



.Net Programming

Unit 3 - Application Development

By – Dr. Jay B. Teraiya

Agenda of Unit 3

- Application Development using Windows Form
- Label Control
- TextBox Control
- Button Control
- RadioButton Control
- CheckBox Control
- LinkLable Control
- RichTextBox Control
- ListBox Control
- CheckedListBox Control
- ListView Control
- Menus
- Validation Control
- SDI and MDI Application
- .Net Server Controls
- Master Page
- Themes, Skins and CSS
- Configuration Overview
- .Net State Management
- Caching in ASP.Net
- Navigation Control in ASP.NET

Application Development using Windows Form

- Windows Forms provides a graphical user interface (GUI) for building windows client applications.
- This GUI represents a part of Microsoft.NET Framework.
- You can create Windows Forms Applications in any language that is supported by CLR.
- Prior to Windows Forms, Programmers had to perform the difficult and inconvenient task of manually writing thousands of lines of code for designing an application.
- Windows Forms provides an easy and ideal way for creating applications.

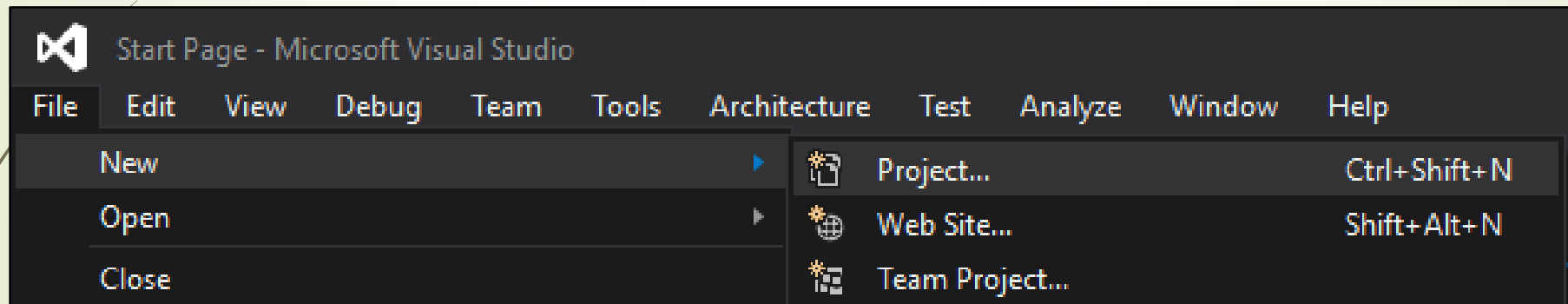
Application Development using Windows Form

- It acts as container that allow you to add different types of controls, such as Button, Label and so many others.
- You can also use the Windows Forms for creating message boxes and dialog boxes, displaying messages, and accepting user data.
- It allows you to design multiple forms to perform various tasks by using different controls with their properties and methods.
- Namespace : `System.Windows.Forms`.
- Examples of a Windows Forms Applications
 - Notepad
 - Calculator

Application Development using Windows Form

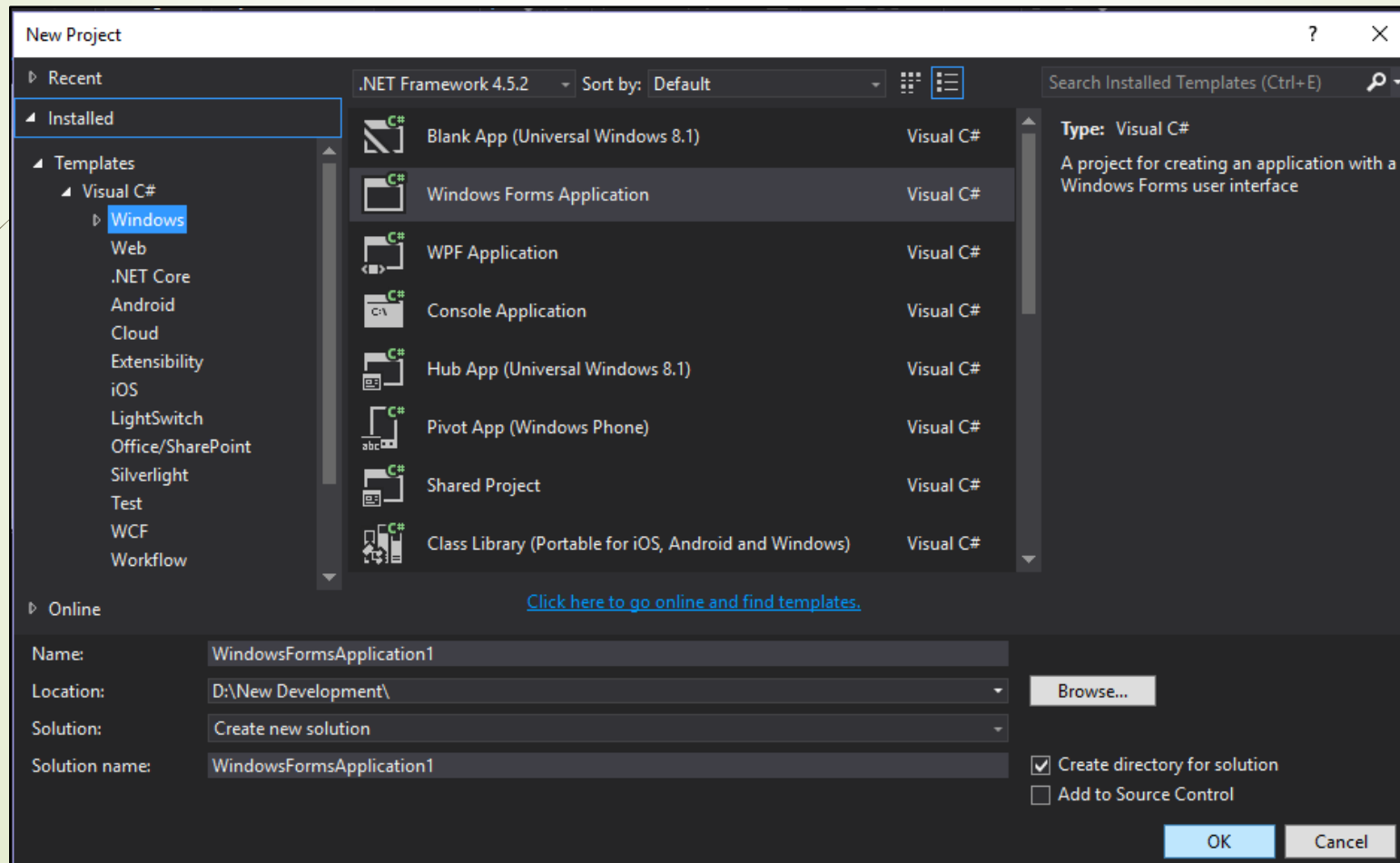
➤ Step : 1

- Open Visual Studio, Go to File menu → New → Project.
- After Clicking on Project, new project window will appear.

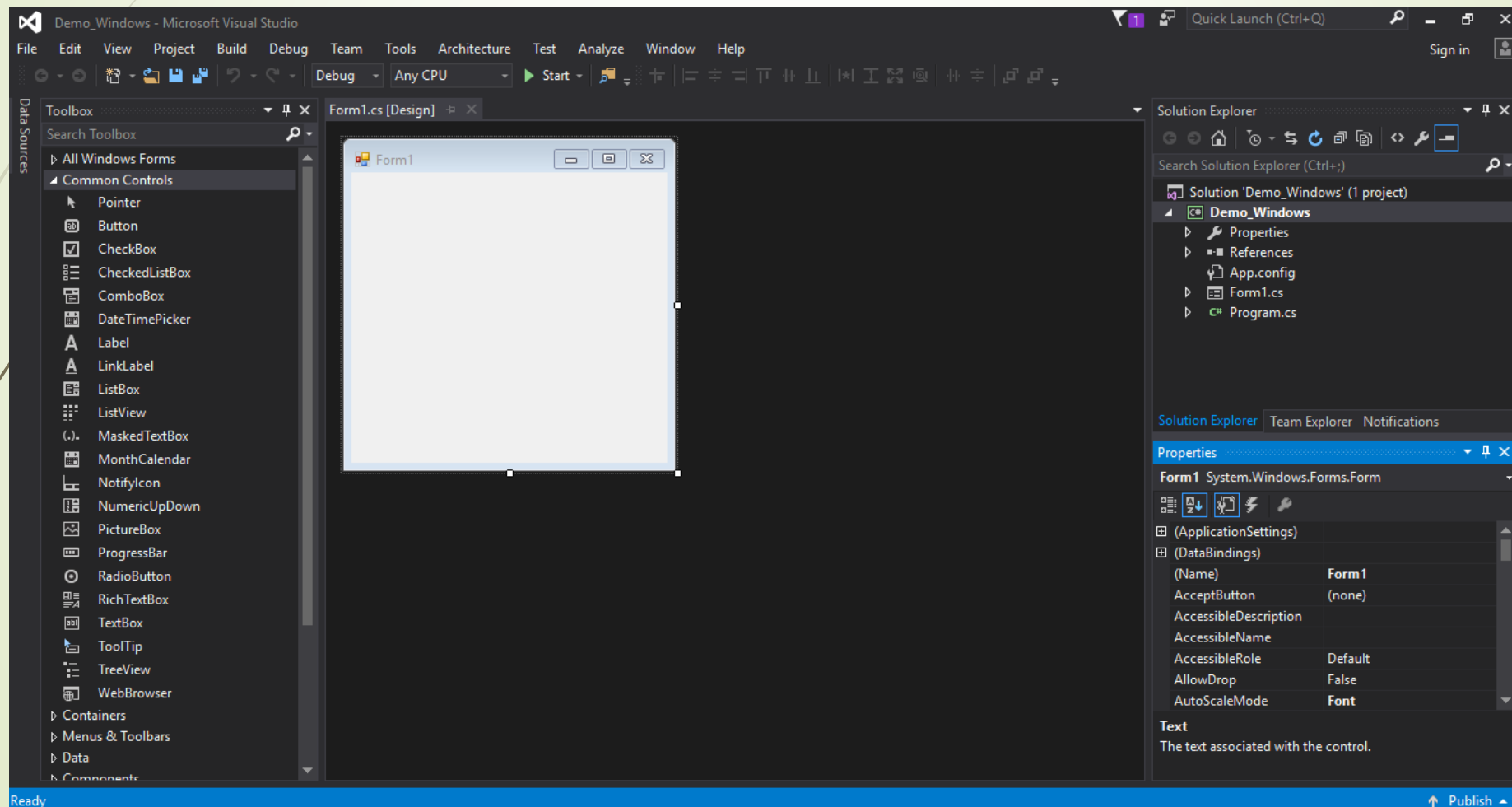


Application Development using Windows Form

- Select Language, Project Type, Set Location & Give Project Name.



Application Development using Windows Form



Application Development using Windows Form

Windows Forms Properties.

Property	Description
Name	This is the actual name of the form.
Text	The text, which will appear at the title bar of the form.
Width	This is the width of the form in pixel.
Height	This is the height of the form in pixel.
Font	This property specify font type, style, size.
BackColor	Sets the form background color.
AcceptButton	The button that's automatically activated when you press Enter, no matter which control has the focus at the time. Usually the OK button on a form is set as AcceptButton for a form.
CancelButton	The button that's automatically activated when you hit the Esc key.
WindowState	Form remains in default size as you show in design, but you can change its state by minimized, maximized & normal.
StartPosition	This property enables you to set the starting position of the form when it is displayed at run time.

Application Development using Windows Form

➤ Windows Forms Events.

Events	Description
Activated	Occurs when the form is activated in code or by the user.
Click	Occurs when the form is clicked.
Closed	Occurs before the form is closed.
Closing	Occurs when the form is closing.
Load	Occurs before a form is displayed for the first time.
MouseHover	Occurs when the mouse pointer rests on the form.
DoubleClick	Occurs when the form control is double-clicked.
KeyDown	Occurs when a key is pressed while the form has focus.
Shown	Occurs whenever the form is first displayed.

Application Development using Windows Form

Windows Forms Basic Controls

- A Windows forms application runs on the desktop computer.
- Normally, GUI is a combination of controls which can be used by user.
- Some of the controls are :

Basic Controls
Label
TextBox
Button
RadioButton
CheckBox
PictureBox

List Controls
ListBox
ComboBox
CheckedListBox

Data Controls
DataGridView

Application Development using Windows Form

Common Properties of Controls

Property	Description
Name	The name of the control. This name can be used to reference the control in code.
Text	Gets or sets the text associated with control.
Visible	Specifies whether or not the control is visible at runtime.
Width	The width of the control in pixel.
Height	The Height of the control in pixel.
BackColor	The background color of a control.
ForeColor	The foreground color of the control.
Font	Gets or sets the font of the text displayed by the control.

Application Development using Windows Form

Common Events of Controls

Event	Description
Click	Occurs when a control is clicked. In some cases, this event will also occur when a user presses Enter.
DoubleClick	Occurs when a control is double-clicked. Handling the Click event on some controls, such as the Button control will mean that the DoubleClick event can never be called.
DragDrop	Occurs when a drag-and-drop operation is completed, in other words, when an object has been dragged over the control, and the user releases the mouse button.
KeyDown	Occurs when a key becomes pressed while the control has focus. This event always occurs before KeyPress and KeyUp.
KeyPress	Occurs when a key becomes pressed, while a control has focus. This event always occurs after KeyDown and before KeyUp.
GotFocus	Occurs when a control receives focus. Do not use this event to perform validation of controls. Use Validating and Validated instead.
LostFocus	Occurs when a control loses focus. Do not use this event to perform validation of controls. Use Validating and Validated instead.

