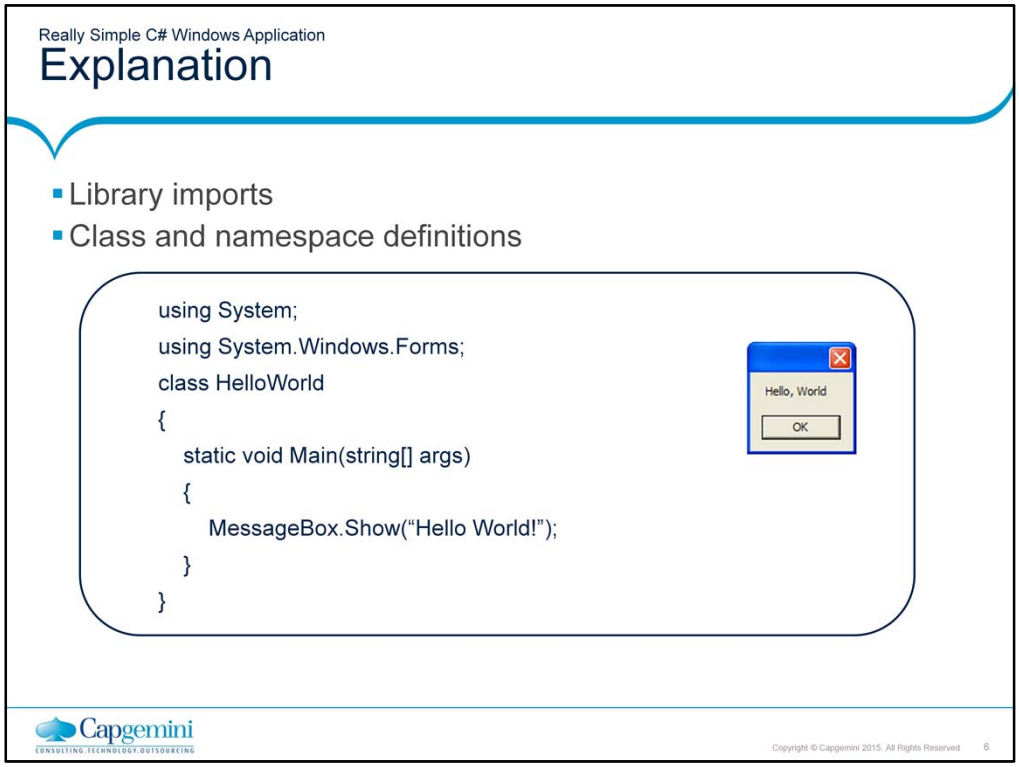


Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

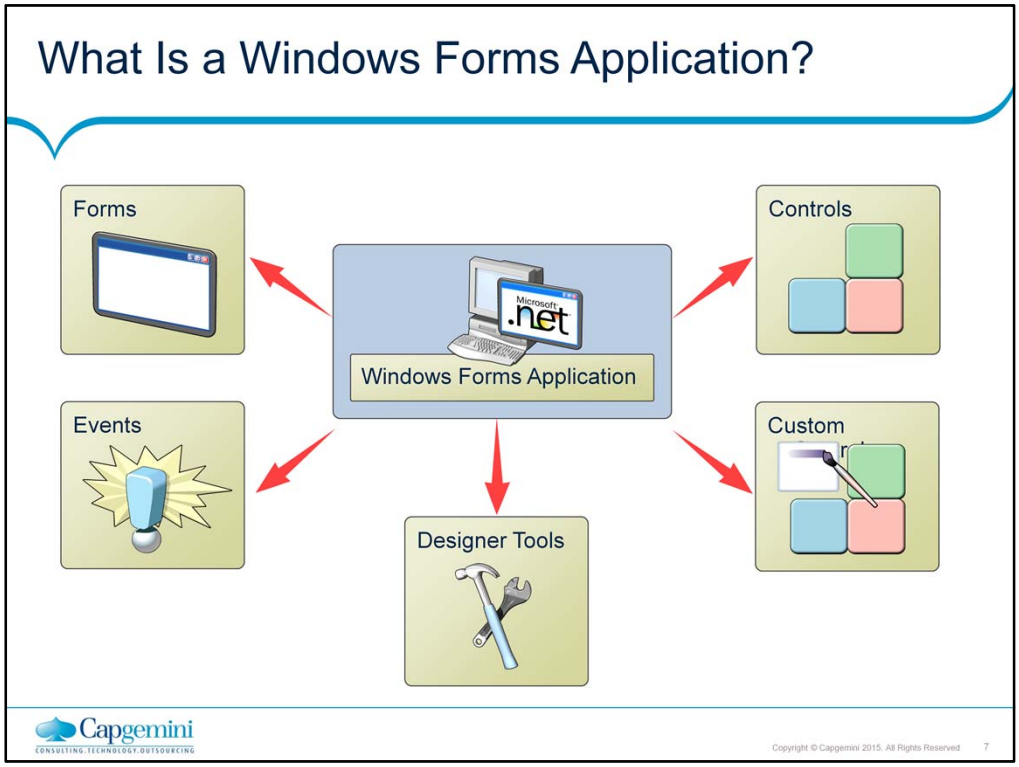
C# Console Application:
C# Console Application has the following characteristics:

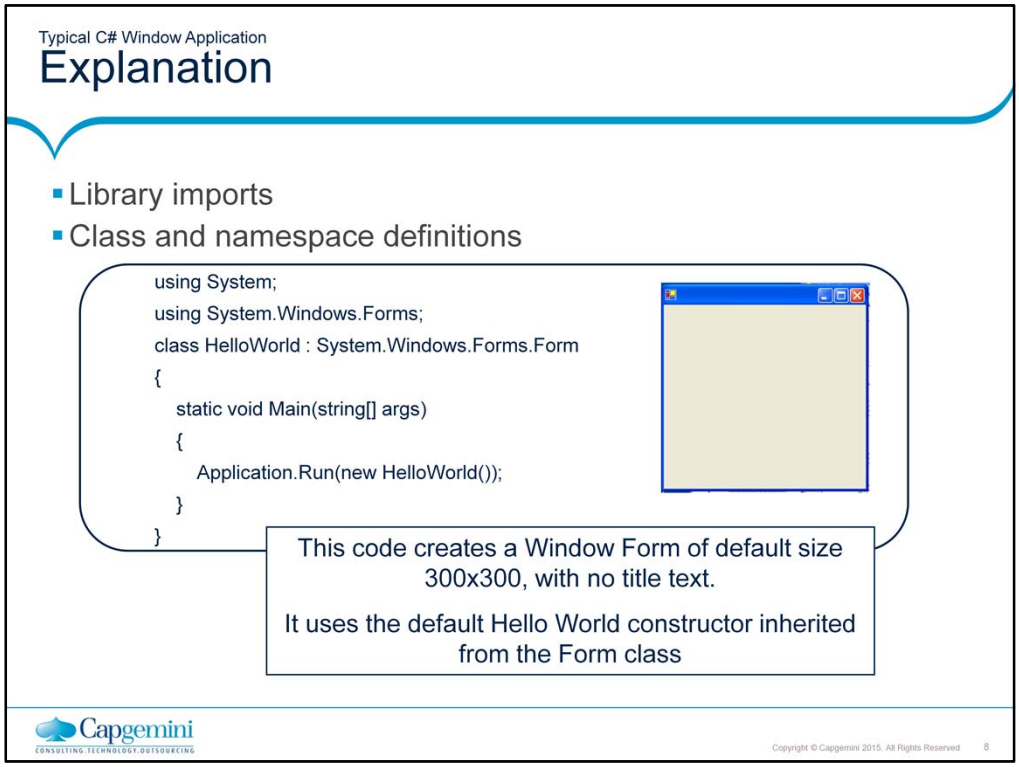
- No visual component
- Only text input and output
- Run under Command Prompt or DOS Prompt



C# Windows Application:
It forms with many different input and output types.
It contains Graphical User Interfaces (GUI).
GUIs make the input and output more user friendly!
Message boxes:

- They are used within the System.Windows.Forms namespace.
- They are used to prompt or display information to the user.


What Is a Windows Forms Application?
Forms
Controls
Microsoft .net
Windows Forms Application
Events
Custom
Designer Tools
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
7



Typical C# Window Application:
A form is created by instantiating an object of the Form class, or of any class derived from Form.
A form contains significant functionality of its own, and it inherits additional functionality. Two of its most important base classes are System.ComponentModel.Component and System.Windows.Forms.Control. The Control class defines features common to all Windows controls. Since Form inherits Control, it too is a control. This fact allows forms to be used to create controls.

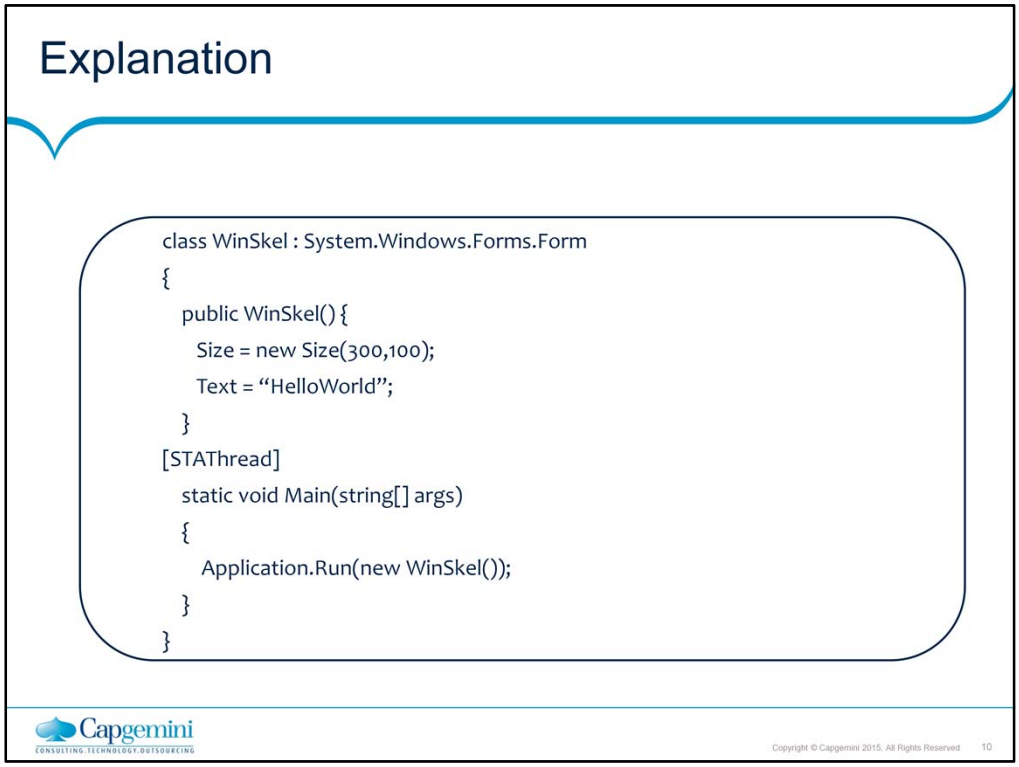
Explanation

- Library imports
  - using System;
  - using System.Drawing;
  - using System.Windows.Forms;
- Class and namespace definitions

New library that defines the Size class!

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9



Typical C# Window Application (contd.):
Let us examine this program line by line.
First, notice that both System and System.Windows.Forms are included. System is needed because of the STAThread attribute that precedes Main( ). The System.Windows.Forms supports the Windows forms subsystem, as just explained.
Next, a class called WinSkel is created. It inherits Form. Thus, WinSkel defines a specific type of form. In this case, it is a minimal form.

Inside the WinSkel constructor, there is the following line of code:

Text = “HelloWorld";

Text is the property that sets the title of the window. Thus, this assignment causes the title bar in the window to contain “HelloWorld”. Text is defined as shown below:

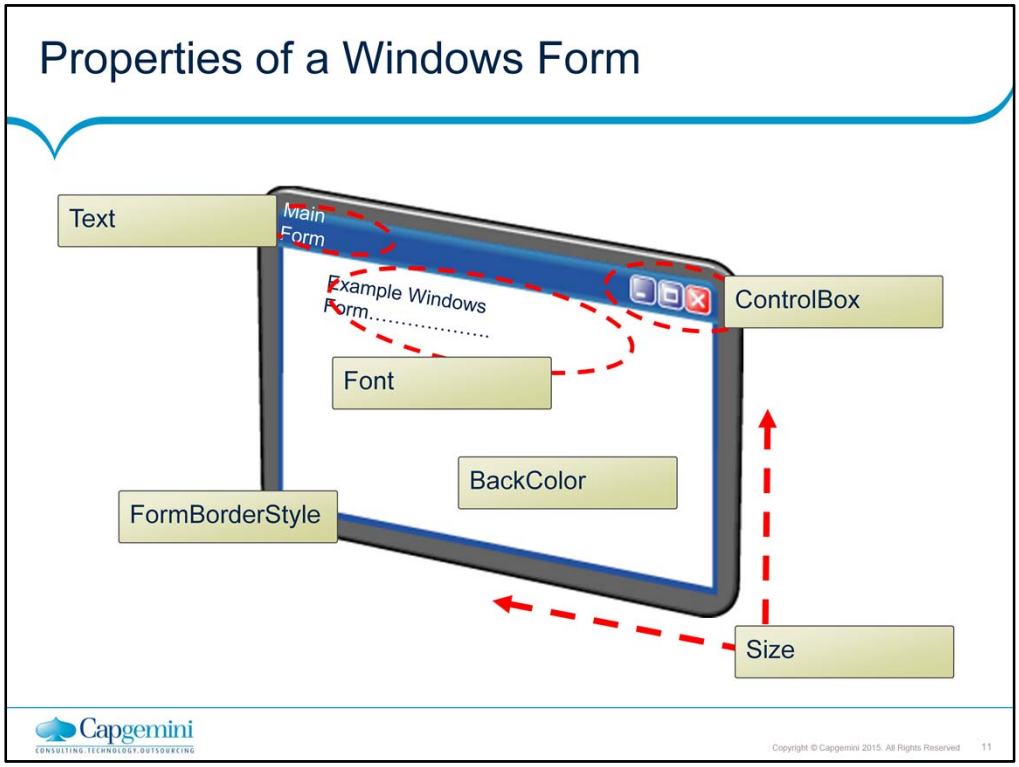
public override string Text { get; set; } Text is inherited from Control.