Label Control

This Label

- Label Control is generally used to display the text that is not supposed to be changed by user.
- It is used as instruction to user that which control is used for what purpose.
- Namespace : System.Windows.Forms.Label

➤ Properties of Label Control

Property	Description
TextAlign	Gets or sets how text is aligned in a Label control.
Visible	Gets or sets a value indicating whether the control and all its child controls are displayed.
AutoSize	Gets or sets a value specifying if the label control should be automatically resized to display all its contents.

Label Control

➤ Methods of Label Control

Method	Description
Focus()	Sets input focus to the control.
ResetText()	Resets the Text property to its default value.
ResetFont()	Resets the Font property to its default value.
ResetForeColor()	Resets the ForeColor property to its default value.
Show()	Displays the control to the user.

➤ Events of Label Control

Event	Description
TextChanged	Occurs when the Text property value changes.
Click	Occurs when user clicks the label.
Leave	Occurs when the input focus leaves the label.
LostFocus	Occurs when the control loses focus.

TextBox Control

This Is Text Box

- The TextBox control is a windows forms control that lets users enter text on a windows form at runtime.
- By default, a TextBox control accepts only a single line of text.
- Namespace : System.Windows.Forms.TextBox
- **Properties of TextBox Control**

Property	Description
TextLength	Gets the length of text in the TextBox control.
PasswordChar	Gets or sets the character used to mask characters of a password in a single-line TextBox control.
Multiline	Gets or sets a value indicating whether this is a multiline TextBox control.
ReadOnly	Gets or sets a value indicating whether text in the text box is read-only.
WordWrap	Indicates whether a multiline text box control automatically wraps words to the beginning of the next line when necessary.

TextBox Control

➤ Methods of TextBox Control

Method	Description
Clear()	Clears all text from the text box control.
Copy()	Copies the current selection in the text box to the Clipboard.
Cut()	Moves the current selection in the text box to the Clipboard.
DeselectAll()	Specifies that the value of the SelectionLength property is zero so that no characters are selected in the control.
SelectAll()	Selects all text in the text box.
Focus()	Sets input focus to the control.
ResetText()	Resets the Text property to its default value.

➤ Events of TextBox Control

Event	Description
TextChanged	Occurs when the text in the textbox changes.
TextAlignChanged	Occurs when the TextAlign property value changes.

Button Control

button1

- A Button Control accepts clicks.
- In Windows Forms we use a Button control, that accepts click and performs other actions on the user interface.
- Example
- When you press a close button, the form will be close.
- Namespace : `System.Windows.Forms.Button`
- **Properties of Button Control**

Property	Description
BackColor	Gets or sets the background color of the button control.
Image	Gets or sets the image that is displayed on a button control.
FlatStyle	Gets or sets the flat style appearance of the button control.

Button Control

button1

➤ Methods of Button Control

Method	Description
Focus()	Sets input focus to the control.
ResetBackColor()	Resets the BackColor property to its default value.
ResetFont()	Resets the Font property to its default value.
ResetText()	Resets the Text property to its default value.
Select()	Activates the control.
Show()	Displays the control to the user.

➤ Events of Button Control

Event	Description
Click	Occurs when user clicks the Button.
Validated	Occurs when the button control is finished validating.
GotFocus	Occurs when the button control receives focus.
TextChanged	Occurs when the Text property value changes.
FontChange	Occurs when font is changed.

RadioButton Control

☐ Male ☐ Female

- RadioButton Control enables the user to select a single option from a group of choices when paired with other RadioButton controls.
- When a user clicks on a radio button, it becomes checked, and all other radio buttons with same group become unchecked.
- The RadioButton control can display text, an Image, or both.
- Namespace : System.Windows.Forms.RadioButton
- Properties of RadioButton Control

Property	Description
Checked	Gets or sets a value indicating whether the radio button is checked.
CheckAlign	Gets or sets the location of the check box portion of the radio button control.
Image	Gets or sets the image that is displayed on a radio button control.
Font	Gets or sets the font of the text displayed by radio button control.

RadioButton Control

☐ Male ☐ Female

➤ Methods of RadioButton Control

Method	Description
Focus()	Sets input focus to the control.
ResetBackColor()	Resets the BackColor property to its default value.
ResetFont()	Resets the Font property to its default value.
ResetText()	Resets the Text property to its default value.
Select()	Activates the control.
Show()	Displays the control to the user.

➤ Events of RadioButton Control

Event	Description
CheckedChanged	Occurs when the value of the checked property of the radio button control is changed.
AppearanceChanged	Occurs when the value of the Appearance property of the radio Button control is changed.

CheckBox Control

- A CheckBox control allows users to select a single or multiple options from a list of options.
- CheckBox Control accepts either True or False as a value.
- Namespace : `System.Windows.Forms.CheckBox`
- Properties of CheckBox Control

Hobbies
☐ Music
☐ Cricket
☐ Reading

Property	Description
CheckAlign	Gets or sets the location of the check box portion of the checkbox.
CheckState	Gets or sets the state of the checkbox.
BackColor	Gets or sets the background color of the checkbox control.
Image	Gets or sets the image that is displayed on a checkbox control.
Font	Gets or sets the font of the text displayed by checkbox control.

CheckBox Control

➤ Methods of CheckBox Control

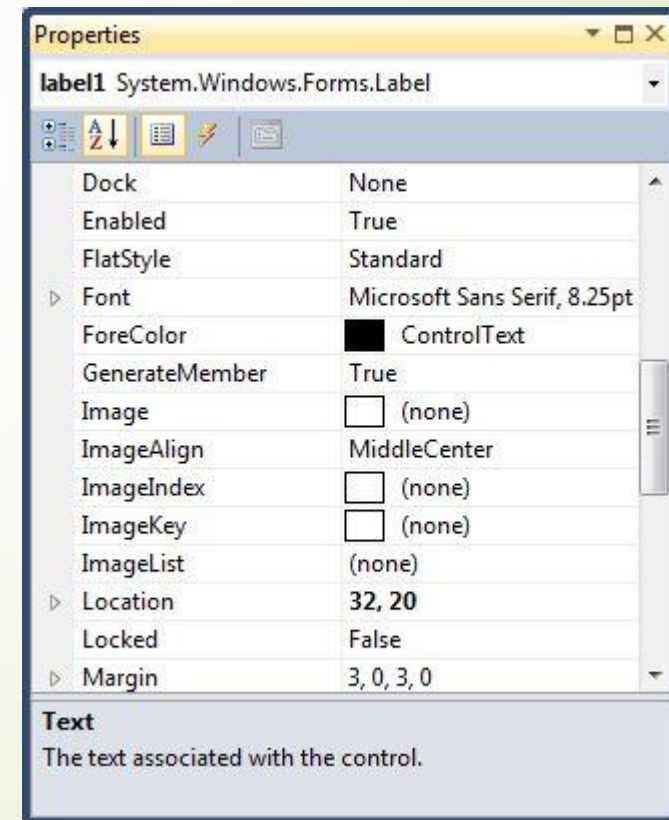
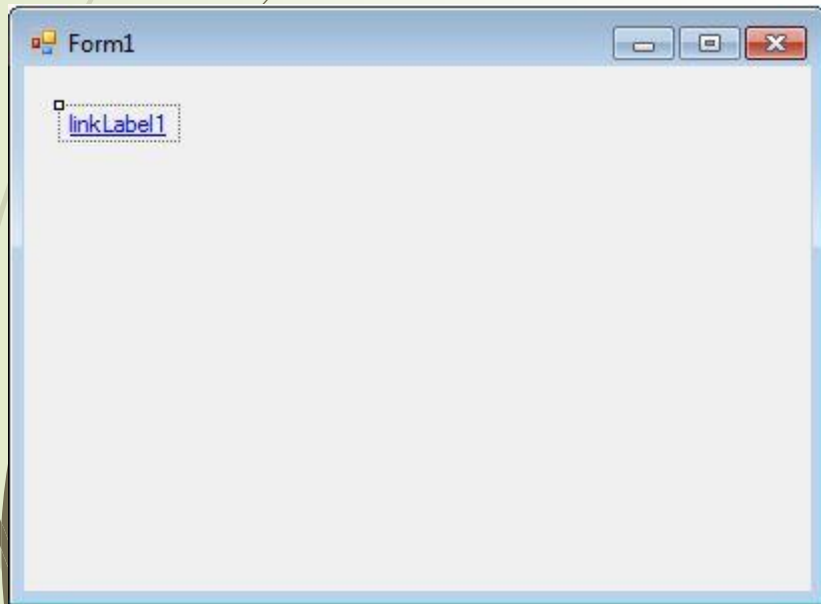
Method	Description
Focus()	Sets input focus to the control.
ResetBackColor()	Resets the BackColor property to its default value.
ResetFont()	Resets the Font property to its default value.
ResetText()	Resets the Text property to its default value.
Select()	Activates the control.
Show()	Displays the control to the user.

➤ Events of CheckBox Control

Event	Description
CheckedChanged	Occurs when the value of the Checked property of the CheckBox control is changed.
CheckStateChanged	Occurs when the value of the CheckState property of the CheckBox control is changed.
AppearanceChanged	Occurs when the value of the Appearance property of the CheckBox is changed

LinkLabel Control

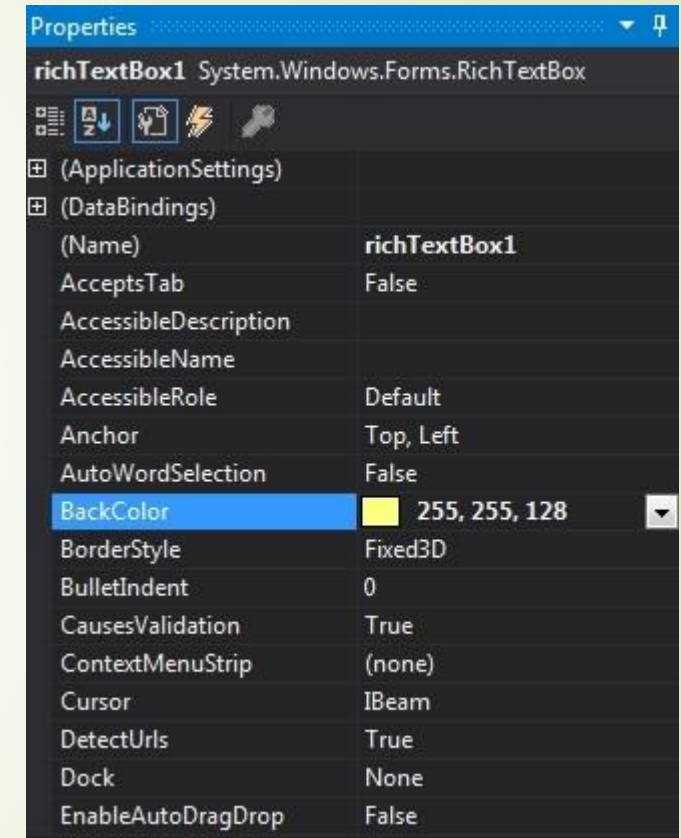
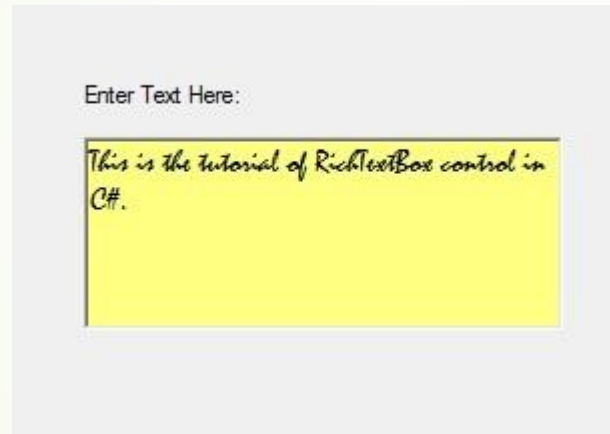
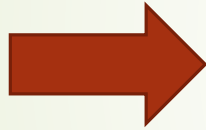
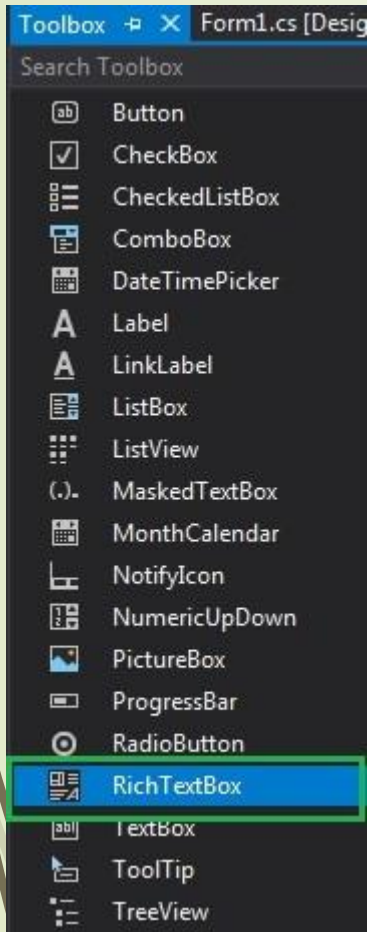
- A LinkLabel control is a label control that can display a hyperlink.
- A LinkLabel control is inherited from the Label class so it has all the functionality provided by the Windows Forms Label control.
- LinkLabel control does not participate in user input or capture mouse or keyboard events.



RichTextBox Control

- RichTextBox control is a textbox which gives you rich text editing controls and advanced formatting features.
- RichTextBox controls allows you to display or edit flow content, including paragraphs, images, tables, etc.
- The RichTextBox class is used to represent the windows rich text box and also provide different types of properties, methods, and events.
- It is defined under System.Windows.Forms namespace.

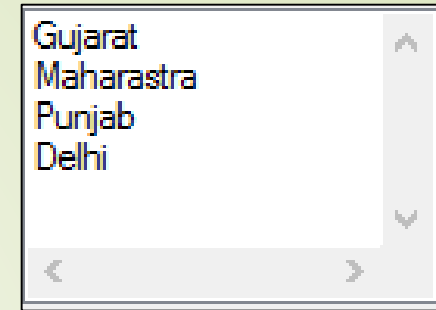
RichTextBox Control



RichTextBox Control

Property	Description
AutoSize	This property is used to get or set a value that indicates whether the control resizes based on its contents.
BackColor	This property is used to get or set the background color for the control.
BorderStyle	This property indicates the border style for the control.
Font	This property is used to get or set the font of the text displayed by the control.
ForeColor	This property is used to get or set the foreground color of the control.
Height	This property is used to get or set the height of the control.
Location	This property is used to get or set the coordinates of the upper-left corner of the RichTextBox control relative to the upper-left corner of its form.
Name	This property is used to get or set the name of the control.
TabStop	This property is used to get or set a value that shows whether the user can press the TAB key to provide the focus to the NumericUpDown.
Size	This property is used to get or set the height and width of the control.
Text	This property is used to get or set the text to be displayed in the RichTextBox control.
Visible	This property is used to get or set a value indicating whether the control and all its child controls are displayed.
Width	This property is used to get or set the width of the control.
ZoomFactor	This property is used to get or set the current zoom level of the RichTextBox.
ShowSelectionMargin	This property is used to get or set a value indicating whether a selection margin is displayed in the RichTextBox.
SelectionTabs	This property is used to get or set the absolute tab stop positions in a RichTextBox control.
SelectedText	This property is used to get or set the selected text within the RichTextBox.
ScrollBars	This property is used to get or set the type of scroll bars to display in the RichTextBox control.
Multiline	This property is used to get or set a value indicating whether this is a multiline RichTextBox control.

ListBox Control



- A ListBox control provides a user interface to display a list of items.
- Users can select one or more items from the list.
- If the items exceed a specified limit, a scroll bar automatically appears to let the user to scroll through the list.
- Namespace : `System.Windows.Forms.ListBox`
- **Properties of ListBox Control**

Property	Description
Items	Gets the items of the ListBox.
SelectedItem	Gets or sets the currently selected item in the ListBox.
SelectedIndex	Gets or sets the zero-based index of the currently selected item in a ListBox.
SelectionMode	Indicates if the list box is to be single-select, multi-select, or non-selectable.

ListBox Control

➤ Methods of ListBox Control

Method	Description
ClearSelected()	Unselects all items in the ListBox.
Sort()	Sorts the items in the ListBox.
ResetBackColor()	Resets the BackColor property to its default value.
ResetFont()	Resets the Font property to its default value.
ResetText()	Resets the Text property to its default value.
Select()	Activates the control.
Show()	Displays the control to the user.

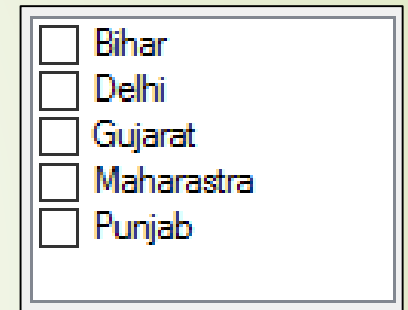
➤ Events of ListBox Control

Event	Description
SelectedIndexChanged	Occurs when the SelectedIndex property has changed.
SizeChanged	Occurs when the Size property value changes.

CheckedListBox Control

- The Windows Forms CheckedListBox control displays a list of items, like the ListBox control, and also can display a check mark next to items in the list.
- Users can select one or more items from the list.
- Namespace : System.Windows.Forms.CheckedListBox
- **Properties of CheckedListBox Control**

Property	Description
CheckedItems	Collection of checked items in this CheckedListBox.
CheckedIndices	Collection of checked indexes in CheckedListBox.
CheckOnClick	Gets or sets a value indicating whether the check box should be toggled when an item is selected.
Items	Gets the collection of items in this CheckedListBox.
Sorted	Gets or sets a value indicating whether the items in the CheckedListBox are sorted alphabetically.



<input type="checkbox"/>	Bihar
<input type="checkbox"/>	Delhi
<input type="checkbox"/>	Gujarat
<input type="checkbox"/>	Maharastra
<input type="checkbox"/>	Punjab

CheckedListBox Control

➤ Methods of CheckedListBox Control

Method	Description
ClearSelected()	Unselects all items in the CheckedListBox.
Sort()	Sorts the items in the CheckedListBox.
ResetBackColor()	Resets the BackColor property to its default value.
ResetFont()	Resets the Font property to its default value.
ResetText()	Resets the Text property to its default value.
Select()	Activates the control.
Show()	Displays the control to the user.

➤ Events of CheckedListBox Control

Event	Description
Click	Occurs when the user clicks the CheckedListBox control.
ItemCheck	Occurs when the checked state of an item changes.
SelectedIndexChanged	Occurs when the SelectedIndex property or the SelectedIndices collection has changed.
SelectedValueChanged	Occurs when the SelectedValue property changes.

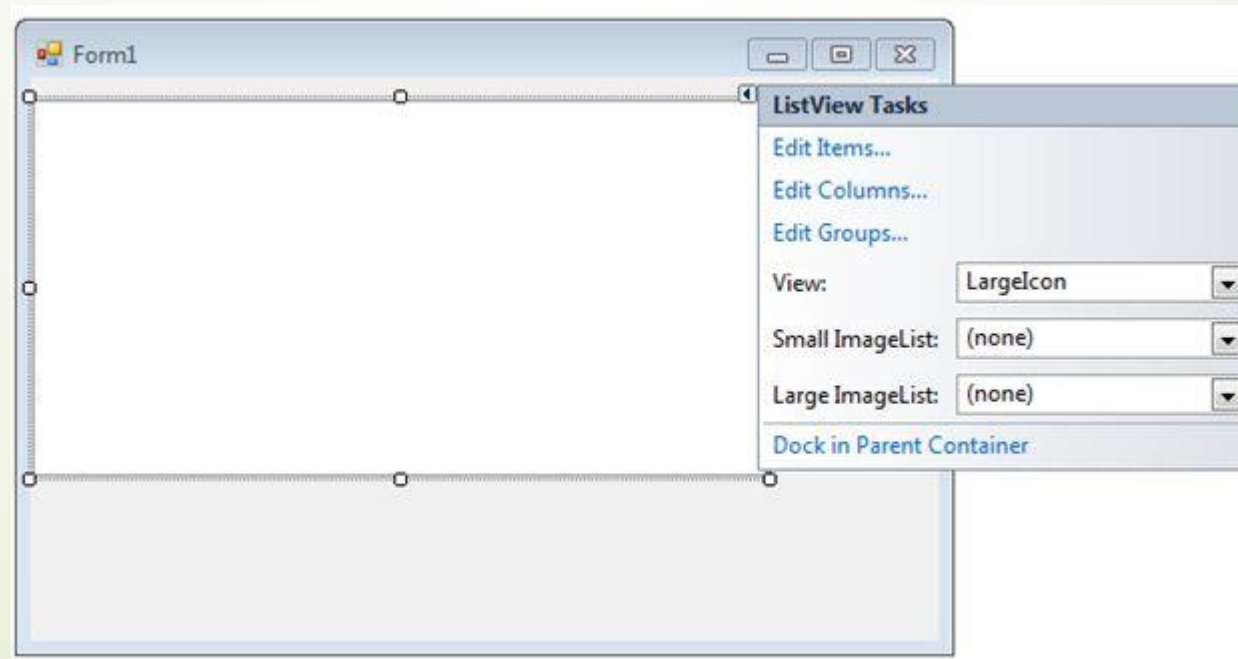
CheckedListBox Control

► Events of CheckedListBox Control

Event	Description
Click	Occurs when the user clicks the CheckedListBox control.
ItemCheck	Occurs when the checked state of an item changes.
SelectedIndexChanged	Occurs when the SelectedIndex property or the SelectedIndices collection has changed.
SelectedValueChanged	Occurs when the SelectedValue property changes.

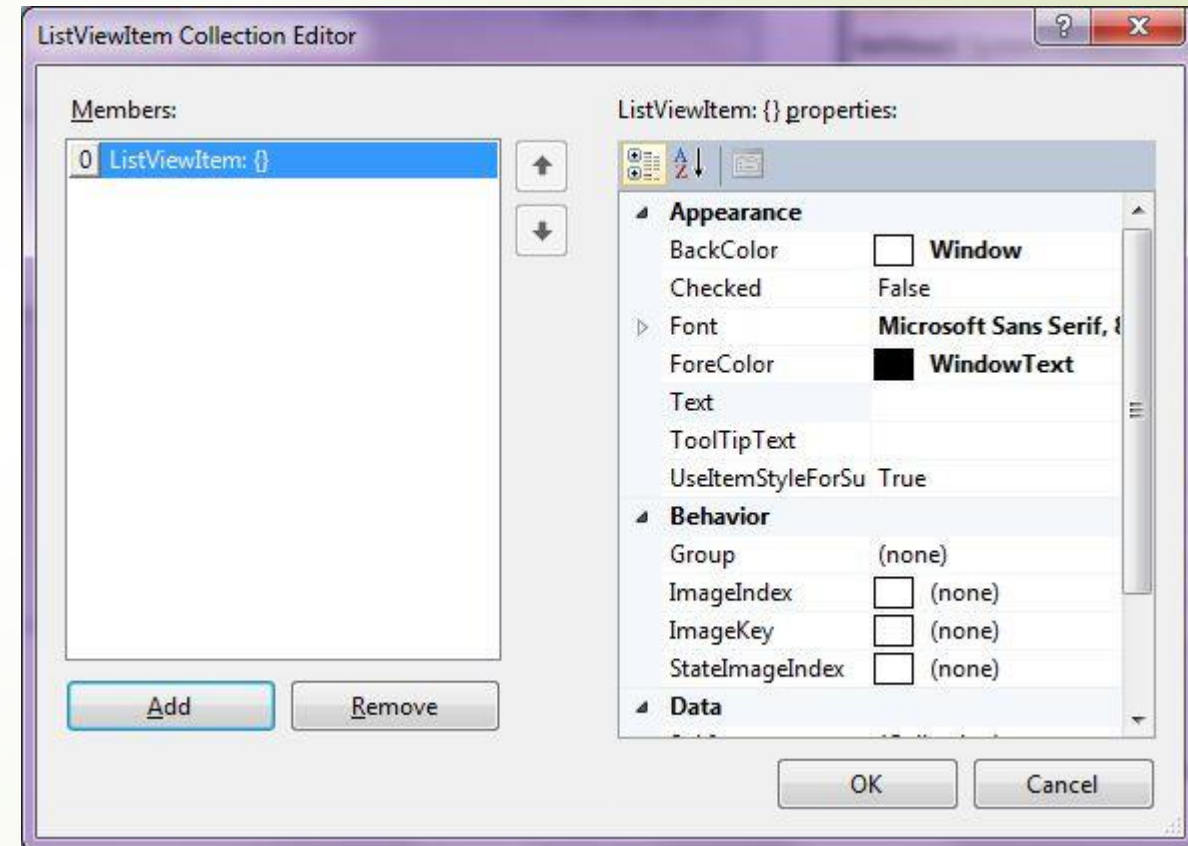
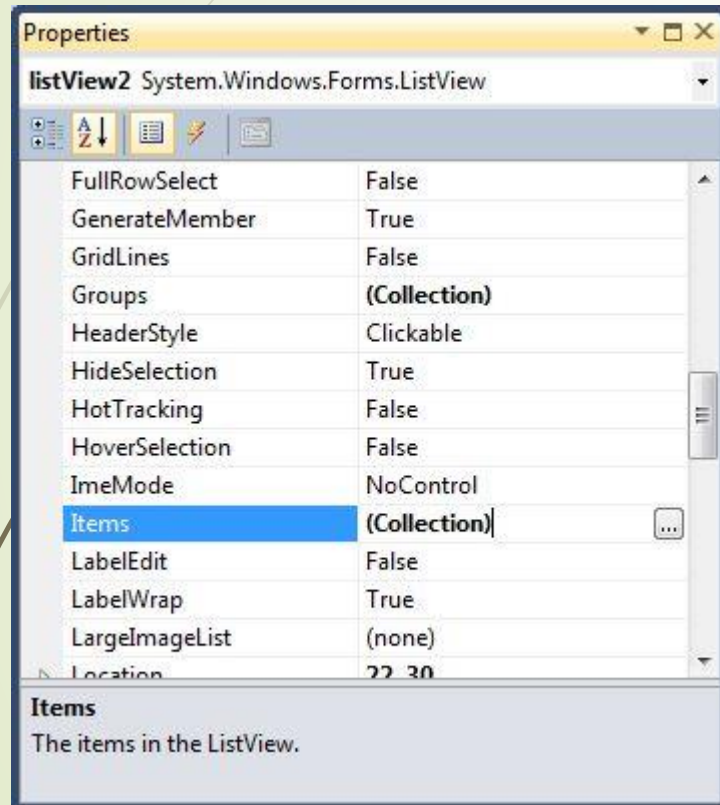
List View Control

- C# ListView control provides an interface to display a list of items using different views including text, small images, and large images.
- There are two approaches to create a ListView control in Windows Forms. Either we can use the Forms designer to create a control at design-time or we can use the ListView class to create a control at run-time.