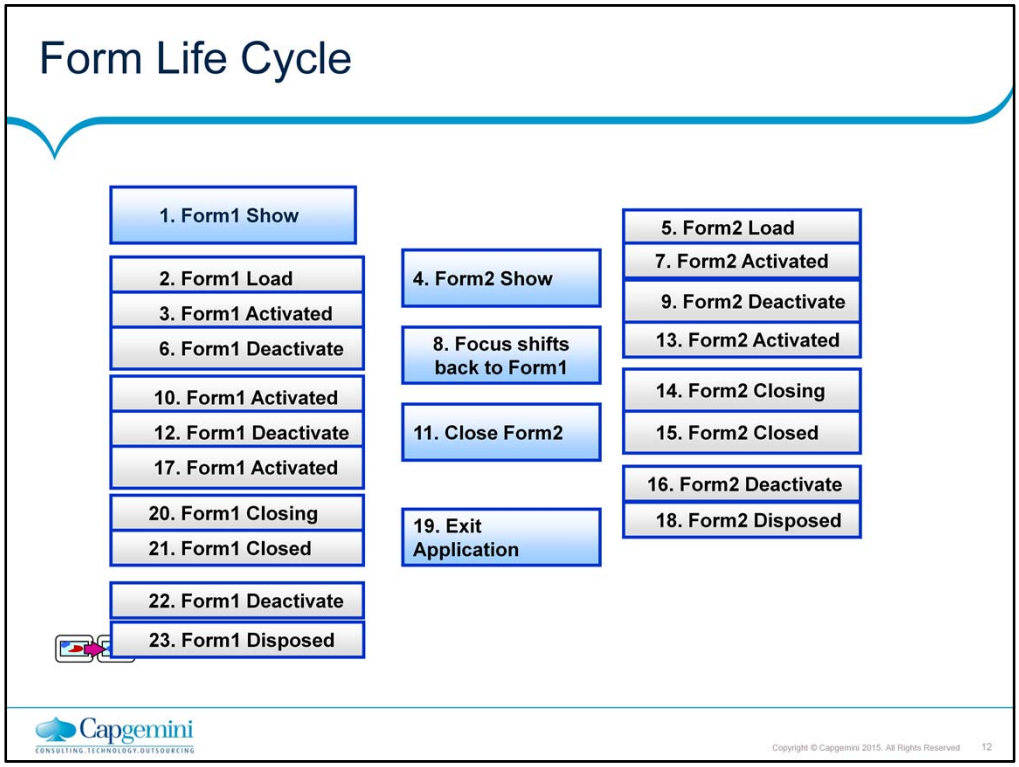
Next is the Main( ) method. It is the method at which program execution begins. Notice, however, that it is preceded by the STAThread property. As a general rule, the Main( ) method of a Windows program should have this property. It sets the threading model for the program to a single-threaded apartment (STA). Inside Main( ), a WinSkel object is created. This object is then passed to the Run( ) method defined by the Application class.
Application.Run(new Winskel());

This starts the window running. The Application class is defined within System.Windows.Forms, and it encapsulates aspects common to all Windows applications. The Run( ) method used by the skeleton is shown here:

public static void Run(Form ob)

It takes a reference to a form as a parameter. Since WinSkel inherits Form, an object of type WinSkel can be passed to Run( ).


Properties of a Windows Form
Text
Main Form
Example Windows Form.....
ControlBox
Font
BackColor
FormBorderStyle
Size
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
11


Form Life Cycle
1. Form1 Show
2. Form1 Load
3. Form1 Activated
6. Form1 Deactivate
10. Form1 Activated
12. Form1 Deactivate
17. Form1 Activated
20. Form1 Closing
21. Form1 Closed
22. Form1 Deactivate
23. Form1 Disposed
4. Form2 Show
8. Focus shifts back to Form1
11. Close Form2
19. Exit Application
5. Form2 Load
7. Form2 Activated
9. Form2 Deactivate
13. Form2 Activated
14. Form2 Closing
15. Form2 Closed
16. Form2 Deactivate
18. Form2 Disposed
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
12

Demo

- Demo on creating Windows Form based Application

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Challenges for Window Applications

Explanation

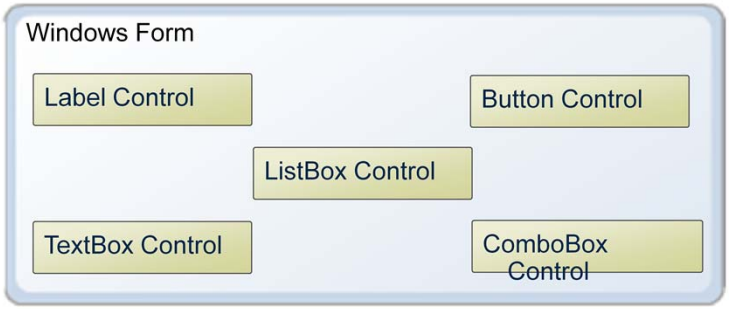
- What are basic inputs?
  - User action, system actions are basic inputs.
  - Some inputs come directly from user actions, some are indirect and implicit.
- How to react to these inputing “events”?
  - Need to write what we call “event handlers”.
- What are the outputs?
  - Display different GUI widgets with proper layout
  - Graphics programming
  - Animation

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

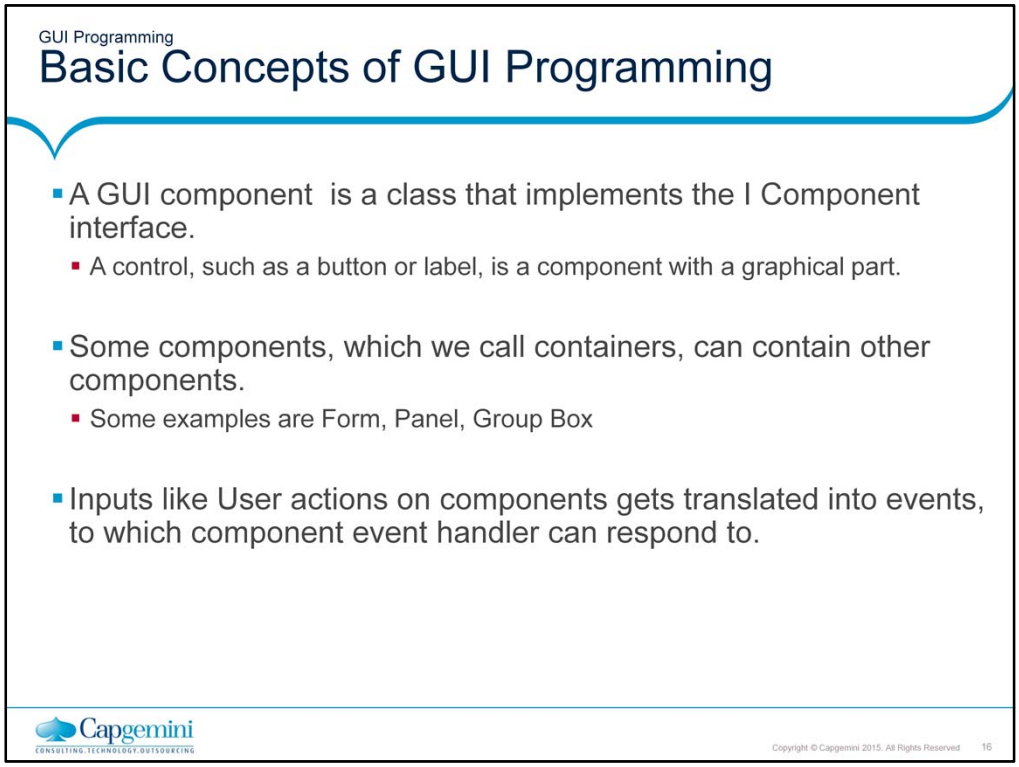
What Are Controls?

- Example Common Controls:



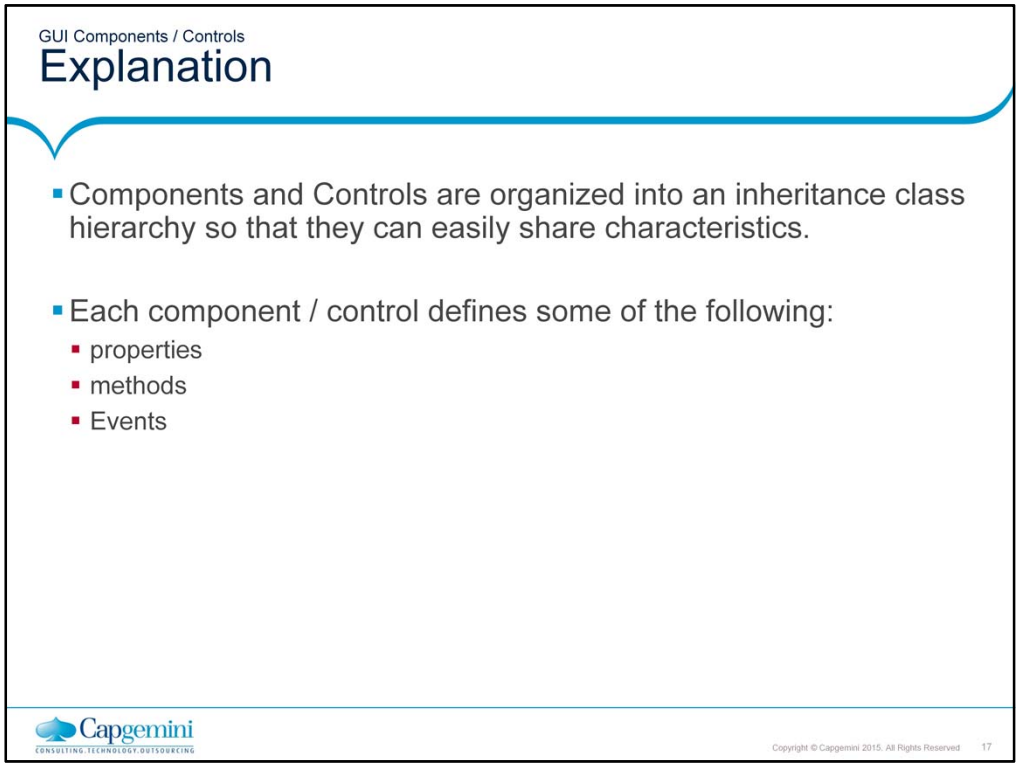
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15



GUI Programming:
The first class of immediate interest is Component. The Component type provides a canned implementation of the IComponent interface. In the .NET Framework, a component is a class that implements the System.ComponentModel.IComponent interface or that derives directly or indirectly from a class that implements IComponent.
In programming, the term component is generally used for an object that is reusable and can interact with other objects. A .NET Framework component satisfies those general requirements, and additionally provides features such as:

control over external resources
design-time support

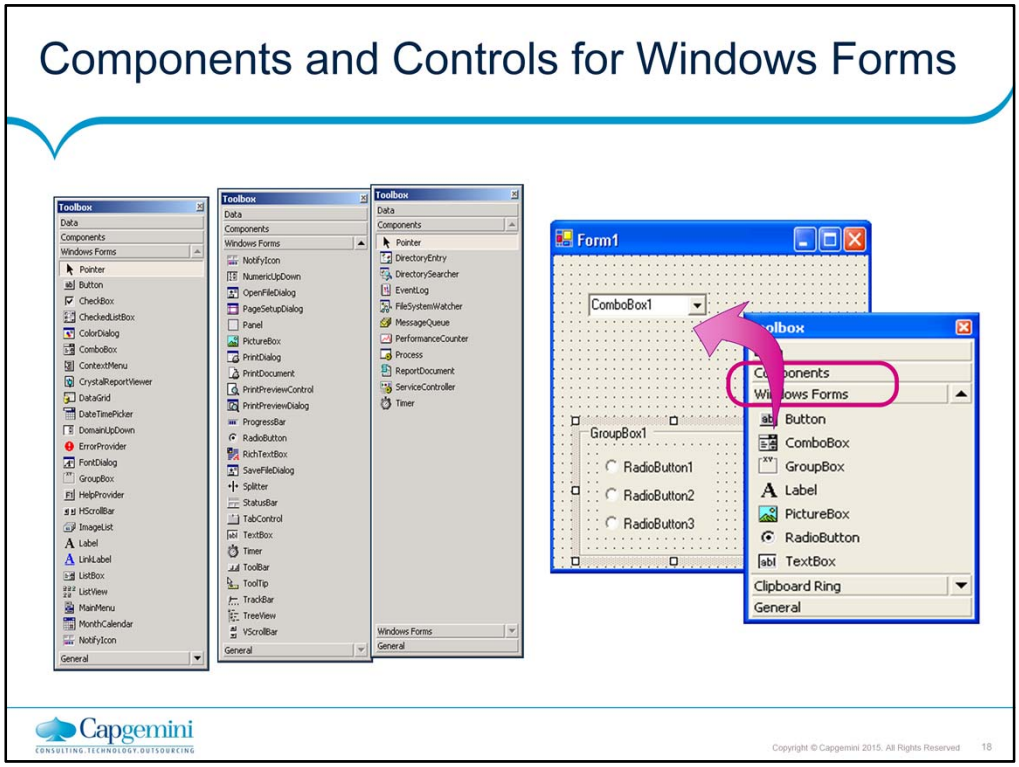


GUI Components / Controls:
The next base class of interest is System.Windows.Forms.Control which establishes the common behaviors required by any GUI-centric type.
A control is a component that provides (or enables) user-interface (UI) capabilities.
The .NET Framework provides two base classes for controls:

one for client-side Windows Forms controls, and
other for ASP.NET server controls
These classes are as follows:
System.Windows.Forms.Control
System.Web.UI.Control.

All controls in the .NET Framework class library derive directly or indirectly from these above two classes.

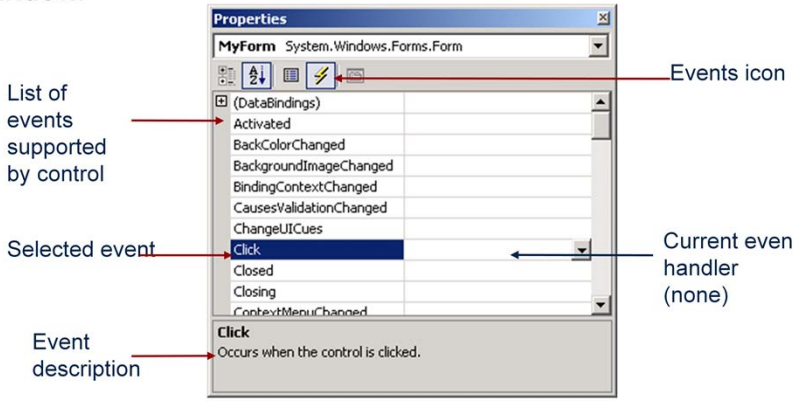
System.Windows.Forms.Control derives from Component and itself provides UI capabilities.
System.Web.UI.Control implements IComponent and provides the infrastructure on which it is easy to add UI.



GUI Components / Controls (contd.):
The System.Windows.Forms namespace contains a number of types that represent common GUI widgets. Using these types, you can respond to user inputs in a Windows Forms application.

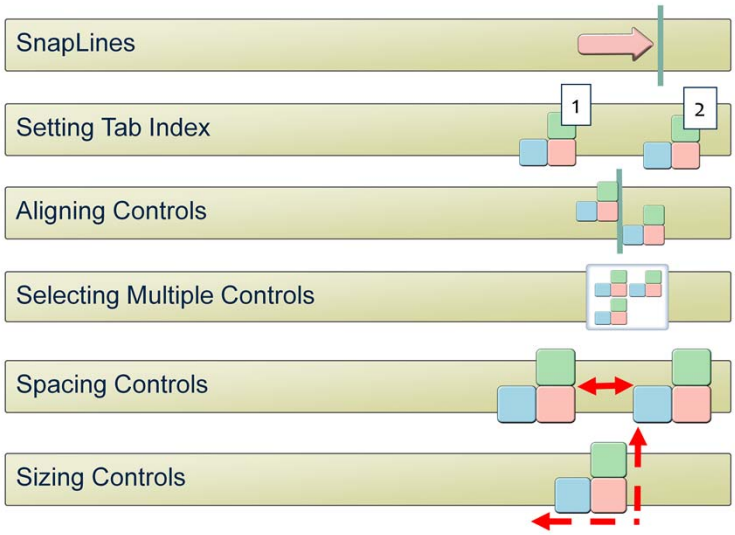
Explanation

- The following figure displays the Events section of the Properties window.



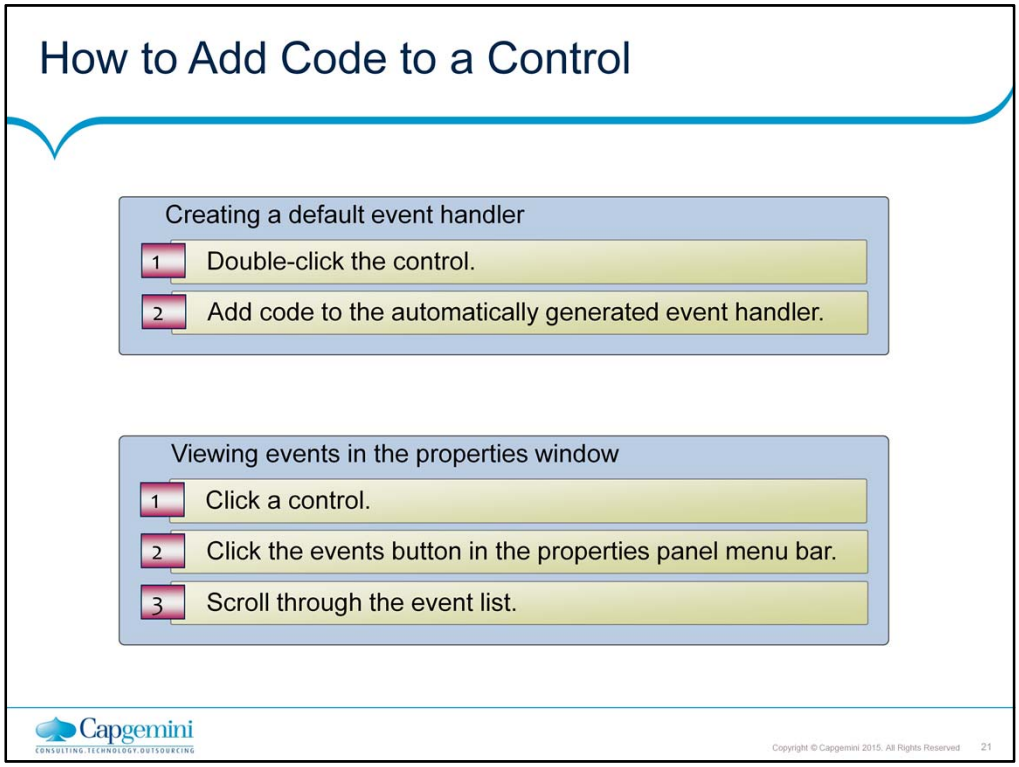
Events section of the **Properties** window.

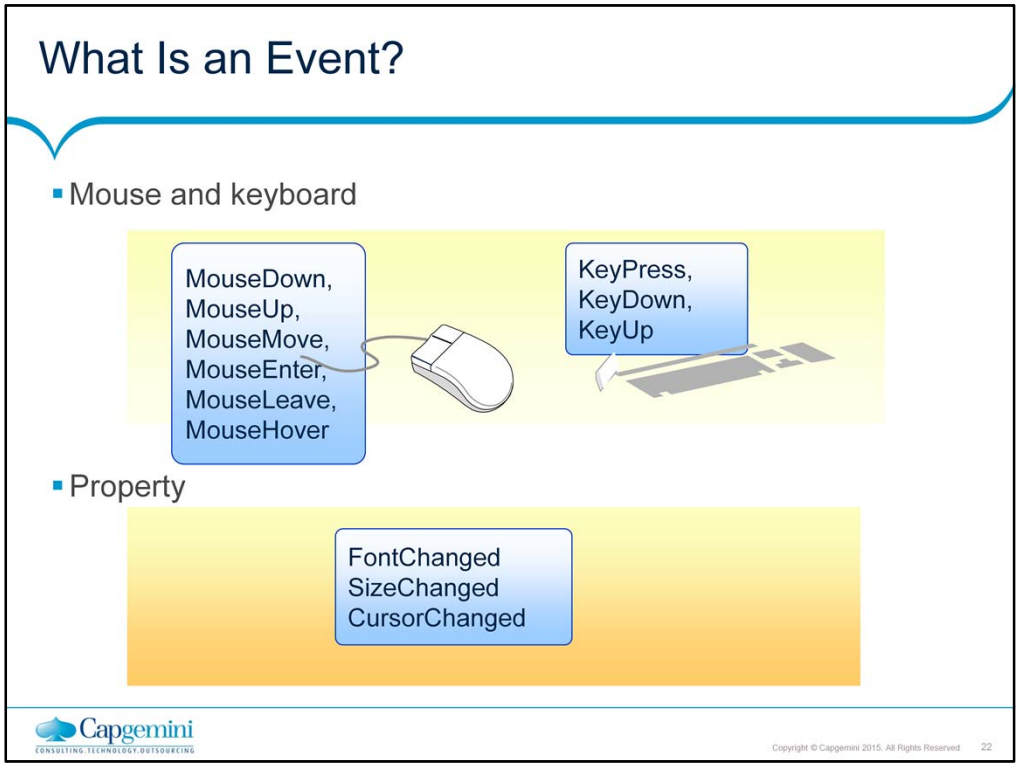
How to Define the Layout of Controls



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

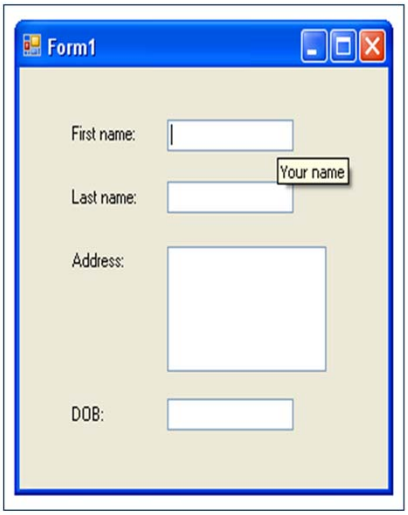

How to Add Code to a Control
Creating a default event handler
1 Double-click the control.
2 Add code to the automatically generated event handler.
Viewing events in the properties window
1 Click a control.
2 Click the events button in the properties panel menu bar.
3 Scroll through the event list.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 21

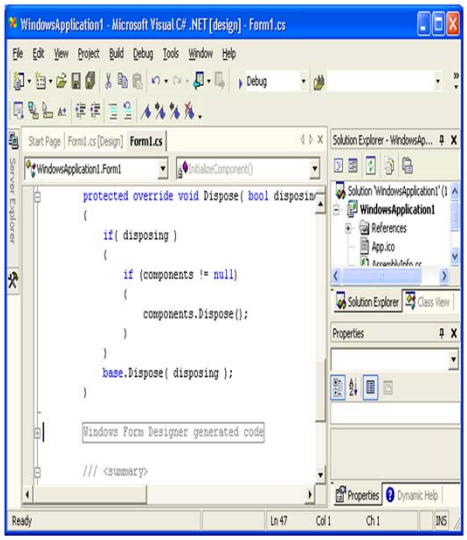


Event can be either
A user action such as MouseClick or
System/Application generated such as Form_Load

Event Handler is a method that processes an event and perform the tasks.

Windows Forms & Designer-Generated Code





Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

Explanation

- In a control, there is one delegate for each event it can handle.
  - A delegate of an event is an object which references (a list of) methods, that is, event handlers.

Event handler:

It must conform to a given signature.

It must be registered with the delegate object of the event of the control.

Add event handlers to the delegate's invocation list.

Event multicasting:

It can have multiple handlers for one event.

Order called for event handlers is indeterminate.

Example:

```
//Step 1. Class that defines data for the event
//
public class AlarmEventArgs : EventArgs
  {
    private readonly bool snoozePressed = false;
    private readonly int nrings = 0;
    // Constructor.
    public AlarmEventArgs(bool snoozePressed, int nrings) {...}
    // Properties.
    public int NumRings{ get { return nrings;}}
    public bool SnoozePressed { get { return snoozePressed;}}
    public string AlarmText { get {...}}
  }

//Step 2. Delegate declaration.
//
public delegate void AlarmEventHandler(object sender, AlarmEventArgs e);

// Class definition.
//
public class AlarmClock
{
//Step 3. The Alarm event is defined using the event keyword.
//The type of Alarm is AlarmEventHandler.
  public event AlarmEventHandler Alarm;
//
//Step 4. The protected OnAlarm method raises the event by invoking
//the delegates. The sender is always this, the current instance of
//the class.
//
protected virtual void OnAlarm(AlarmEventArgs e)
  {
   if (Alarm != null)
    {
      //Invokes the delegates.
      Alarm(this, e);
    }
  }
}
```

Delegate and Events

- A delegate for the click event of a button. The argument to the constructor contains a reference to the method that performs the event handling logic.
  - EventHandler handler = new EventHandler(button_Click);
  - // Create button, sets their properties, and attaches // an event handler to button.
    - red = new Button();
    - red.Text = "Red";
    - red.Location = new Point(100, 50);
    - red.Size = new Size(50, 50);
    - red.Click +=handler;
    - Controls.Add(red);