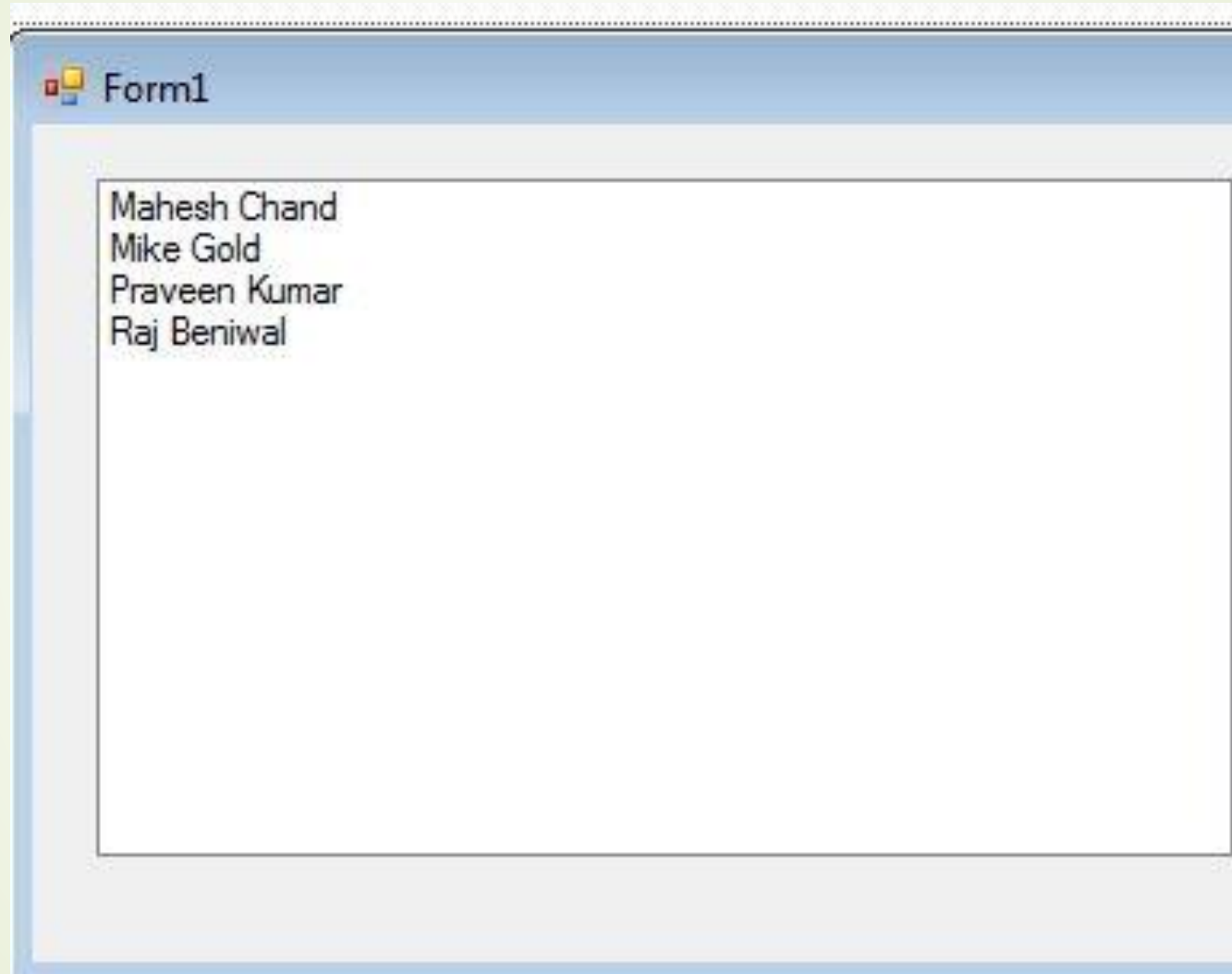
List View Control

➡ C#



List View Control

➤ C#

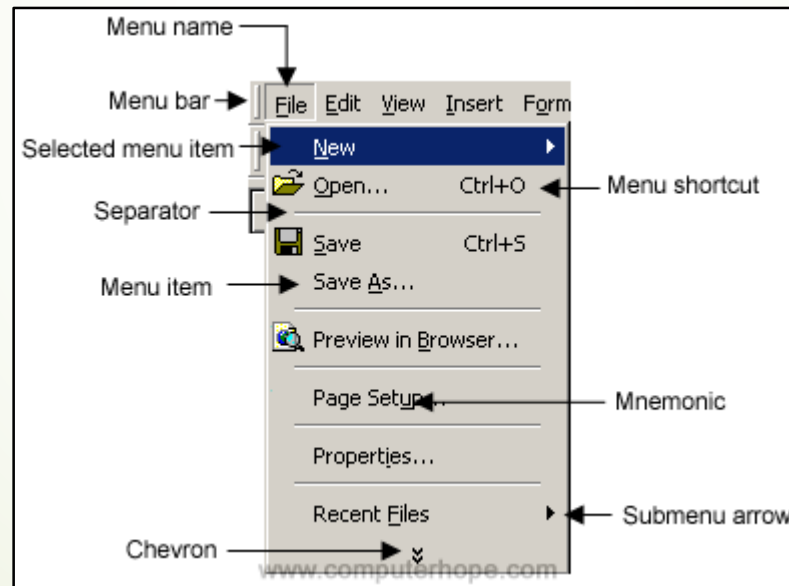


Menus

- Menus are controls that allows a user to make selections.
- They also hide the selections that are not needed, thereby saving space in windows applications.
- **Menus act like containers for the ToolStripMenuItem** objects and are used to display menu and menu items in a menu bar.
- Any item that resides on a menu is known as a menu item and represents an individual part of a menu.
- Namespace : `System.Windows.Forms.MenuStrip`.

Menus

- There are main two classes in the standard menu handling.
- MenuStrip : Acts as a container for the menu structure of a form.
- ToolStripMenuItem : Supports the items in a menu system (Including the menus such as File & Edit)

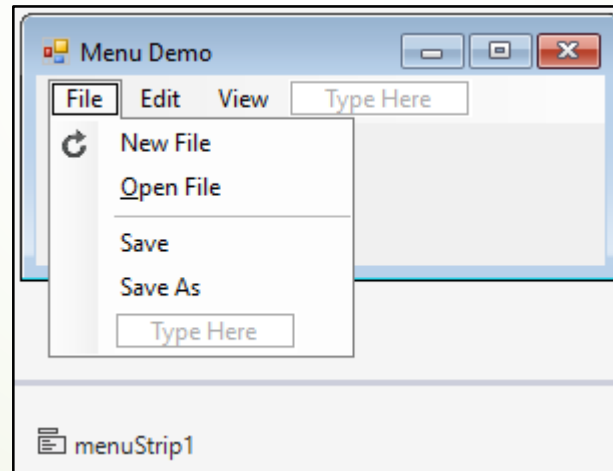


Menus

- **ToolStripMenuItem Class**
- The ToolStripMenuItem class supports the menus and menu items in a menu system.
- These menu items are objects that can handle through Click events in a menu system.
- It has the properties that permits you to configure the functionality and appearance of a menu item like add shortcut keys, font, back color etc.
- Namespace : **System.Windows.Forms.ToolStripItem.**

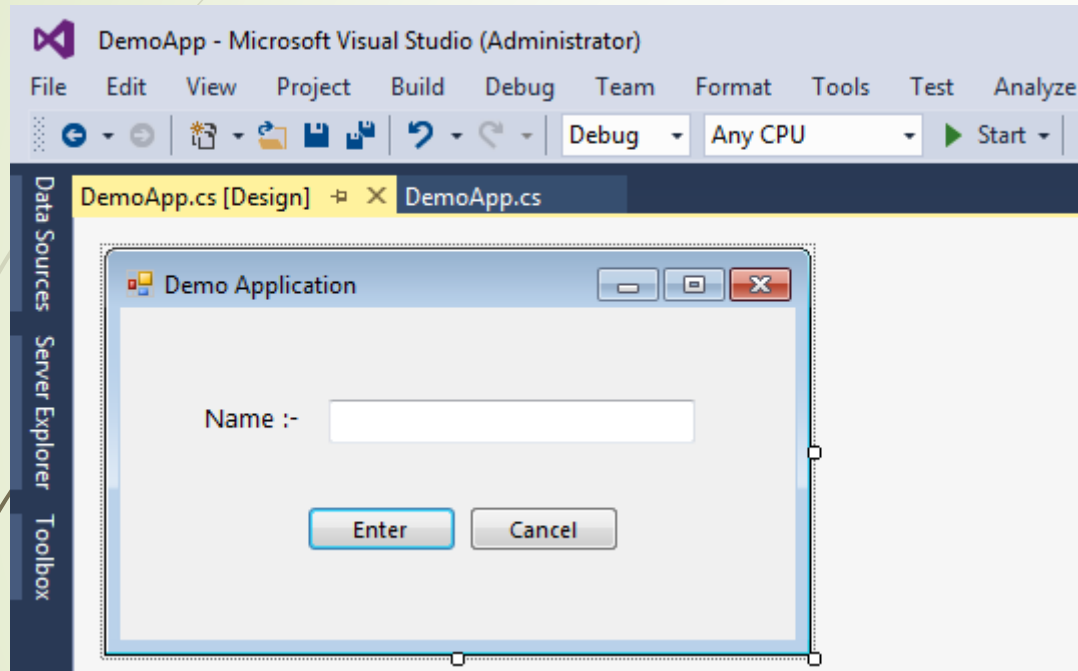
Menus

- **ToolStripMenuItem Class**
- Click the Type Here text to open a text box and enter 'File' text in it. Now you can add some more items to the File menu.
- You can make the menu items perform some actions by creating event handlers for handling their Click event in the code editor.



Validation Control

- Step 1: Create a Windows form application.



Validation Control

- Step 2: Choose “ErrorProvider” form toolbox..

The screenshot displays the Visual Studio IDE with a Windows Forms application named 'DemoApp.cs'. The 'Toolbox' on the left shows the 'ErrorProvider' control selected under the 'Components' section. The 'Properties' window on the right shows the properties for the 'errorProviderApp' control, including 'Icon', 'Behavior', 'Data', and 'Design' sections. The 'Design' section shows the control's name as 'errorProviderApp'.

Toolbox

- All Windows Forms
- Common Controls
- Containers
- Menus & Toolbars
- Data
- Components
 - Pointer
 - BackgroundWorker
 - DirectoryEntry
 - DirectorySearcher
 - ErrorProvider**
 - EventLog
 - FileSystemWatcher
 - HelpProvider

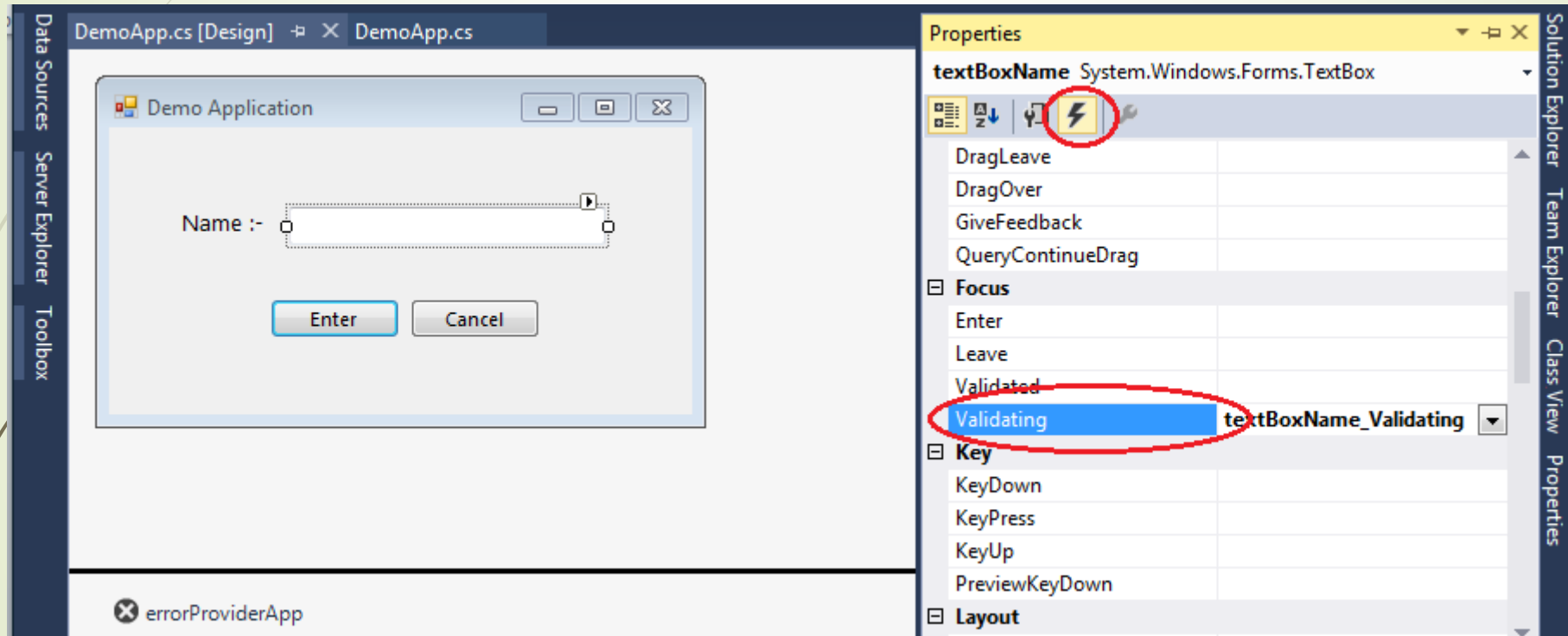
Properties

errorProviderApp System.Windows.Forms.ErrorProvider

Icon	(Icon)
RightToLeft	False
Behavior	
BlinkRate	250
BlinkStyle	BlinkIfDifferentError
Data	
(ApplicationSettings)	
ContainerControl	DemoApp
DataMember	
DataSource	(none)
Tag	
Design	
(Name)	errorProviderApp
GenerateMember	True
Modifiers	Private