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26

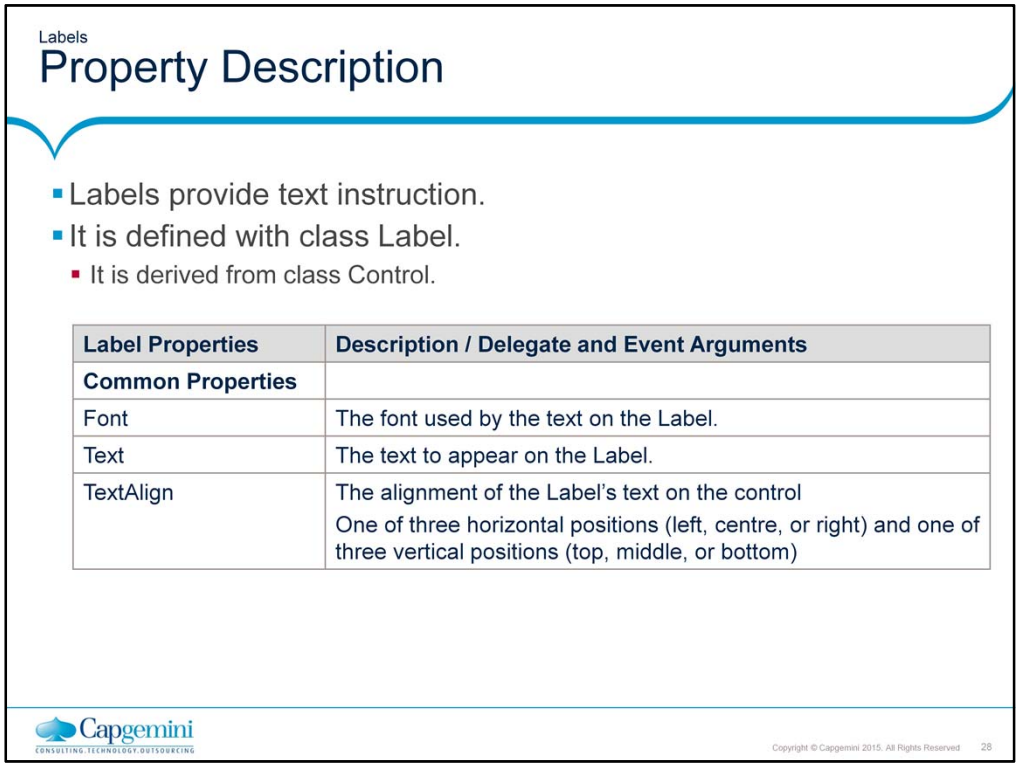
Some Common Control Properties

Explanation

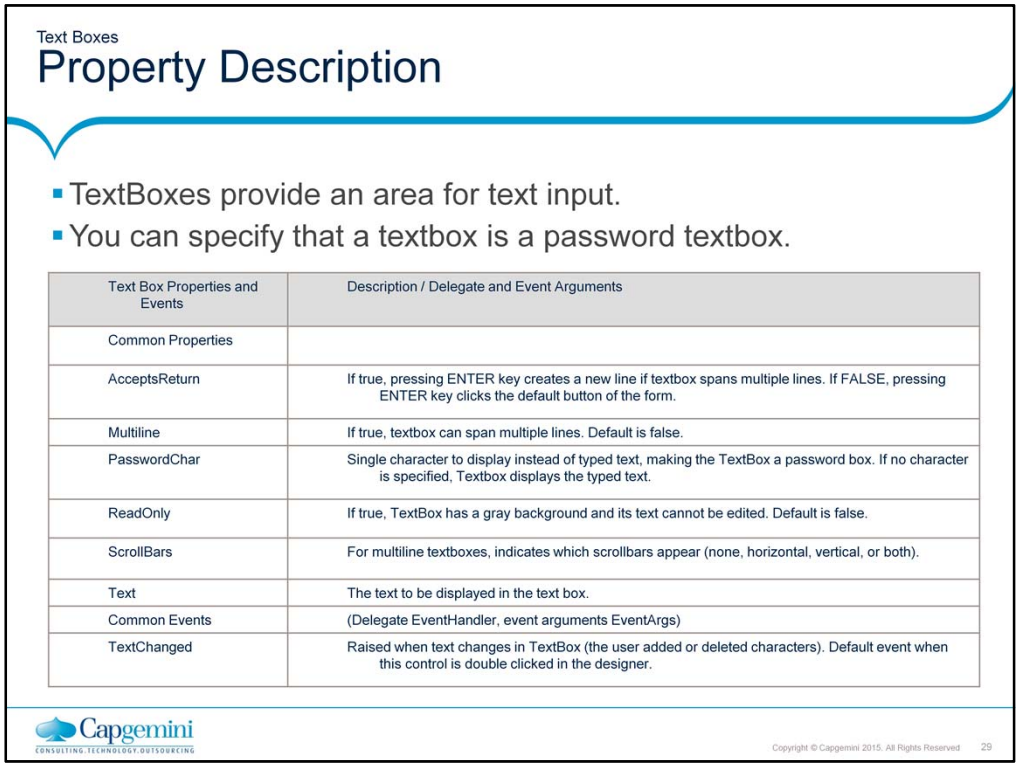
- Text property:
  - It specifies the text that appears on a control.
- TabIndex property:
  - It is the order in which controls are given focus.
  - It is automatically set by Visual Studio .NET.
- Enable property:
  - It indicates a control's accessibility.
- Visibility control:
  - It hides control from user.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 27



Labels:
Labels are generally used to provide descriptive text to the user. The text might be related to other controls or the current system state.
You usually see a label together with a text box. The label provides the user with a description of the type of data to be entered in the text box. The Label control is always read-only - the user cannot change the string value of the Text property. However, you can change the Text property in your code.



Text Boxes

Property Description

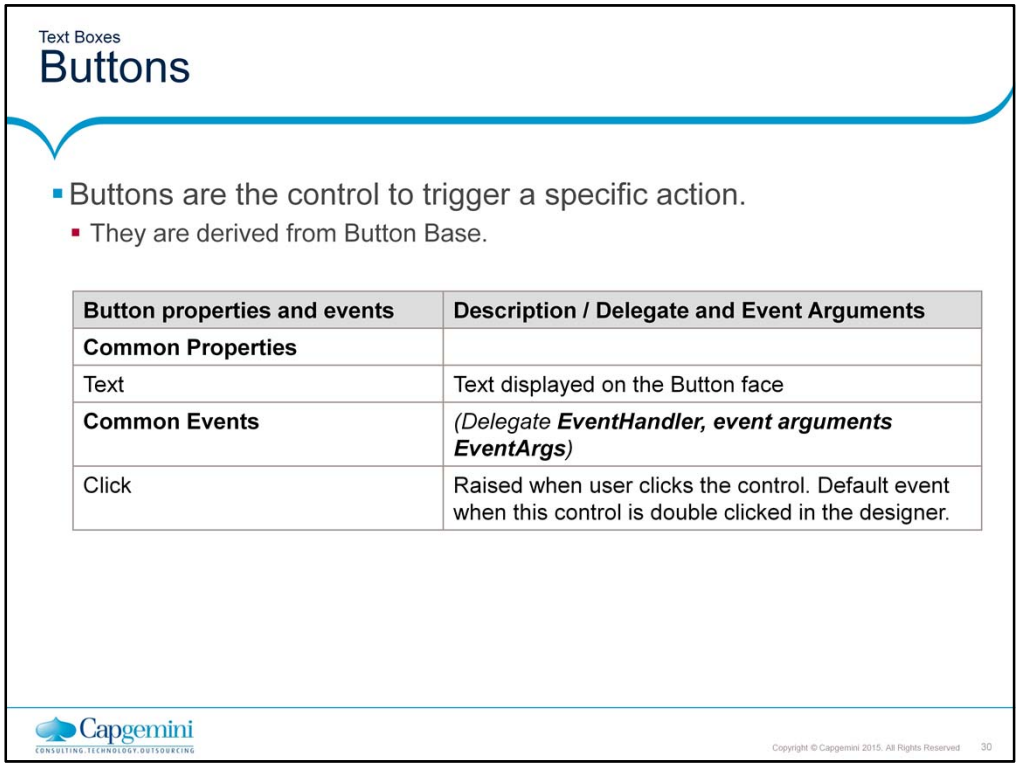
- TextBoxes provide an area for text input.
- You can specify that a textbox is a password textbox.

Text Box Properties and Events	Description / Delegate and Event Arguments
Common Properties	
AcceptsReturn	If true, pressing ENTER key creates a new line if textbox spans multiple lines. If FALSE, pressing ENTER key clicks the default button of the form.
Multiline	If true, textbox can span multiple lines. Default is false.
PasswordChar	Single character to display instead of typed text, making the TextBox a password box. If no character is specified, Textbox displays the typed text.
ReadOnly	If true, TextBox has a gray background and its text cannot be edited. Default is false.
ScrollBars	For multiline textboxes, indicates which scrollbars appear (none, horizontal, vertical, or both).
Text	The text to be displayed in the text box.
Common Events	(Delegate EventHandler, event arguments EventArgs)
TextChanged	Raised when text changes in TextBox (the user added or deleted characters). Default event when this control is double clicked in the designer.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 29

TextBoxes:
The TextBox control holds text or possibly multiple line of text.
A TextBox control can also be configured as read only and support scroll bars. The immediate base class of TextBox is TextBoxBase, which provides many common behaviors for the TextBox and RichTextBox Controls.



Buttons:
The role of Button type is to provide a simple vehicle for user input, typically in response to mouse click or key press.
The button class immediately derives from an abstract type named ButtonBase, which provides number of key behaviors for all Button related types (CheckBox, RadioButton, and Button).

Picture Box

Property Description

- Picture Box displays an image.
  - It reads image from file:
    - Image.FromFile() method (need to add System.IO name space)

PictureBox properties and events	Description / Delegate and Event Arguments
Common Properties	
Image	Image to display in PictureBox
SizeMode	Enumeration that controls image sizing and positioning Values **Normal** (default), **StretchImage**, **AutoSize,** and **CenterImage**. **Normal** puts image in top-left corner of **PictureBox**, and **CenterImage** puts image in middle (both cut off image if too large). **StretchImage** resizes image to fit in **PictureBox**. The **AutoSize** resizes PictureBox to hold image.
Common Events	(*Delegate **EventHandler**, event arguments **EventArgs***)
Click	Raised when user clicks the control Default event when this control is double-clicked in the designer

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 31

PictureBox:
The PictureBox control is used to display an image. The image can be a BMP, JPEG, GIF, PNG, metafile, or icon. The SizeMode property uses the PictureBoxSizeMode enumeration to determine how the image is sized and positioned in the control. The SizeMode property can be AutoSize, CenterImage, Normal, and StretchImage.
You can change the size of the display of the PictureBox by setting the ClientSize property. You load the PictureBox by first creating an Image-based object.
For example: To load a JPEG file into a PictureBox you will do the following:

```
Bitmap myJpeg = new Bitmap("mypic.jpg"); pictureBox1.Image
= (Image)myJpeg;
```

Notice that you will need to cast back to an Image type because that is what the Image property expects.

Group Boxes

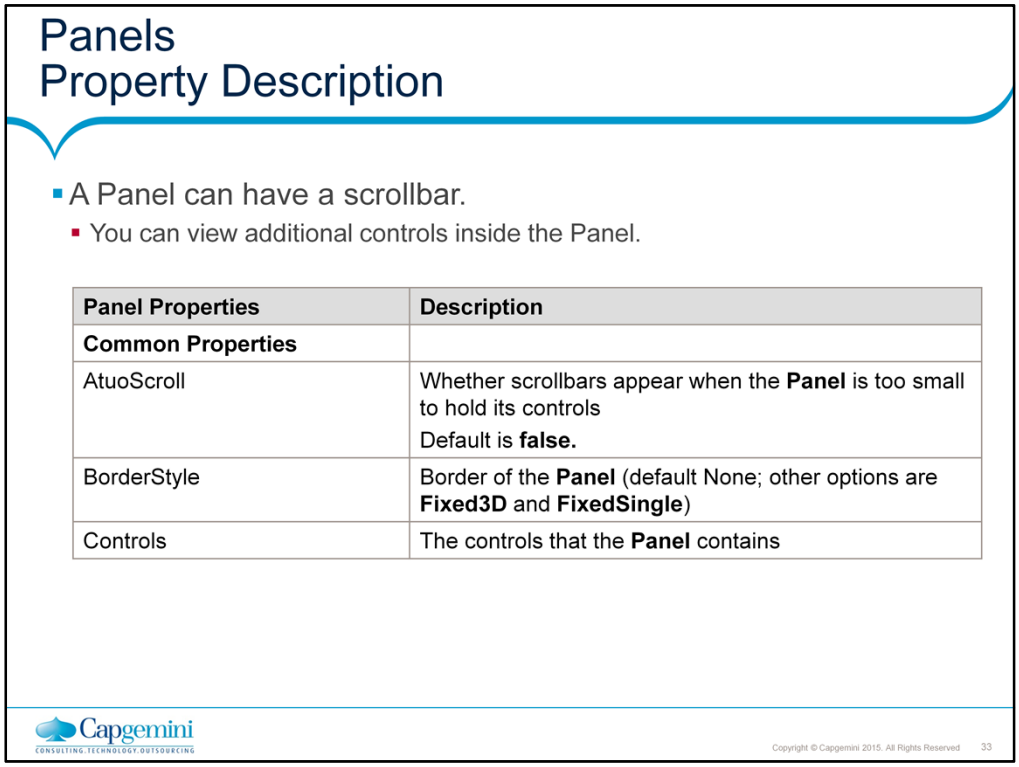
Property Description

- A Group Box can display a caption.
  - The Text property determines its caption.

GroupBox Properties	Description
Common Properties	
Controls	The controls that the GroupBox contains
Text	Text displayed on the top portion of the **GroupBox** (its caption)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 32

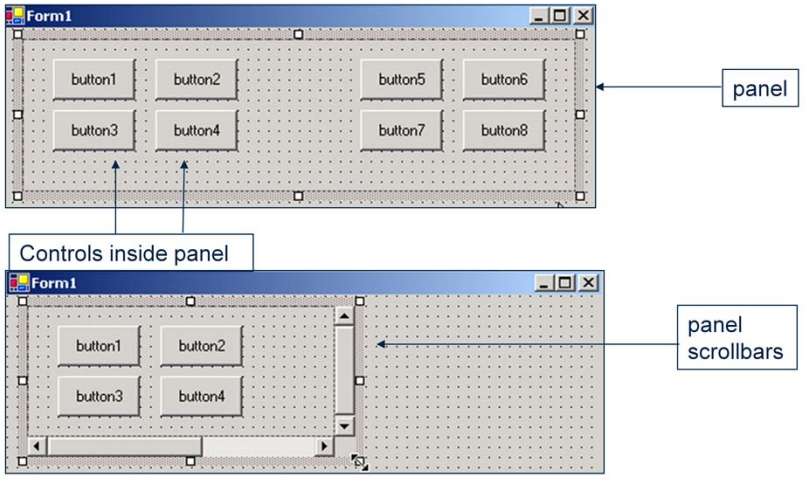


Panels:
Panel is simply a control that contains other controls. By grouping the controls together and placing them in a panel, it is a little easier to manage the controls. For example: You can disable all the controls in the panel by disabling the panel. Since the Panel control is derived from ScrollableControl, you also can get the advantage of the AutoScroll property. If you have too many controls to display in the available area, place them in a Panel and set AutoScroll to true. Now, you can scroll through all of the controls.
Panels do not show a border by default. However, by setting the BorderStyle property to something other than none, you can use the Panel to visually group related controls. This makes the user interface more user-friendly.

Group Boxes and Panels

Illustration

- Let us see an example of creating Panel with scrollbars.