Validation Control

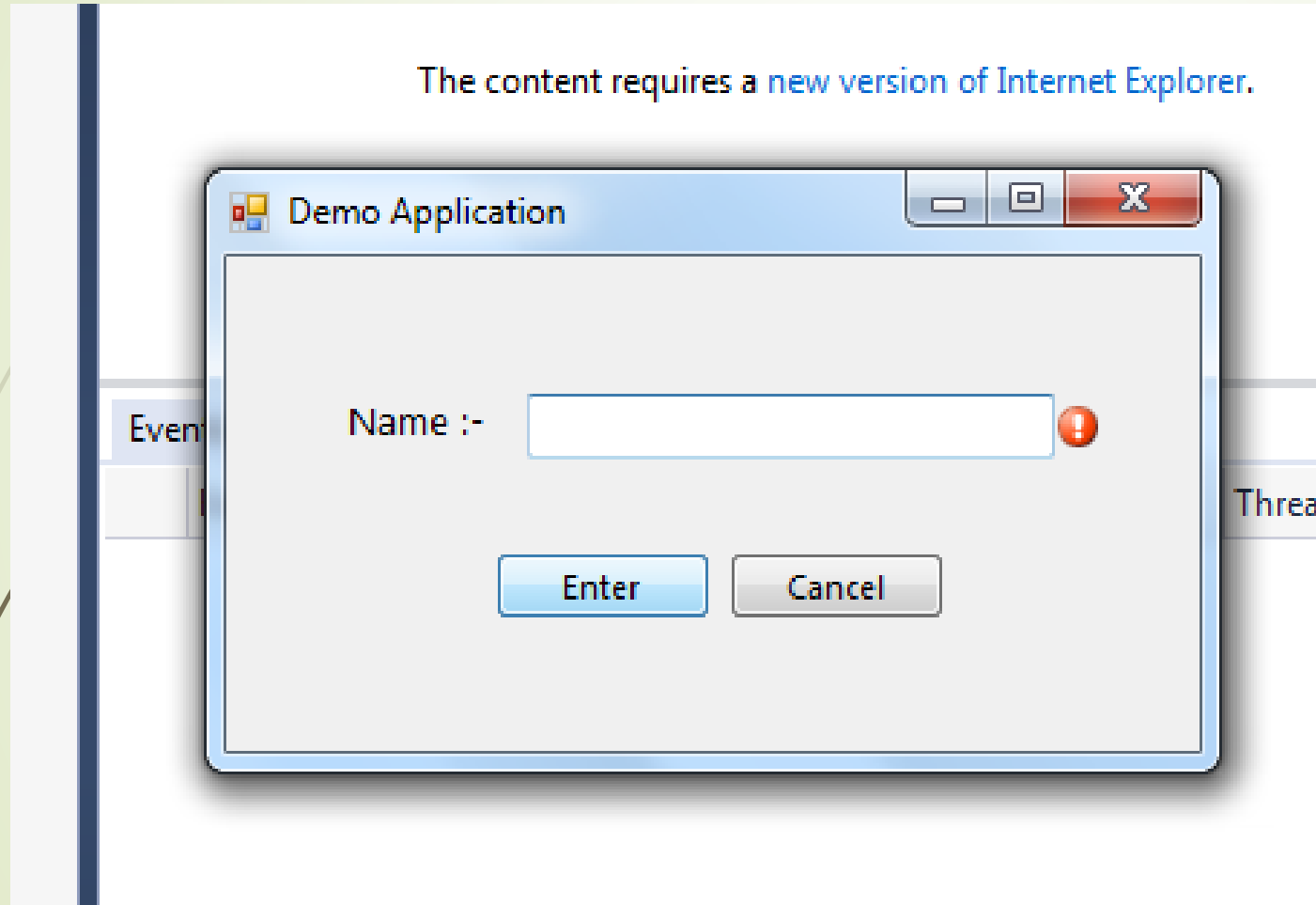
- Step 3: Select the Text box and go to its properties.



Validation Control

```
private void textBoxName_Validating(object sender, CancelEventArgs e)
{
    if (string.IsNullOrEmpty(textBoxName.Text))
    {
        e.Cancel = true;
        textBoxName.Focus();
        errorProviderApp.SetError(textBoxName, "Name should not be left blank!");
    } else
    {
        e.Cancel = false;
        errorProviderApp.SetError(textBoxName, "");
    }
}
```

Validation Control

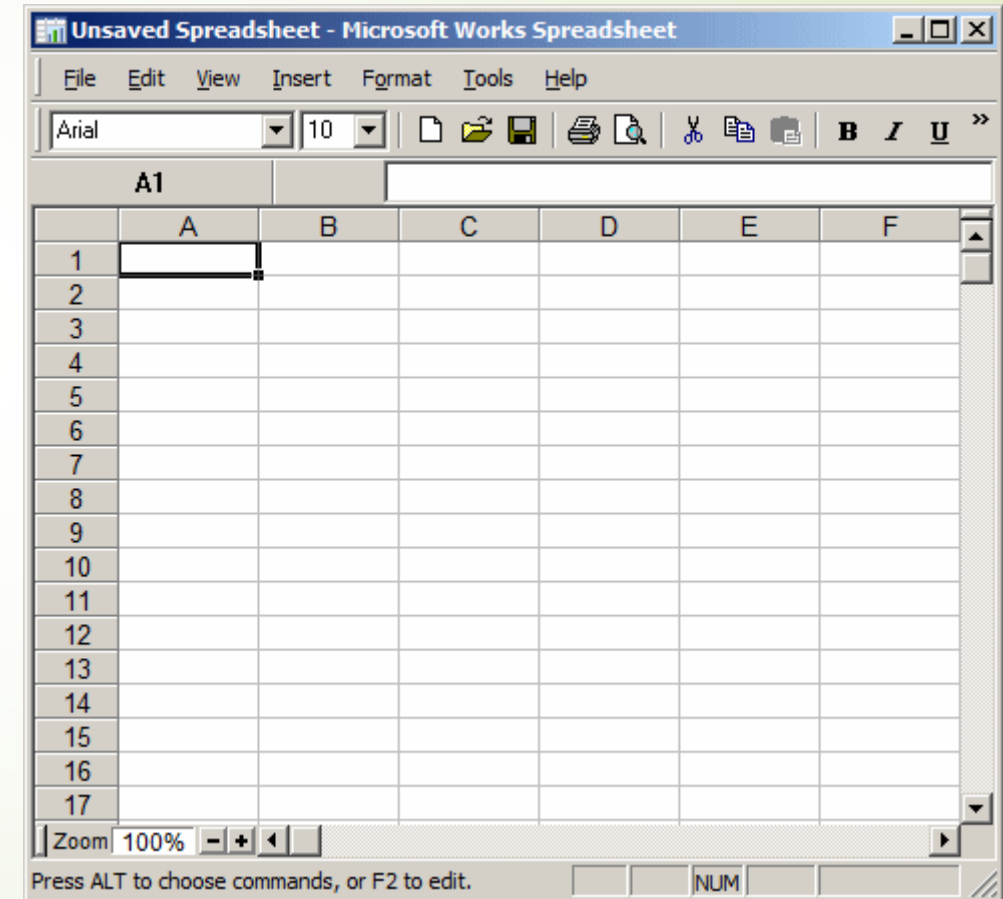
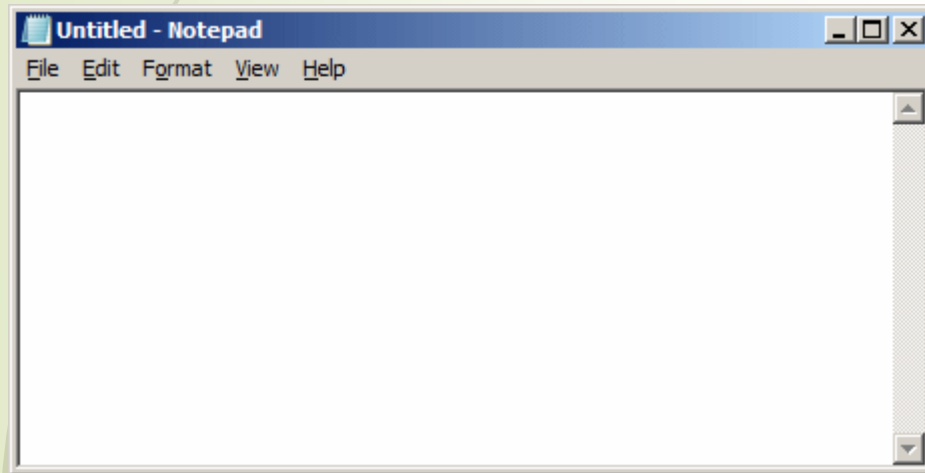


SDI and MDI Application

- A **single-document interface, SDI**, is an application primarily made of a form equipped with a menu. An example is Notepad:
- In some cases, an SDI can also have a toolbar and/or a status bar. Here is an example from Microsoft Works Spreadsheet:
- All these features are left to the programmer to add and configure.
- Although Notepad is text-based, an SDI can be any type of application, text, graphics, spreadsheet, anything. Therefore, to create an SDI, start from a normal form, add a menu to it, and configure it to do what you want.
- To create a document using an SDI, the user launches the application. The SDI then presents a rectangular window with one frame and the inside is the document the user will use. In most cases, the application itself creates the document. The user can work on it and do whatever the application allows. To create another document of the same type, the user must open another instance of the application.

SDI and MDI Application

- Example of single-document interface - SDI

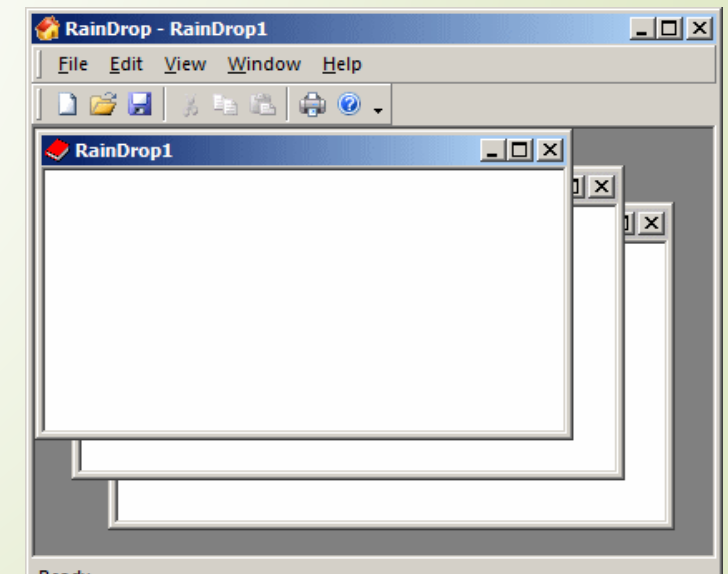
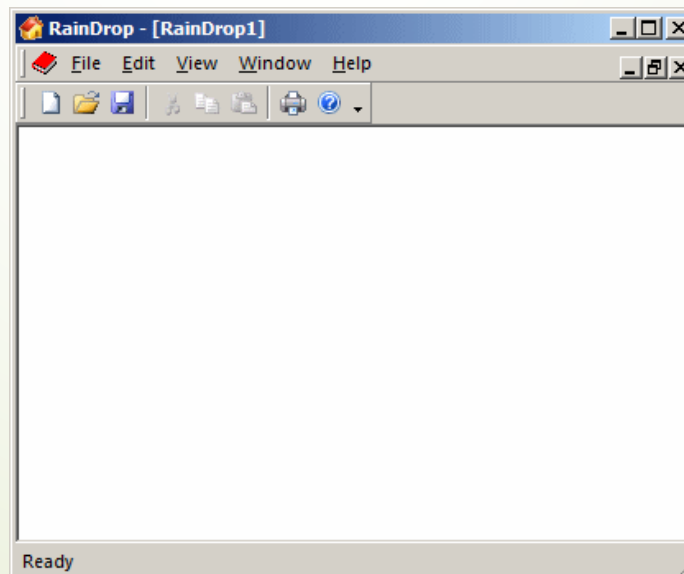
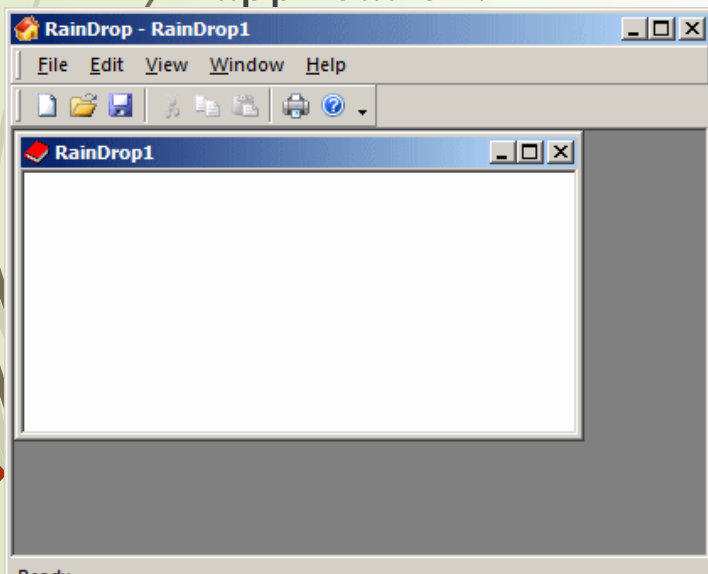


SDI and MDI Application

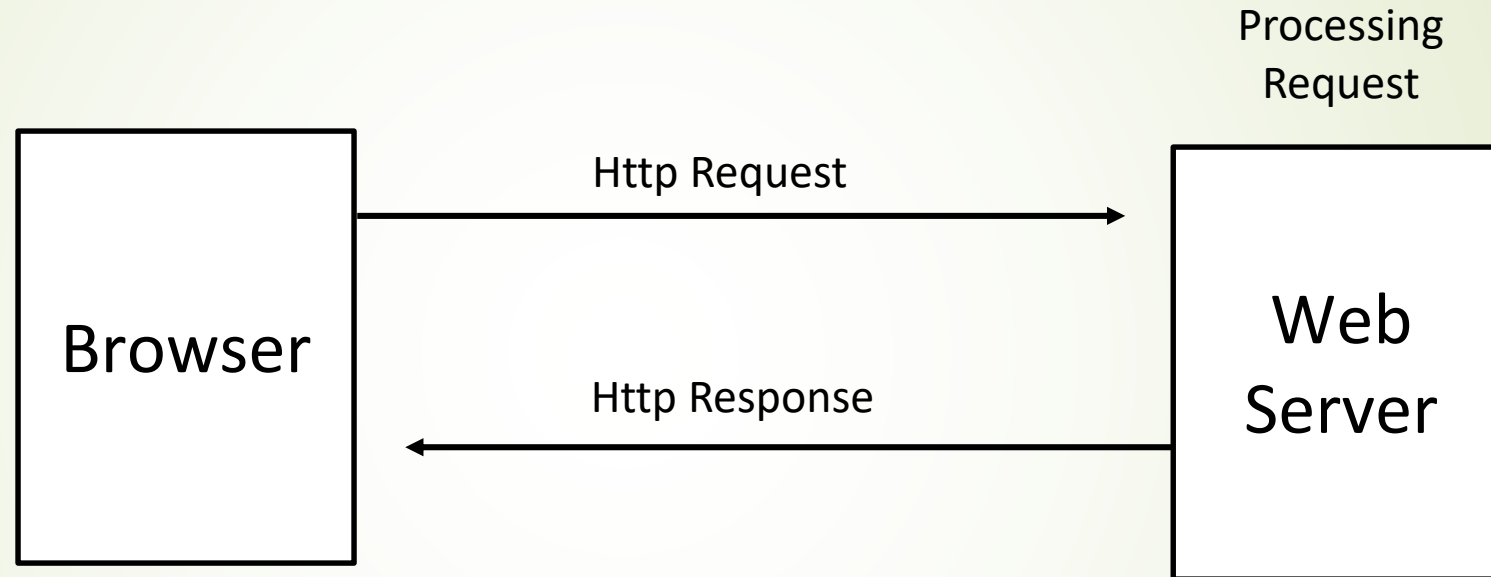
- **A multiple-document interface, MDI**, is an application that primarily has a form and a menu. Some, if not most MDIs also have one or more toolbars and/or a status bar.
- Like a normal application, to use an MDI, the user must launch it.
- In some cases, when the application starts, it is empty; that is, no document is created and the title bar displays a caption, usually the name of the application.
- Usually, there are steps the user must follow to create a document. In some other cases, when the application is launched, it automatically creates a document. A document resides inside the parent frame of the application. That is, a child document can use only the area reserved for it. The child document has its own system icon, its own title bar, and its system buttons (Minimize, Maximize/Restore, and Close).

SDI and MDI Application

- To use the whole area, the user can maximize the child document. When this is done, the child merges its title bar with the parent's. The new caption of the title bar becomes made of the text of the parent, followed by -, and followed by the caption the child window was using. The system buttons of the child document display under those of the parent frame:
- Once a document has been created, the user can use it. The application must give the user the ability to create other documents while still using the application. If many documents have been created, all of them are confined in the frame of the application:



.Net Server Controls



.Net Server Controls

➤ What is a Website ?

- A website is a collection of related web pages, including multimedia content, typically identified with a common domain name, and published on at least one web server.

➤ What is webpage?

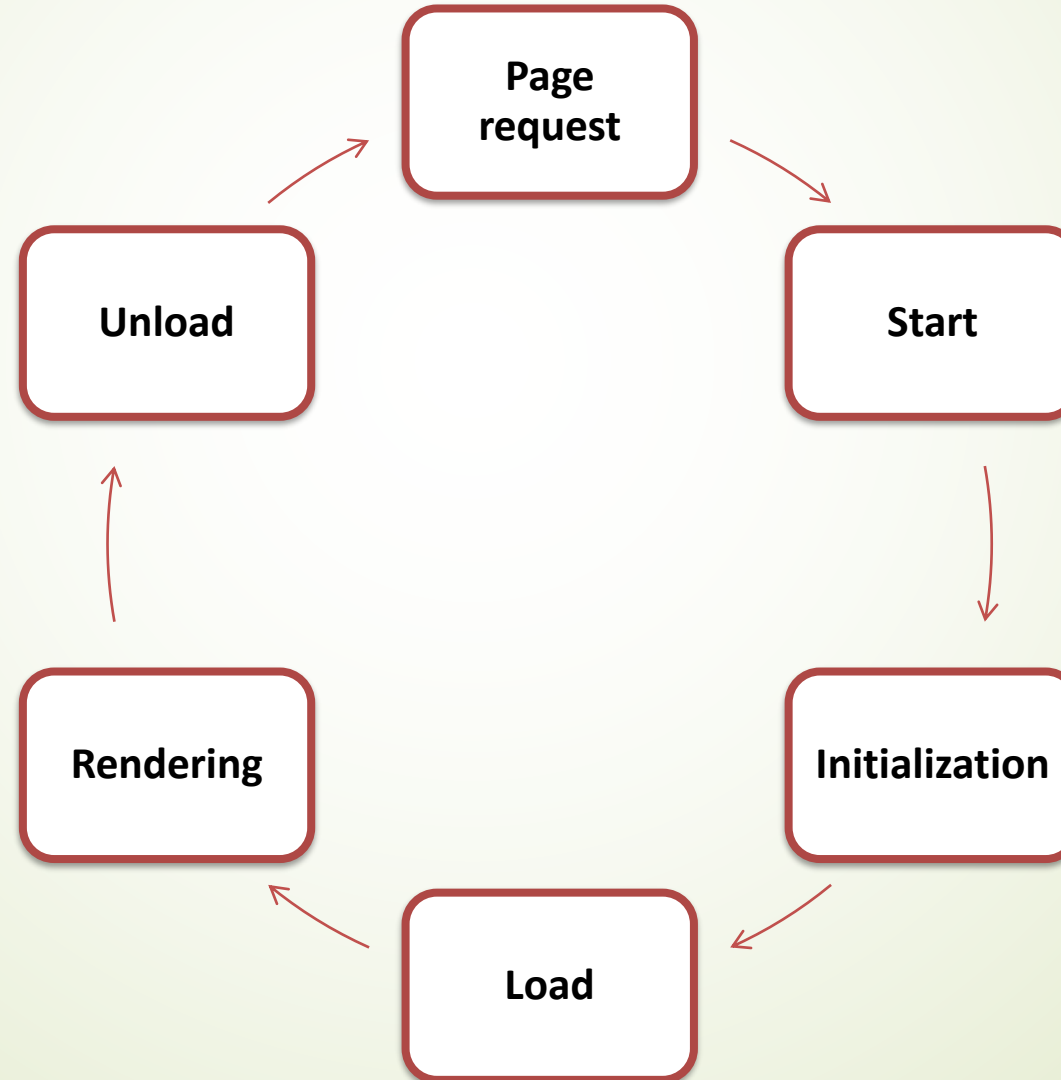
- A webpage is a document commonly written in Hypertext Markup Language (HTML) that is accessible through the Internet or other network using an Internet browser.

➤ ASP Refers to Active Server Pages.

- ASP.NET is a server-side web application framework designed for web development to work with dynamic web pages.
- It is developed by Microsoft to allow programmers to build **dynamic web sites, web applications and web services.**
- ASP.NET pages have the extension .aspx.

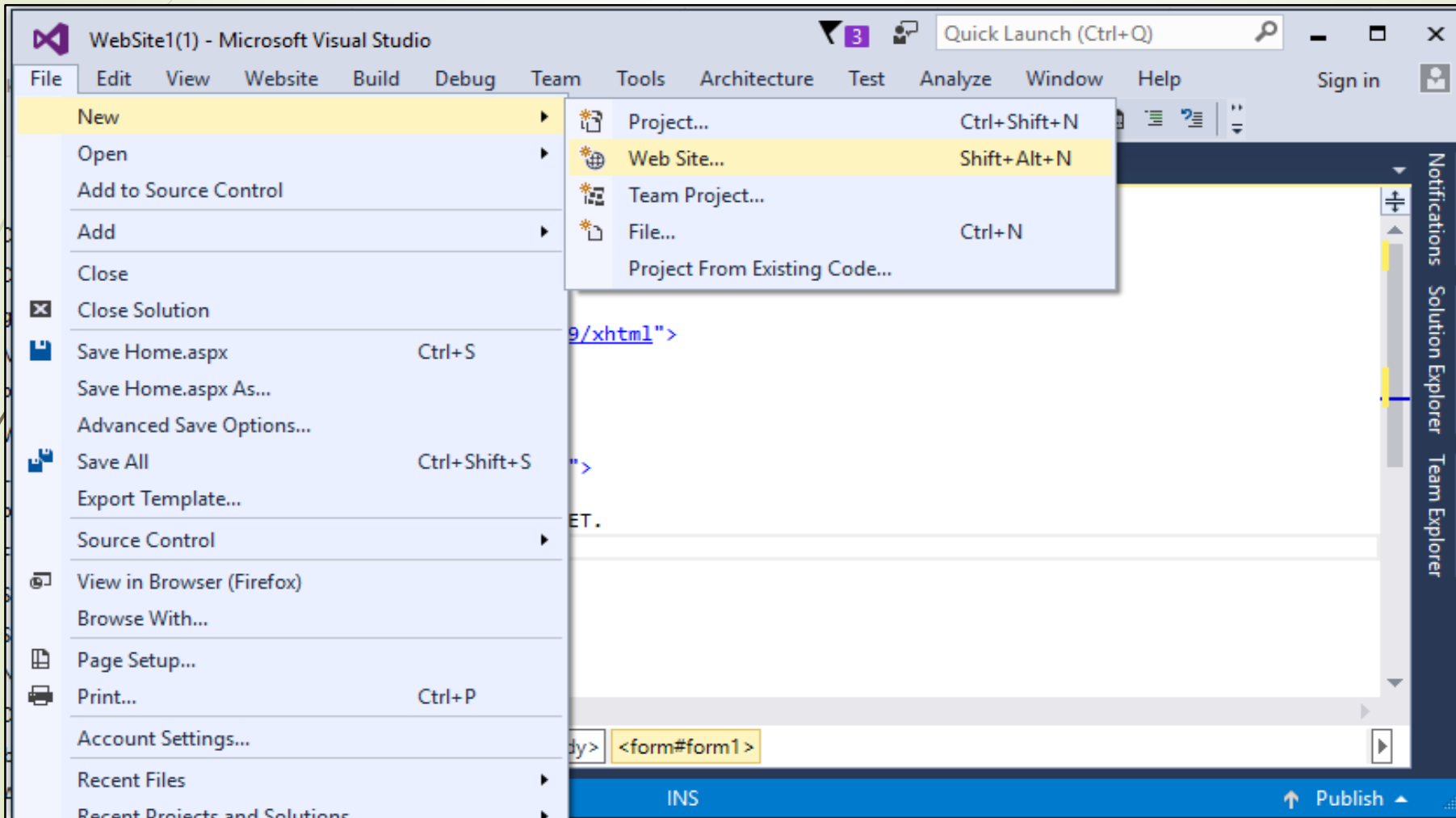
.Net Server Controls

- The life cycle of the ASP.NET page specifies how the page is instantiated and processed to generate the specific output.