Demo

- Demo with important controls

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 35

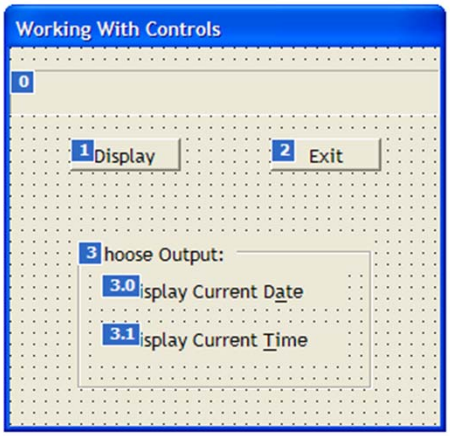
How to Set the Tab Order for Controls

- To set the tab order for controls
  - On the View menu, select Tab Order
  - Click a control to change its tab order

-- OR --

Set the TabIndex property

Set the TabStop property to True

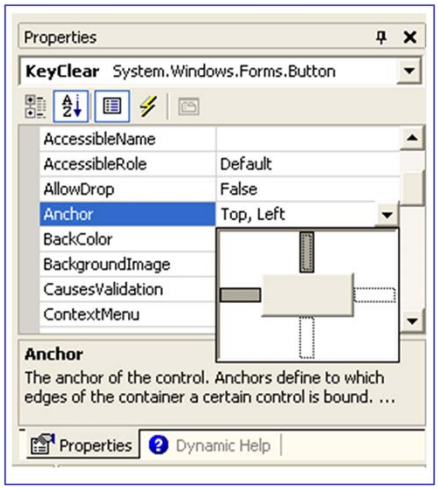


Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 36

How to Anchor a Control in Windows Forms

- Anchoring
  - Ensures that the edges of the control remain in the same position with respect to the parent container
- To anchor a control to the form
  - Set its Anchor property
  - Default value: Top, Left
  - Other Styles: Bottom, Right



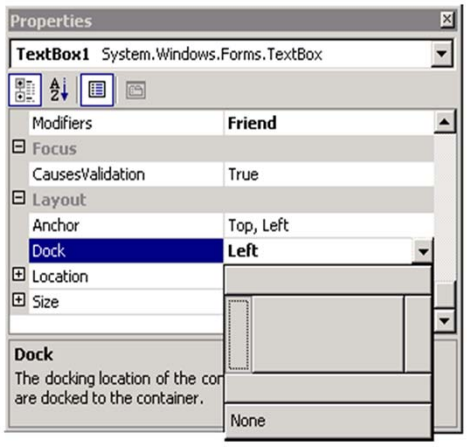
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 37

Set its Anchor Property
Default Value Is Top,Left
Other Styles are Bottom,Right.

How to Dock a Control in Windows Forms

- Docking
  - Enables you to glue the edges of a control to the edges of its parent control
- To dock a control
  - Set the Dock property

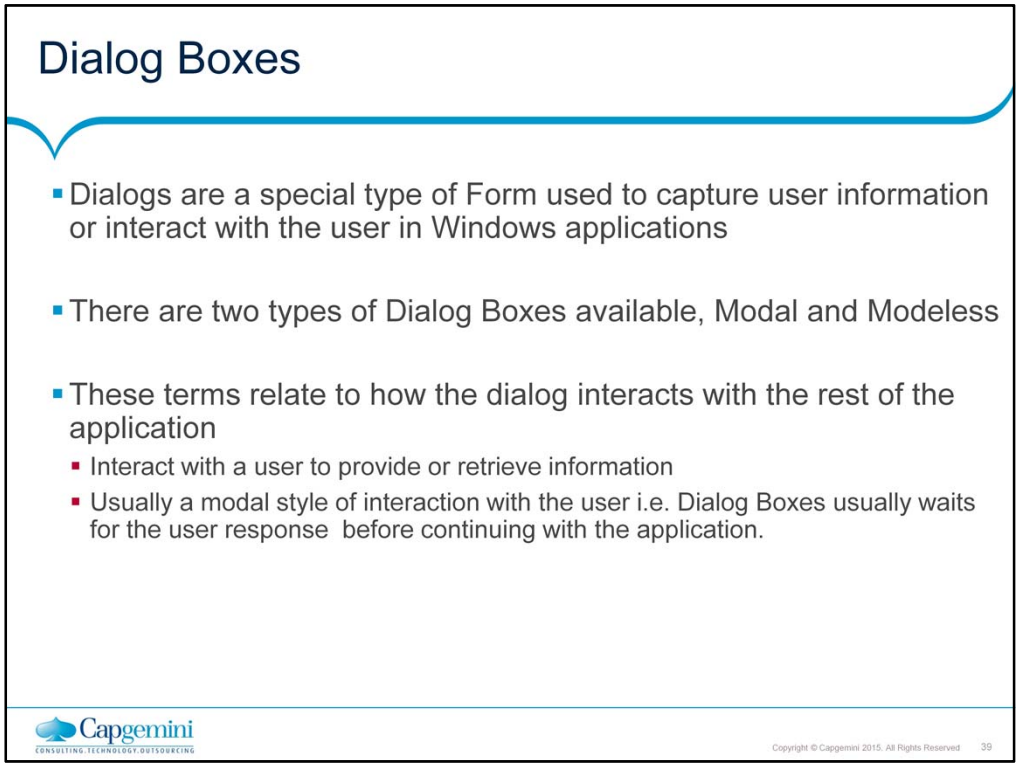


Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 38

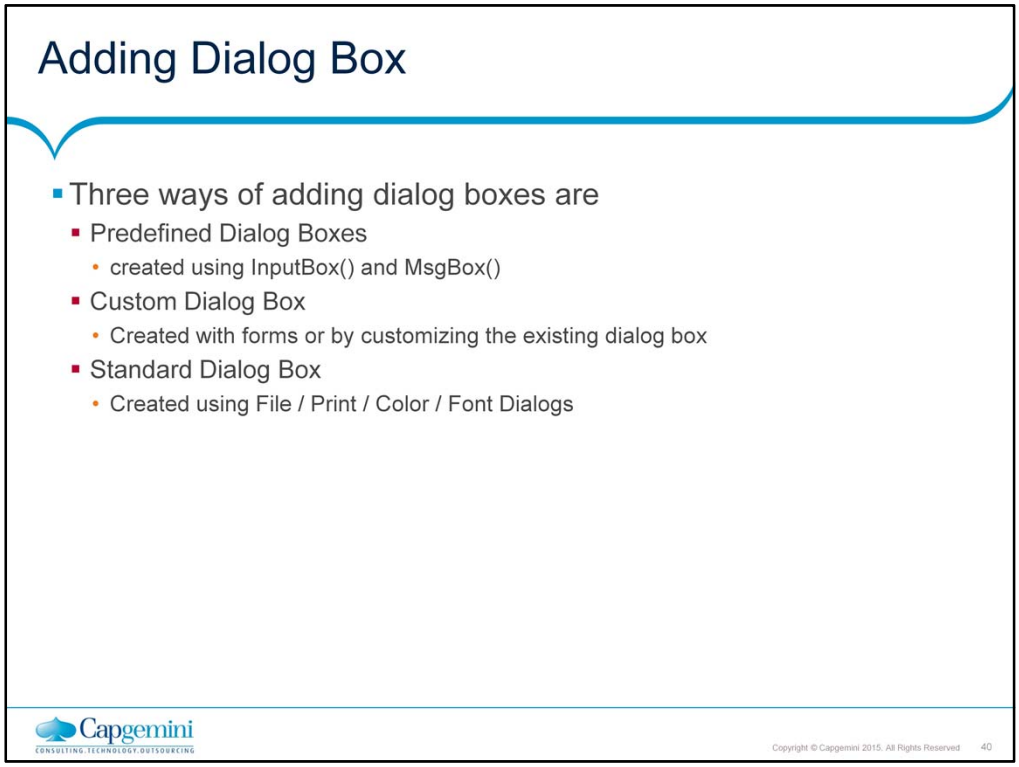
You can dock controls to edges of the form.For e.g Windows Explorer docks Treeview control on left side and listview on right side

Select the control you want to dock
In properties window docking contains buttons click the button you want to dock the control .Fill the contents of the control click the Fill button
None- No docking



Number of predefined dialog boxes that provide familiar functionality

There exists a set of predefined dialog boxes for capturing common information such as file locations, colors, and printer settings. A custom application will often use custom dialog boxes to facilitate the collection of data from endusers.
When we want to display the dialog box itself, there are two choices: modal or modeless. A modal dialog blocks the current thread and requires the user to respond to the dialog box before continuing with the application. A modeless dialog box is more like a standard window that allows the user to switch the focus back and forth between it and other windows.

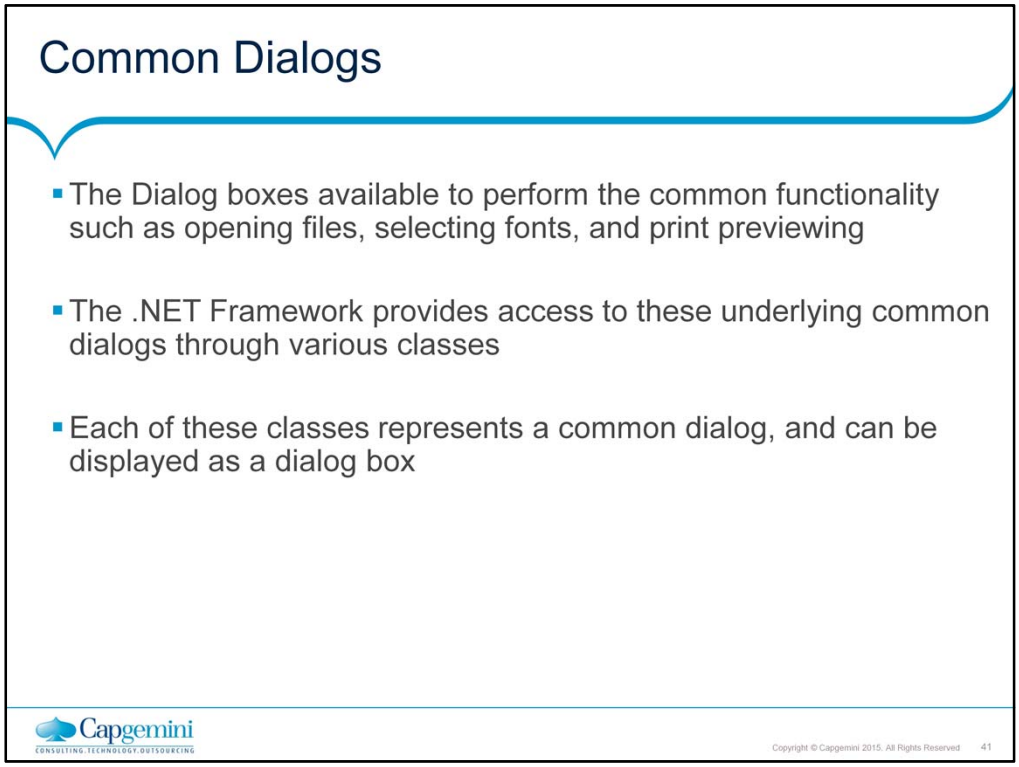


Using Msgbox Method

Prompts a message in the dialog box
Waits for the user to respond by clicking a button
Returns an integer indicating the button clicked

Using InputBox Method

Displays a modal dialog box that prompts user to enter some data
Contains Ok and Cancel buttons along with the message to the user



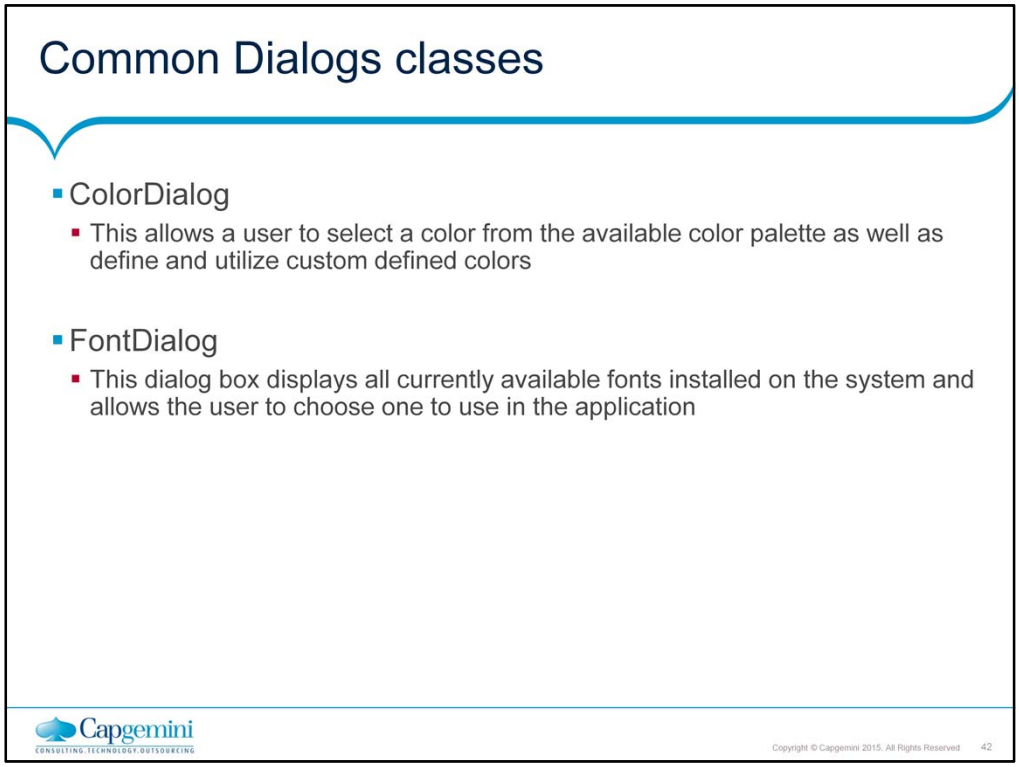
Common Dialog classes -
ColorDialog - This allows a user to select a color from the available color palette as well as define and utilize custom defined colors

FontDialog - This dialog box displays all currently available fonts installed on the system and allows the user to choose one to use in the application

OpenFileDialog - Allows a user to open a file using the standard open file dialog box
SaveFileDialog - This allows a user to select a file, directory or network location to save the application's data to

PrintDialog - This dialog box allows the user to set common page formatting and printing features via the standard print properties dialog box

PrintPreviewDialog - This displays a document as it will appear on the currently-selected printer with the current page settings



Example of Color Dialog Box

```
private void button1_Click(object sender, System.EventArgs e)
 {
   ColorDialog MyDialog = new ColorDialog();
   // Keeps the user from selecting a custom color.
   MyDialog.AllowFullOpen = false ;
   // Allows the user to get help. (The default is false.)
   MyDialog.ShowHelp = true ;
   // Sets the initial color select to the current text color.
   MyDialog.Color = textBox1.ForeColor ;

   // Update the text box color if the user clicks OK
   if (MyDialog.ShowDialog() == DialogResult.OK)
      textBox1.ForeColor =  MyDialog.Color;
 }
```

Example of FontDialog

```
private void button1_Click(object sender, System.EventArgs e)
 {
   fontDialog1.ShowColor = true;

   fontDialog1.Font = textBox1.Font;
   fontDialog1.Color = textBox1.ForeColor;

   if(fontDialog1.ShowDialog() != DialogResult.Cancel )
   {
     textBox1.Font = fontDialog1.Font ;
     textBox1.ForeColor = fontDialog1.Color;
   }
 }
```

Demo

- Demo on Dialog Boxes

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 43

MDI Applications

Explanation

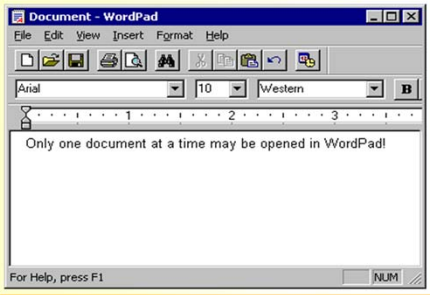
- Multiple Document Interface (MDI) applications have a single, primary window (the parent window) that contains a set of windows (the child windows) within its client region.
  - Each child window is a form that is constrained to appear only within the parent.
  - Children typically share the menu bar, tool bar, and other parts of the parent's interface.
  - Secondary windows like dialog boxes are not constrained to the parent window's client region.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 44

SDI vs. MDI Applications

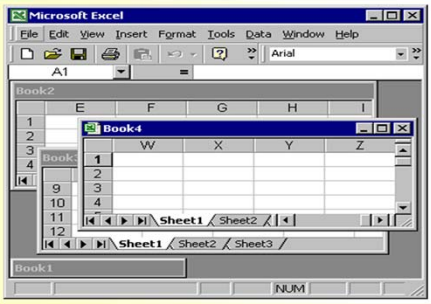
SDI



Only one document is visible

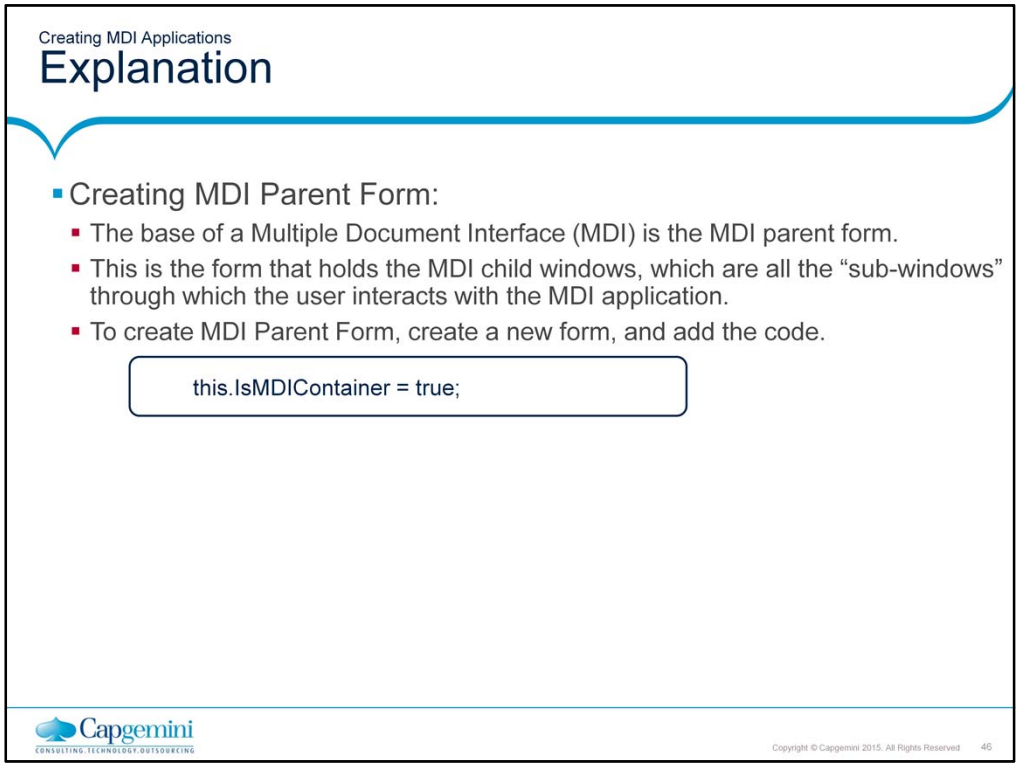
You must close one document before you open another

MDI



Displays multiple documents at the same time

Each document is displayed in its own window

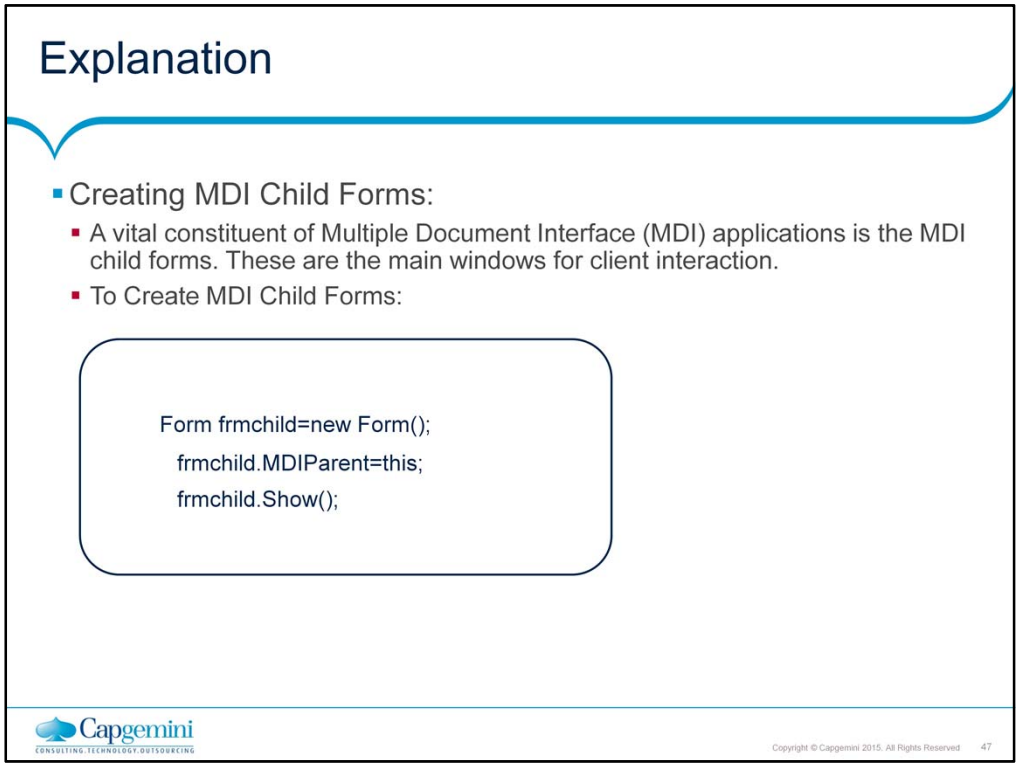


Creating MDI Applications:

You can create an MDI application by using the following steps:

Create a Form (MainForm) that represents the MDI parent window.
Set its IsMdiContainer property to true. The following code demonstrates how to set this property.

```
this.IsMdiContainer = true;
```

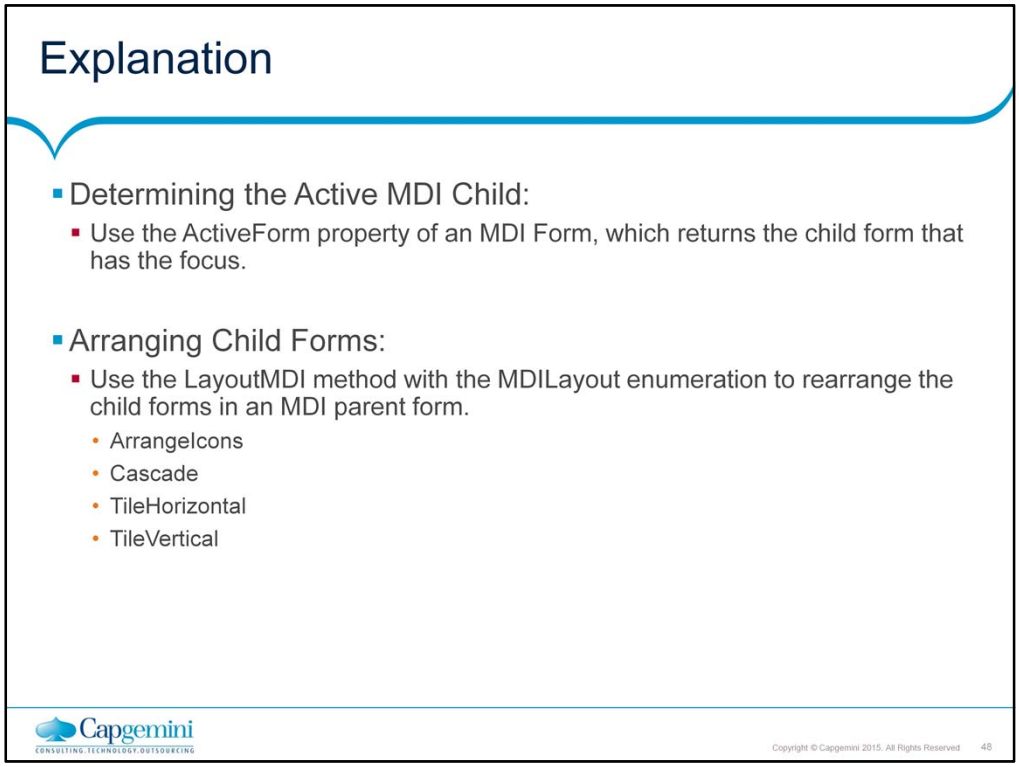


Creating MDI Applications – Creating MDI Child Forms:
Create child forms and set the MdiParent property of each form to reference the parent form. The following code demonstrates setting the MDI parent for an instance of a child form.

```
Form frmchild=new Form();
   frmchild.MDIParent=this;
```

If you have different types of data to display, you can have multiple types of child forms. To display a child form, create an instance of the child form and call its Show method.

```
frmchild.Show();
```



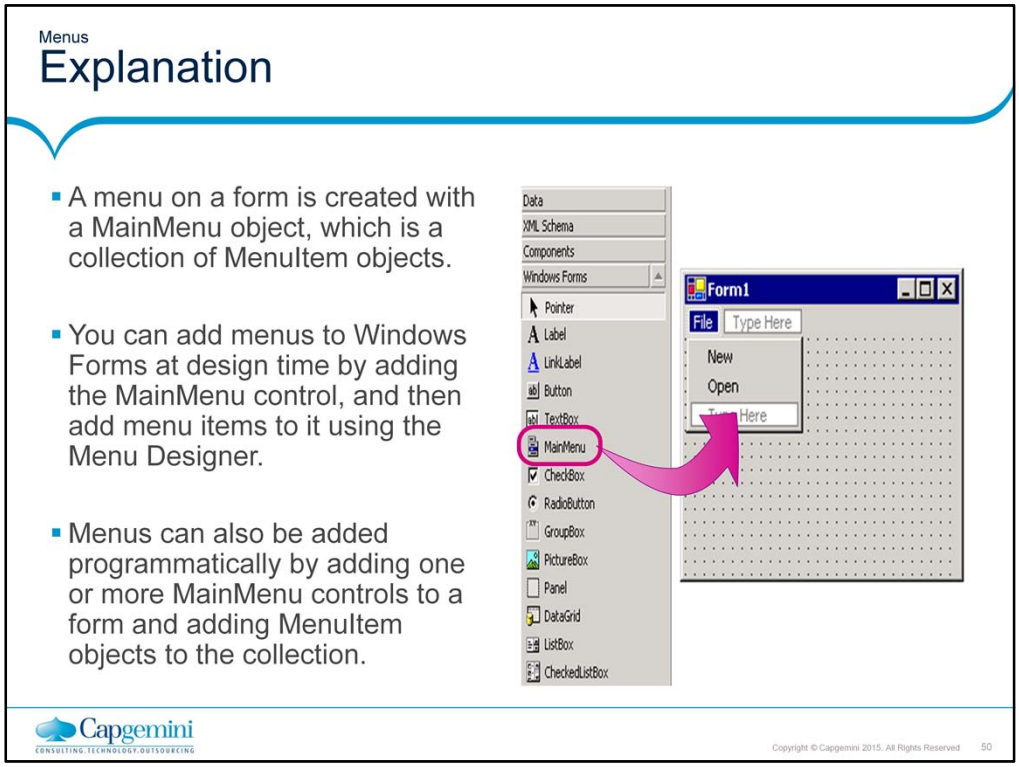
Creating MDI Applications – Creating MDI Child Forms:
Determining the Active MDI Child:
In a number of circumstance, we desire to give a command that operates on the control with the focus on the current active child form.
For the reason that an application can have many instances of the same child form, the process wants to be acquainted with which form to use. To specify this, use the ActiveForm property of an MDI Form, which returns the child form that has the focus.
In any case, one MDI child form must be loaded and visible when you access the ActiveForm property, or an error is returned.
Arranging Child Forms:
Frequently, applications will have menu commands for actions such as Tile, Cascade, and Arrange, which concern to the open MDI child forms. One can use the LayoutMDI method with the MDILayout enumeration to rearrange the child forms in an MDI parent form.
The MDILayout enumeration can be set to four different values, which will display child forms as cascading, either horizontally or vertically. Often, these methods are used as the event handlers called by a menu item's Click event. In this way, a menu item with the text “Cascade Windows” can have the desired effect on the MDI child windows.