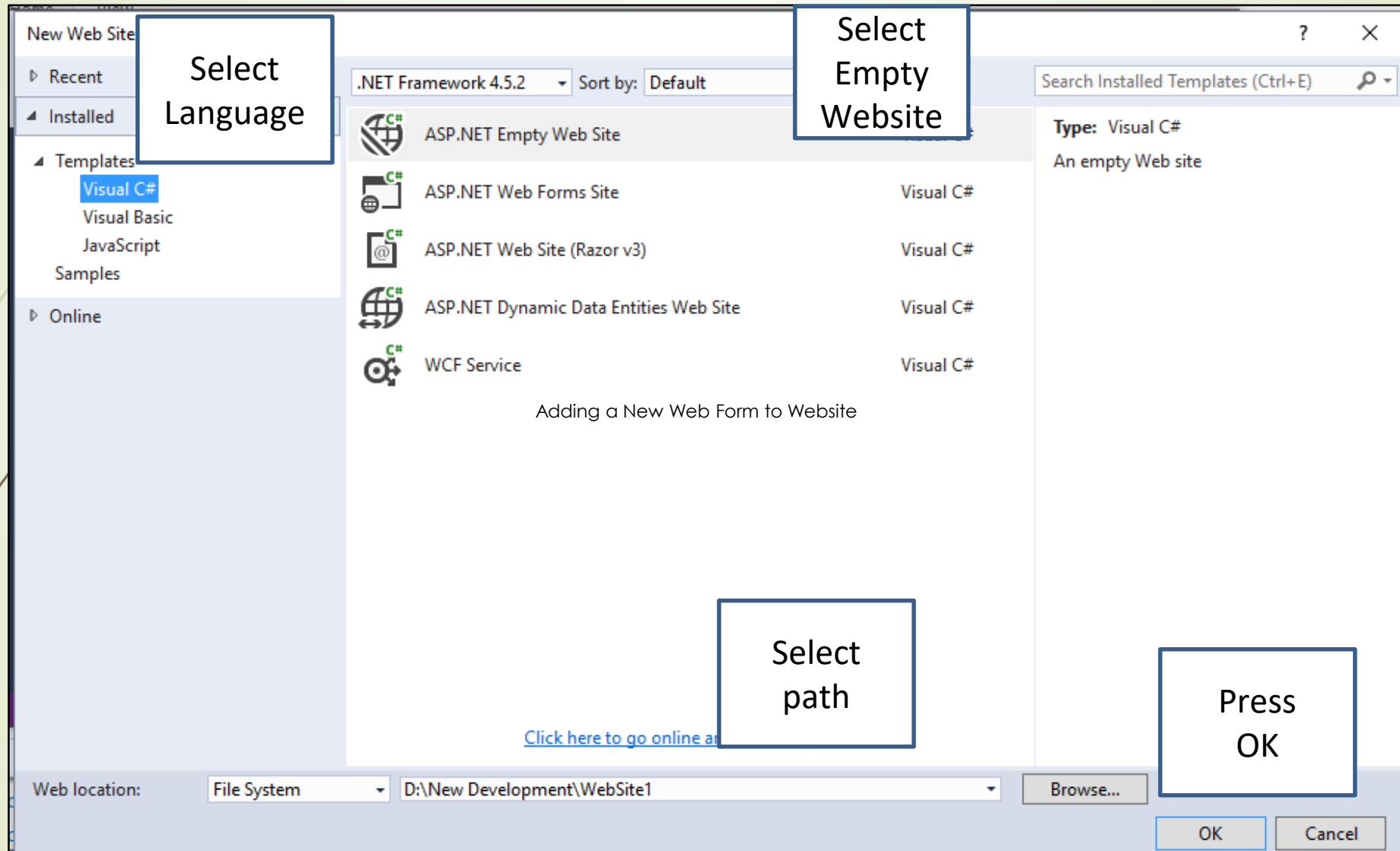


.Net Server Controls

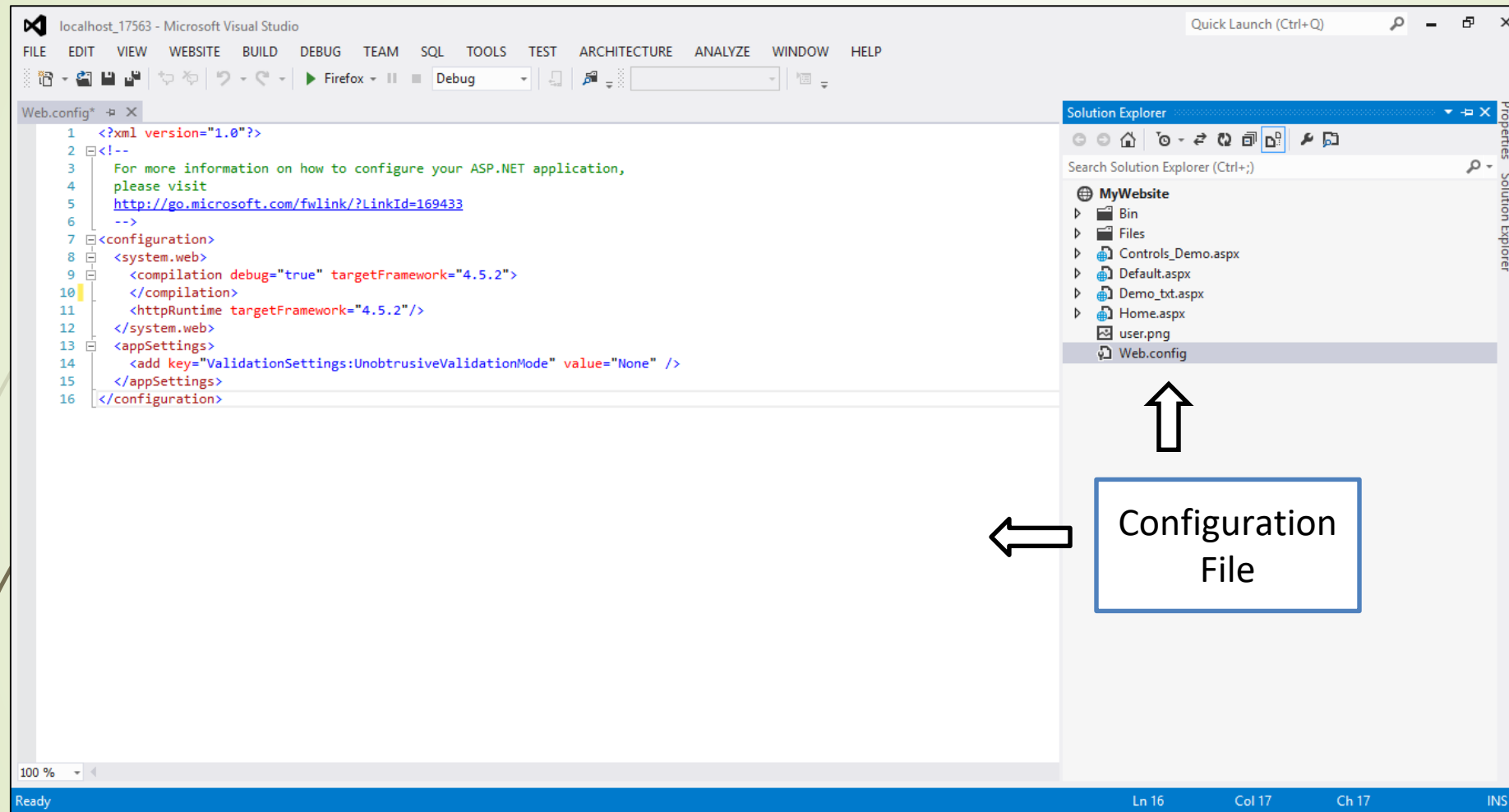
- How to Create Website? → Start Visual Studio and click File menu → New → Web Site.