Demo

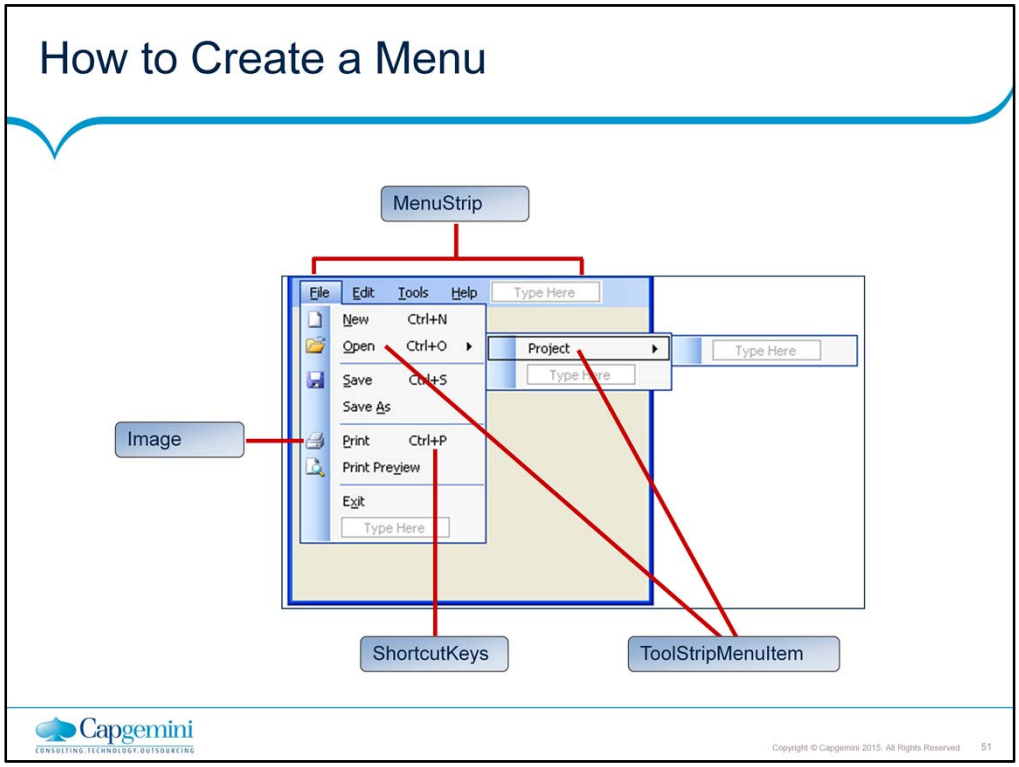
- Demo on creating MDI application

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

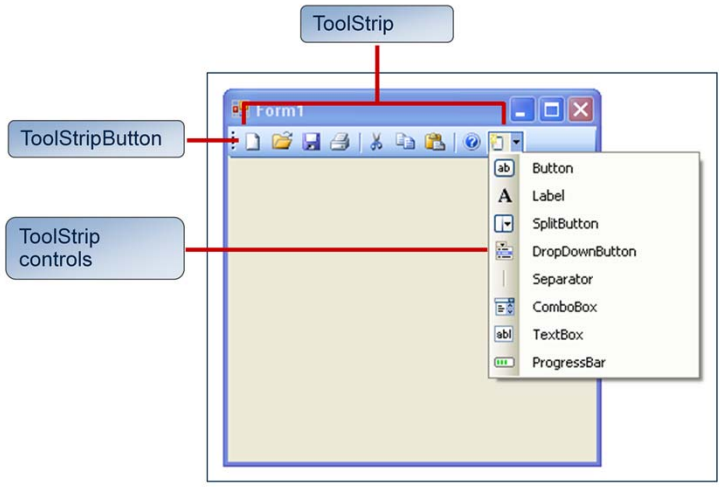
 49



Menus:
The main window of nearly all Windows applications includes a menu across the top. This is called the main menu. The main menu typically contains top-level categories, such as File, Edit, and Tools. From the main menu descend drop-down menus, which contain the actual selections associated with the categories. When a menu item is selected, a message is generated. Therefore, to process a menu selection, your program will assign an event handler to each menu item.
Prior to C# 2.0, there was only one way to create a main menu: that is by using the classes MainMenu and MenuItem, both of which inherit the Menu class. These classes create the traditional-style menus that have been used in Windows applications for years. C# 2.0 still supports these classes and traditional-style menus. However, it adds a second way to add menus to a window by using a new set of classes based on ToolStrip, such as MenuStrip and ToolStripMenuItem. Menus based on ToolStrip have many additional capabilities and give a modern look and feel.


How to Create a Menu
MenuStrip
File
Edit
Tools
Help
Type Here
New Ctrl+N
Open Ctrl+O
Save Ctrl+S
Save As
Print Ctrl+P
Print Preview
Exit
Type Here
Project
Type Here
Type Here
Image
ShortcutKeys
ToolStripMenuItem
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
51

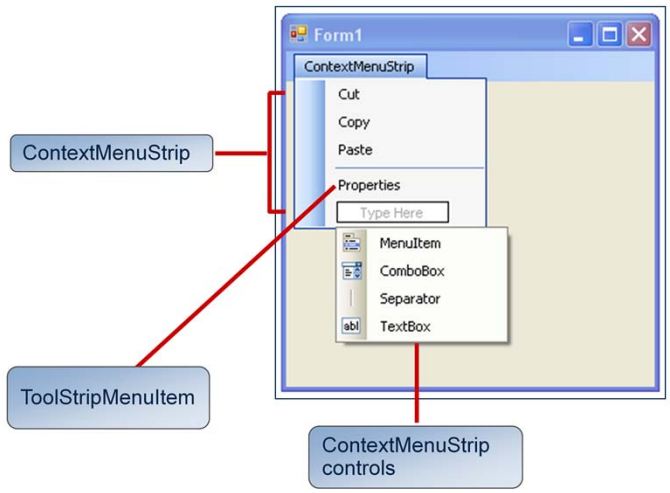
How to Create a Tool Strip



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 52

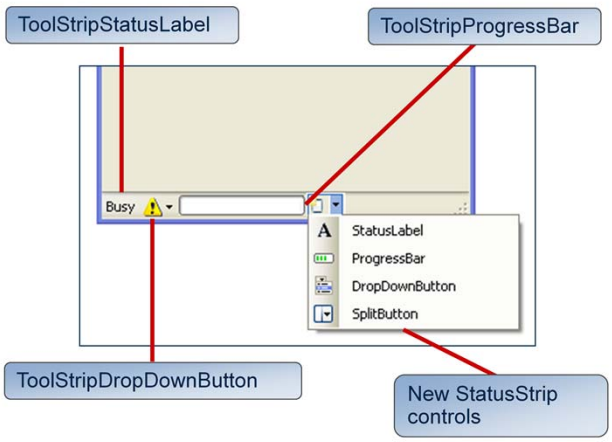
How to Create a Context Menu



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 53

How to Create a Status Bar



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 54

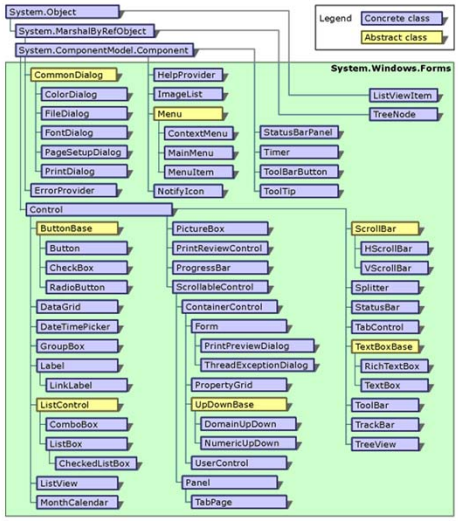
Demo

- Demo on creating Menus,ToolStrip…

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 55

Windows Forms Hierarchy



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 56

Summary

- In this lesson, you have learnt:
  - Differences between Console Application and Windows Application
  - Delegate based Event Handling Model in .NET
  - Different Controls in .NET and their properties
  - What is Anchoring and Docking?
  - How do you create MDI Application in C# ?

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 57

Summary

- In this lesson, you have learnt (contd.):
  - How do you create Menus in C# ?

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 58

Review Question

- Question 1: What are the challenges for a Windows application?
- Question 2: Can I have two event handlers for one event? How?
- Question 3: Can two events point to the same event Handler? How?
- Question 4: What is Anchoring and Docking?
- Question 5: What are the different types of Menus you can create using C#?

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 59

Windows Forms

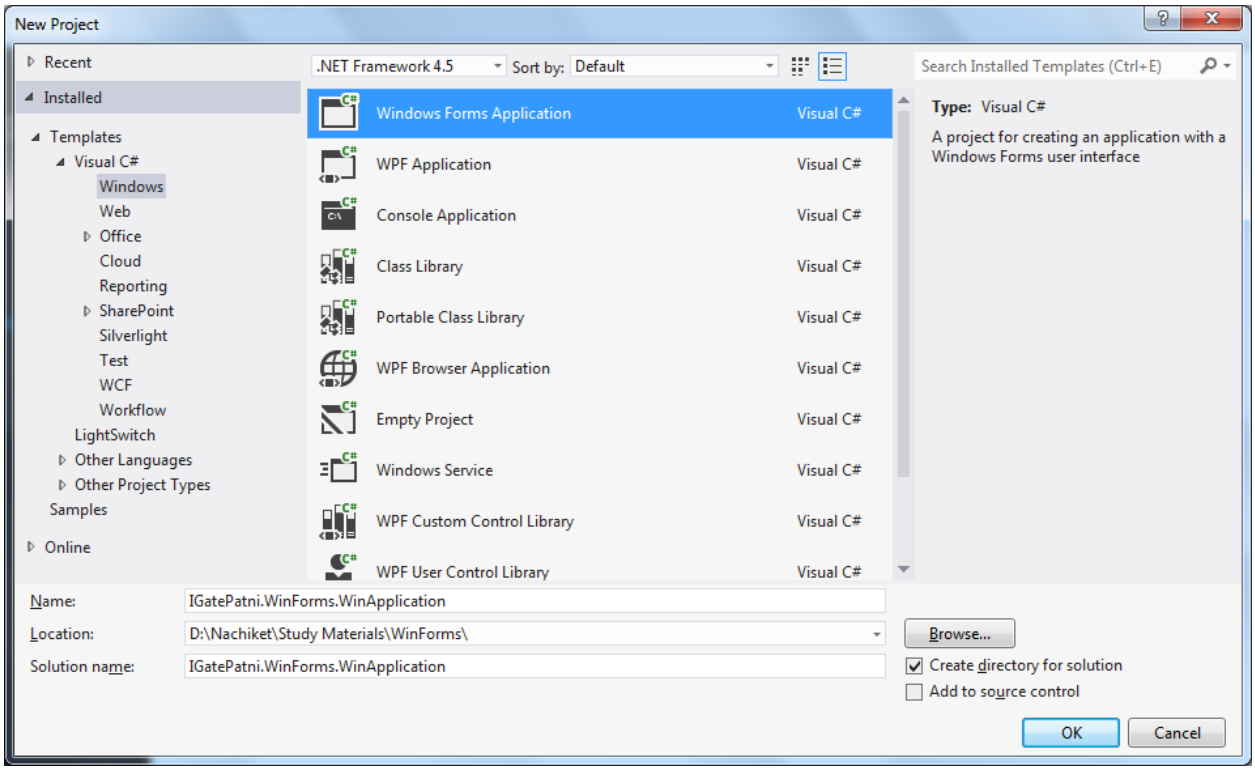
Lab Book

Table of Contents

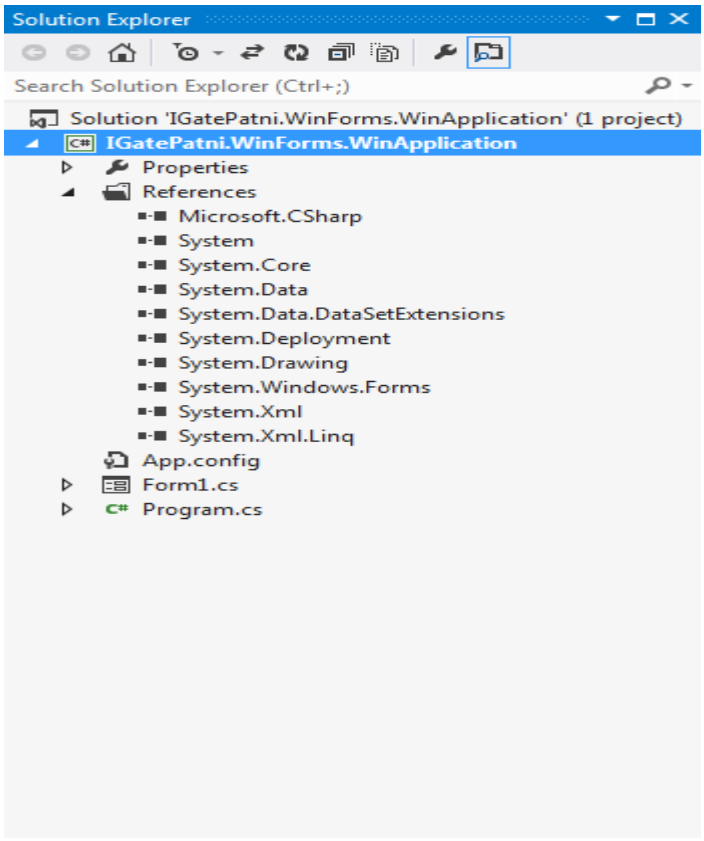
Lab 1.Getting familiar with Windows Application development environment

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



2. Take a look at solution explorer:



- 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.

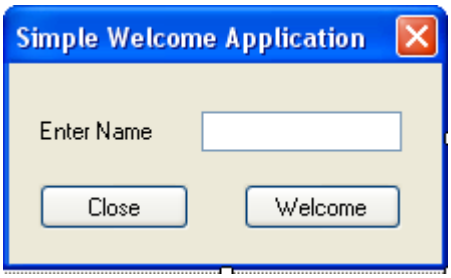
  - Button1: To launch Solution Explorer.
  - Button2: To launch Properties Window
  - Button3: To launch Object Browser to take a look at various assemblies and types in it as well as type metadata.
  - Button4: To launch Toolbox to drag controls.

3. Open Form1 designer and drag 3 controls:

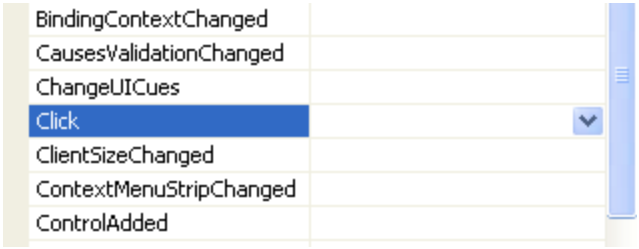
a. Label, TextBox and 2 Buttons.

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



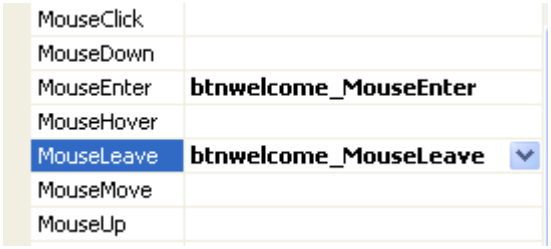
4. To add a click event handler for button; double click welcome button OR select welcome button and go to properties window click the event's small button in property window  to get events list
   a. Find click event and double click.



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.



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.

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.

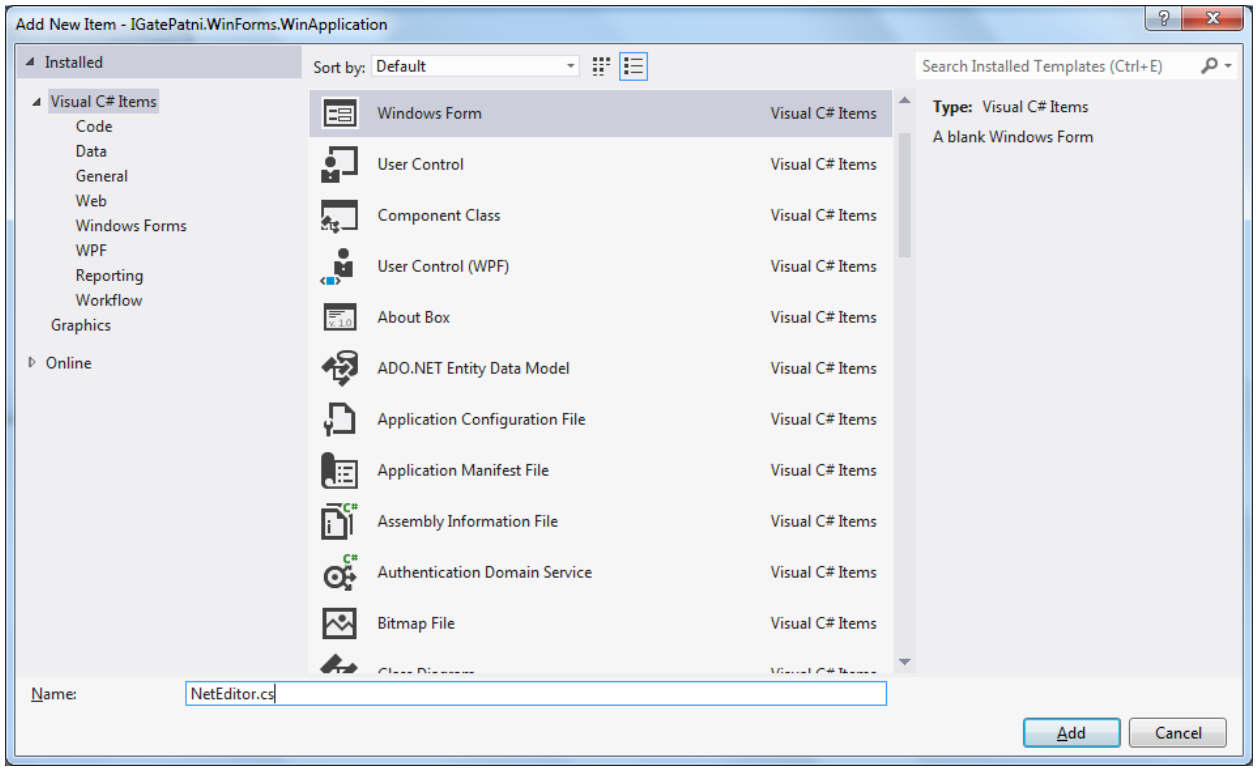
Assignment

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

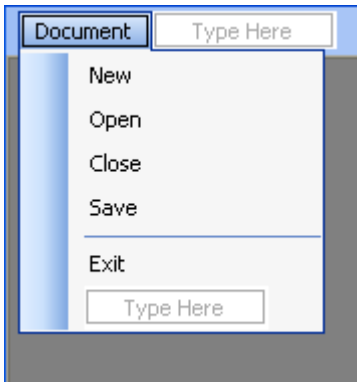
Lab 2. Understand using MDI application and Menus

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



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

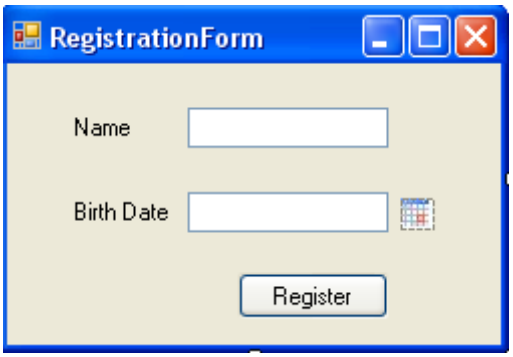
To Do:

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");
}
```

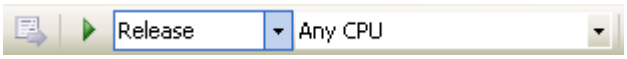
10. Make the RegistrationForm as startup and execute.

Lab 4. Deployment using ClickOnce

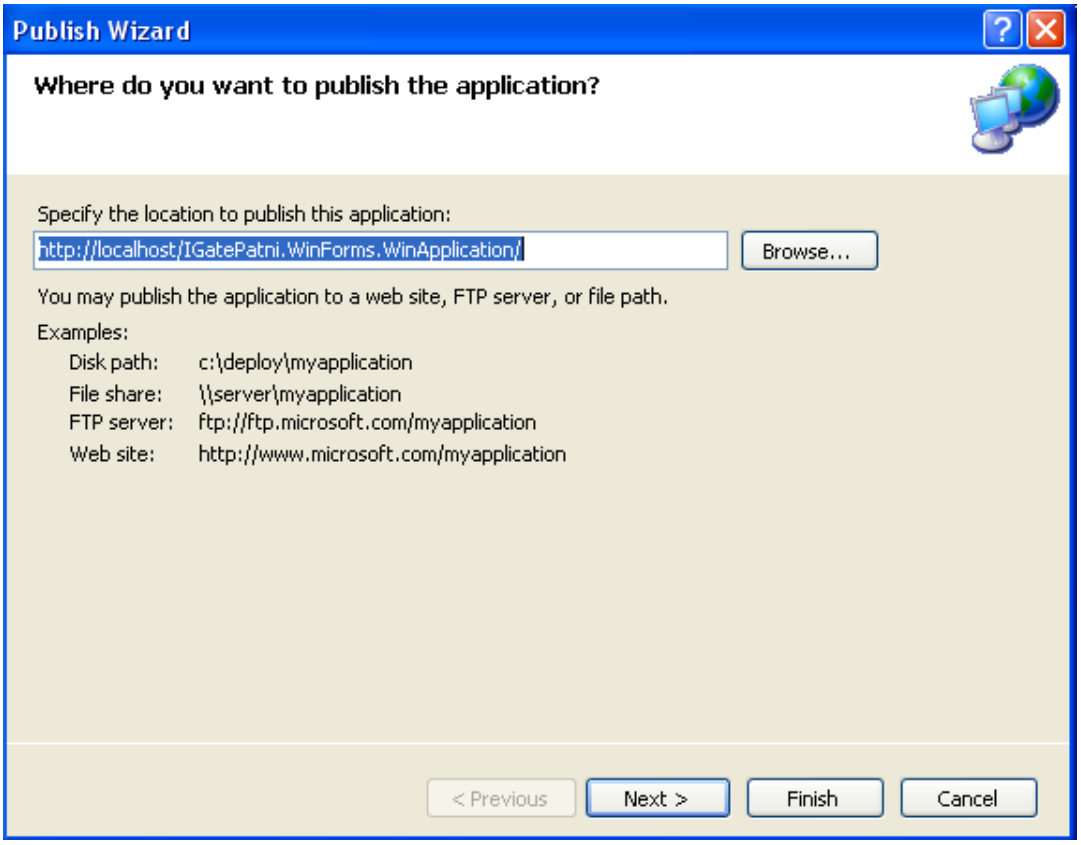
Description	We will be deploying our current windows application using click once deployment feature.
Objective	To Learn: • How to deploy a windows application using ClickOnce • Check the auto-updates feature of ClickOnce
Time	60 Mins

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.

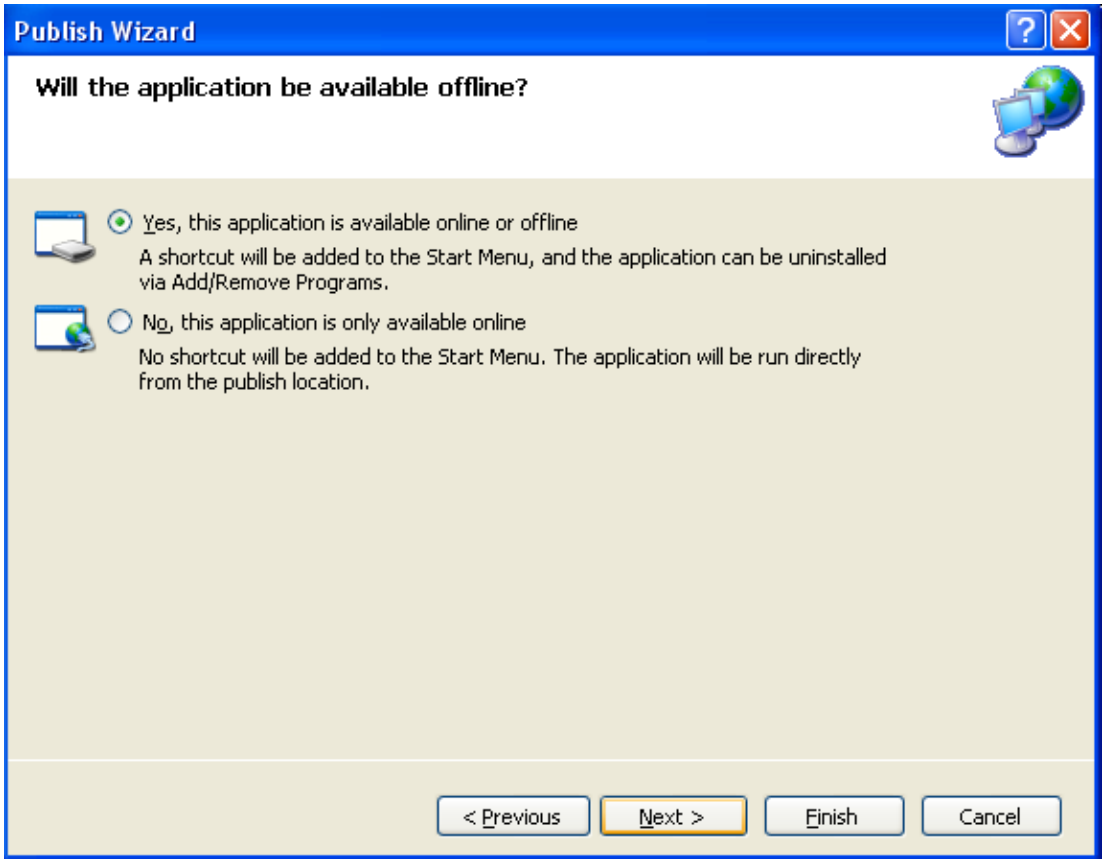


3. Build the project.
4. Publish the application Setup on the local IIS web server
    a. Build => Publish OR Right Click the project in solution explorer and select Publish
5. A publish popup dialog will come up:



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.

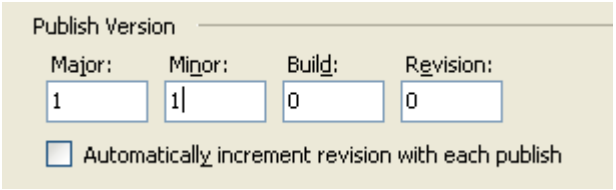


8. Click Next.
9. Click Finish on last screen.
10. The setup will be published and browser will show you the URL.

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

Lets check auto update feature

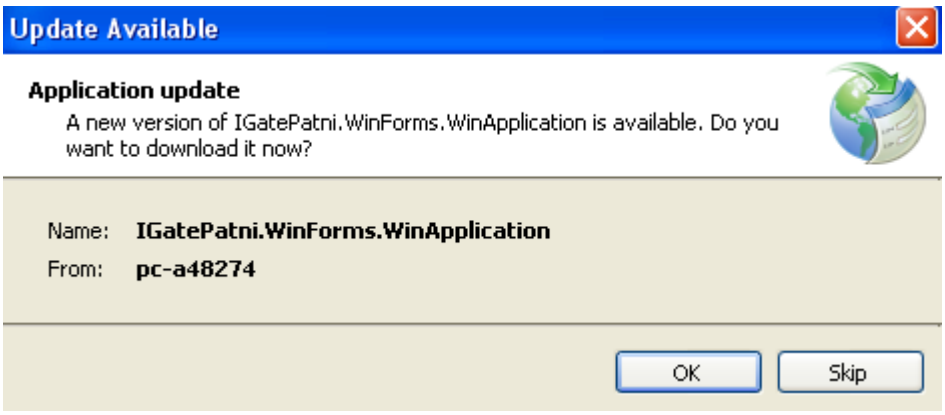
13. Open Visual Studio and change the BackColor of TextBox on NetEditorChild Form to Yellow. Rebuild the project.
14. Go to Project => Properties OR Right Click Project in solution explorer and select properties, Move to Publish TAB.
15. Make the build version as 1.1.0.0, click save



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

Do not click Install

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



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.