.Net Server Controls

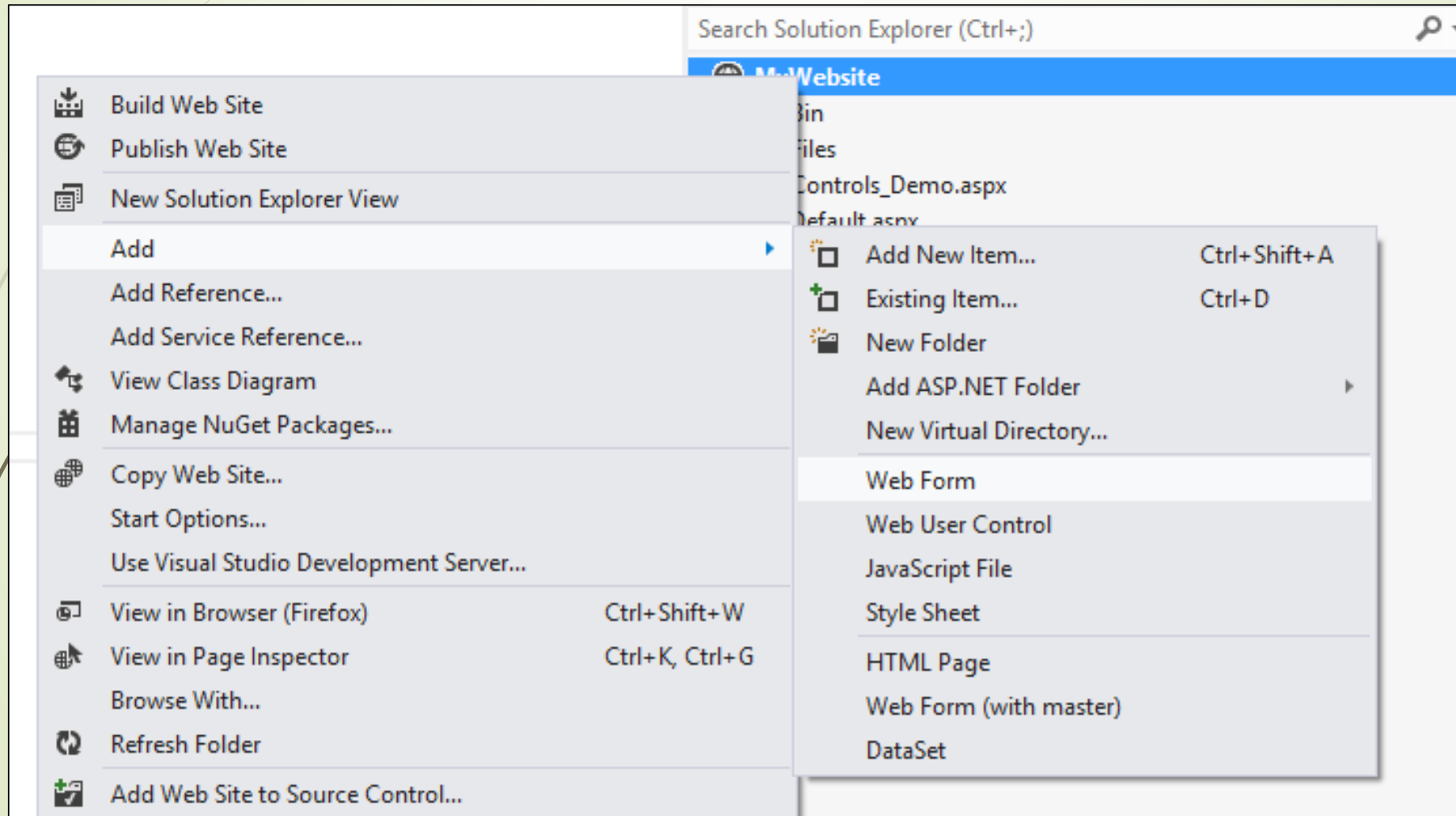


.Net Server Controls



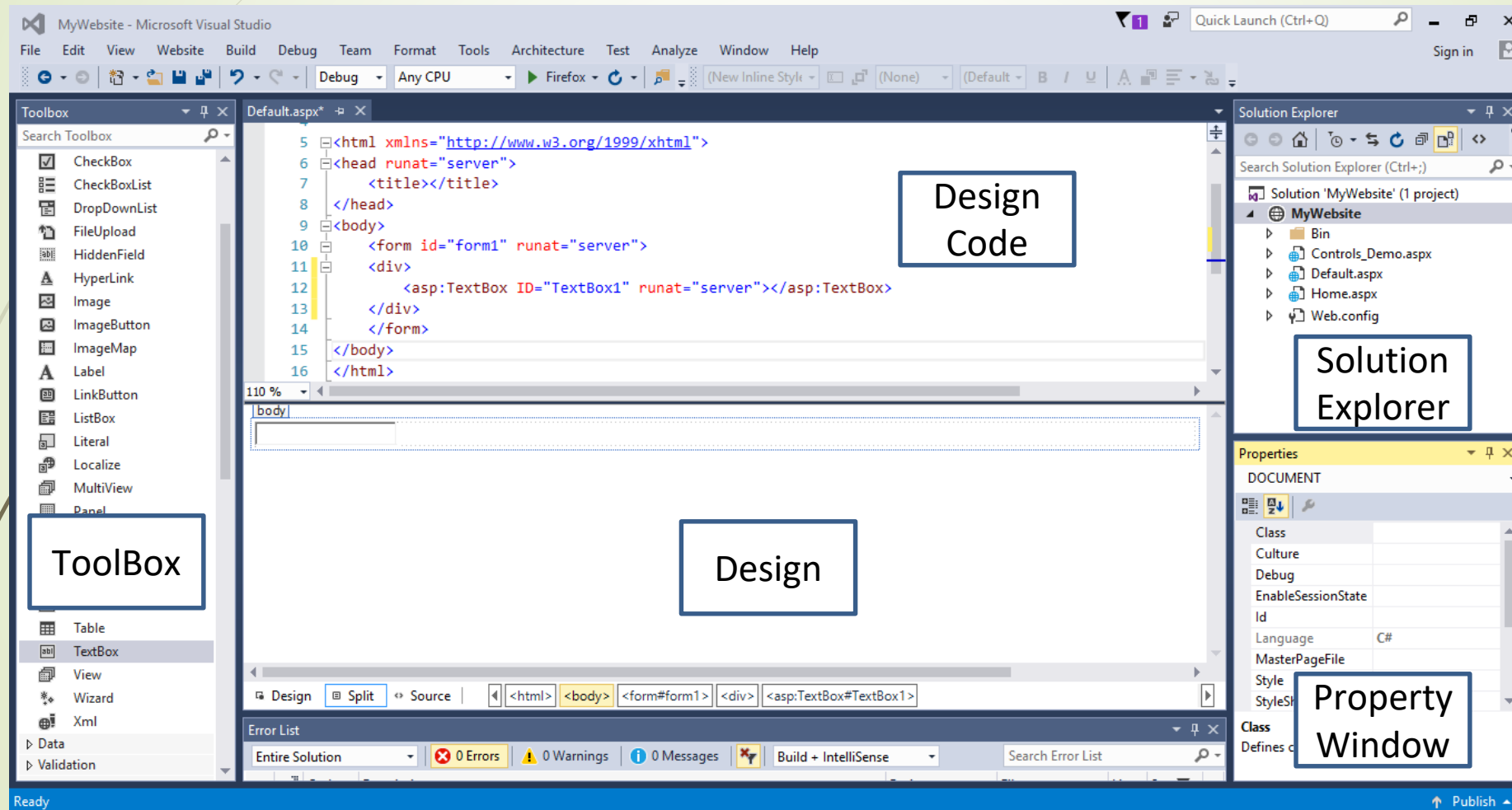
.Net Server Controls

➤ Adding a New Web Form to Website



.Net Server Controls

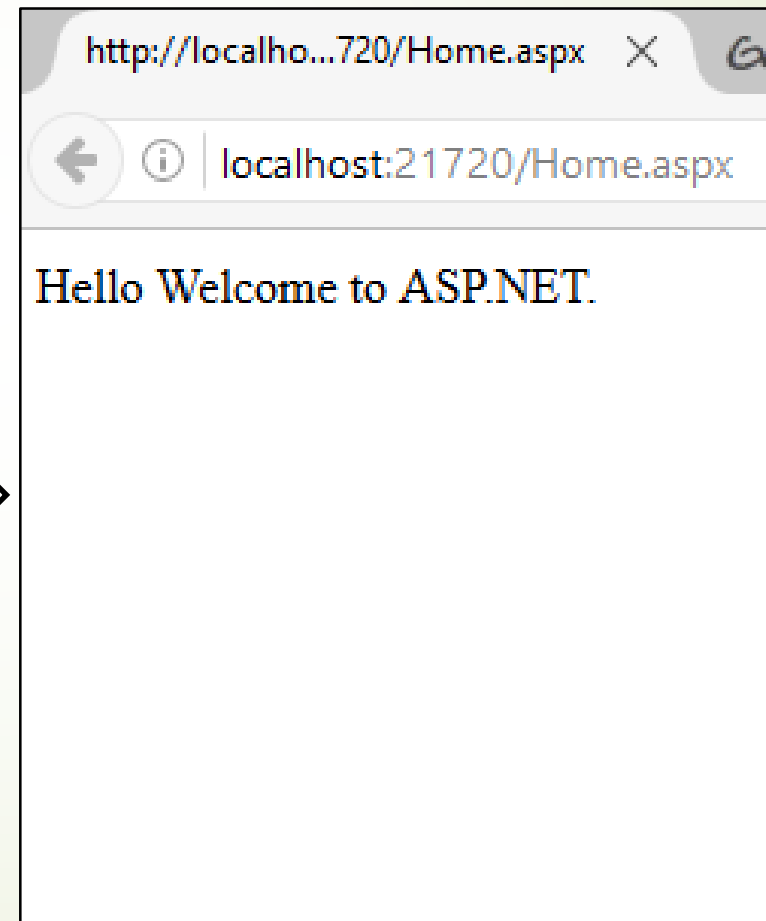
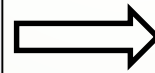
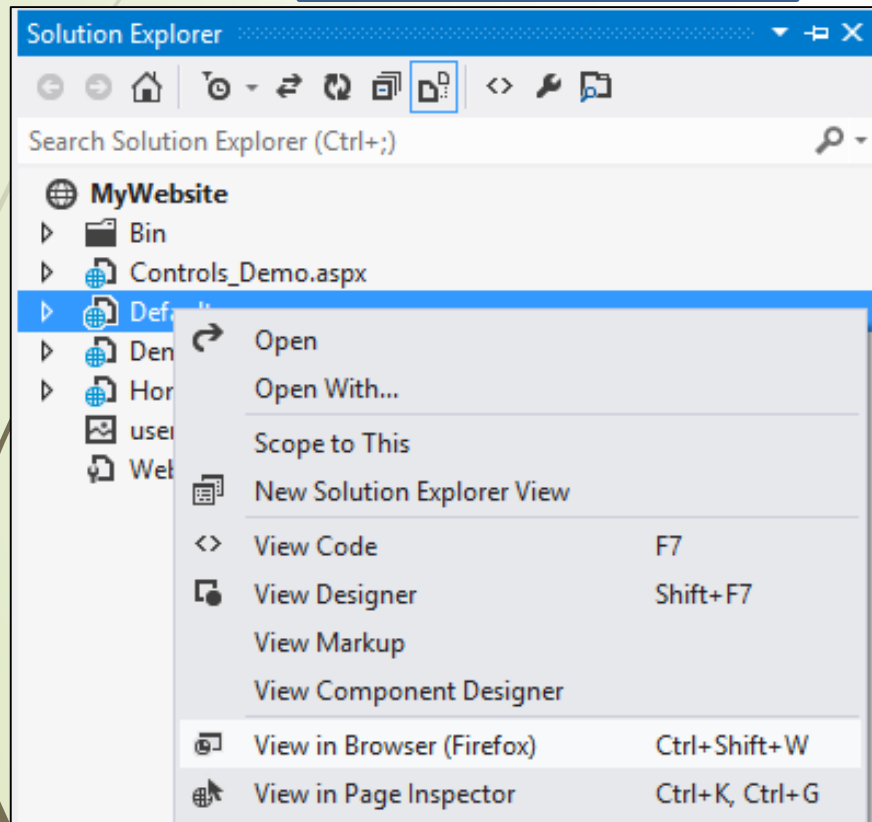
► Design Page of Website (.aspx)



.Net Server Controls

► Viewing an Output

Right click on Web
Form & click View in
Browser



.Net Server Controls

- The ASP.NET controls are designed similar to **standard Windows forms controls**.
- This controls are used to design the interface of a web application.
- **Example**
 - When you visit the website of Gmail, you type your user name and password in **text box**, which is one control.
 - **ASP.NET provides a standard set of controls that can be used to develop web applications.**

.Net Server Controls

- These controls can be easily used by dragging and dropping them at any desired location on a web form.
- Based on tasks performed by them, these controls are grouped under various categories.
- For example controls for validating the data are placed under the validation tab and controls used for logging on to websites are placed under the Login tab.
- The **System.Web.UI.Control** namespace is the base class for all web server controls.

.Net Server Controls

- The standard web form controls are inherited from a common base class namely **System.Web.UI.WebControls** class.

Standard Controls

Label Control

TextBox Control

Button Control

CheckBox Control

RadioButton Control

HyperLink Control

Image Control

FileUpload Control

LinkButton Control

ImageButton Control

HiddenField Control

Validation Controls

RequiredFieldValidator

RangeValidator

CompareValidator

RegularExpressionValidator

CustomValidator

ValidationSummary

List Controls

DropDownList

CheckBoxList

RadioButtonList

ListBox

Master Page

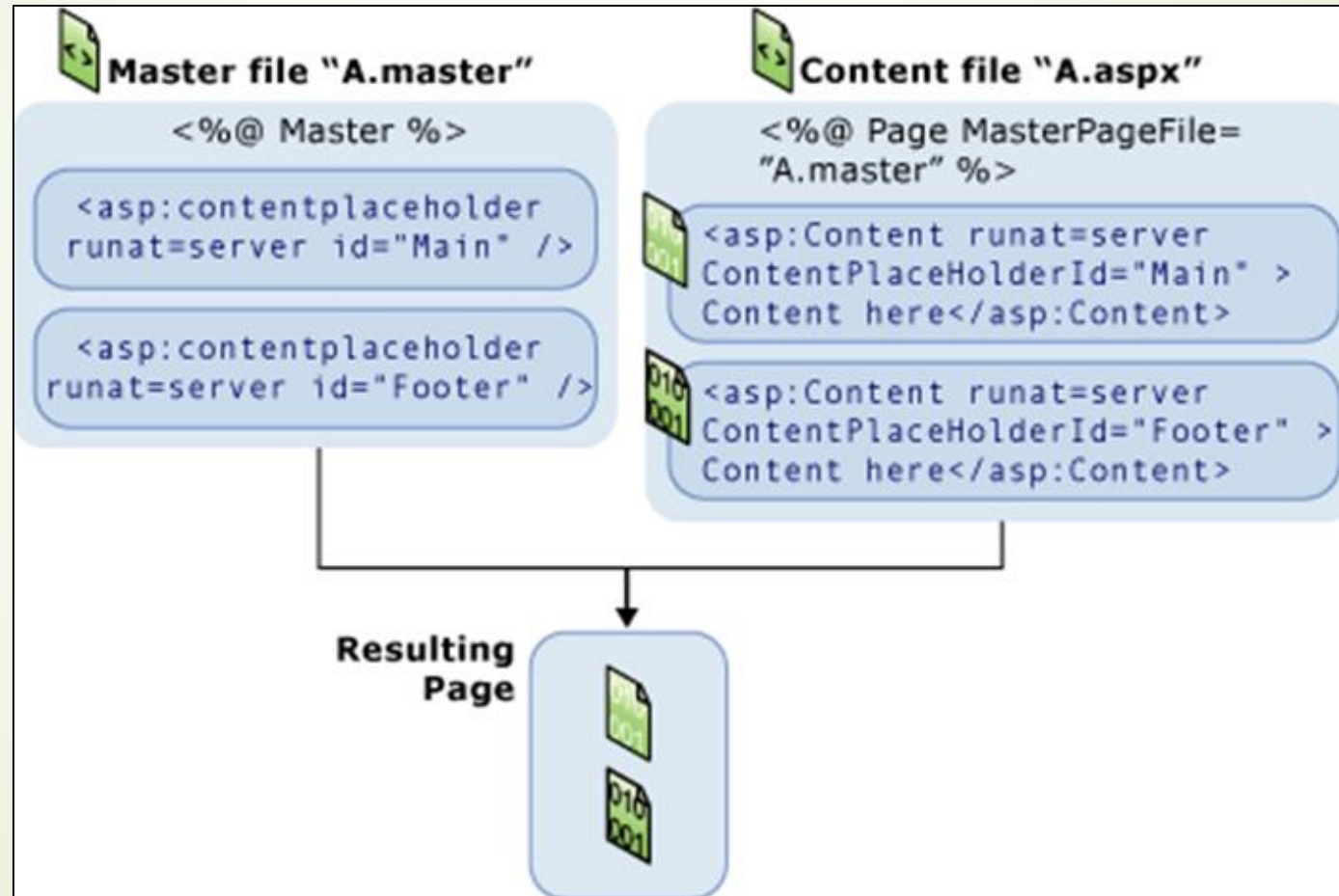
- ASP.NET master pages allow you to create a consistent layout for the pages in your application.
- A single master page defines the look and feel and standard behavior that you want for all of the pages (or a group of pages) in your application.
- You can then create individual content pages that contain the content you want to display.
- When users request the content pages, they merge with the master page to produce output that combines the layout of the master page with the content from the content page.

Master Page

- Master pages actually consist of two pieces, the master page itself and one or more content pages.
- **Master Pages**
 - A master page is an ASP.NET file with the extension **.master** (for example, **MySite.master**) with a predefined layout that can include static text, HTML elements, server controls and one or more content place holders.
 - The master page is identified by a special @ Master directive that replaces the @ Page directive that is used for ordinary .aspx pages.
- **Replaceable Content Placeholders**
 - In addition to static text and controls that will appear on all pages, the master page also includes one or more ContentPlaceHolder controls.
 - These placeholder controls define regions where replaceable content will appear.

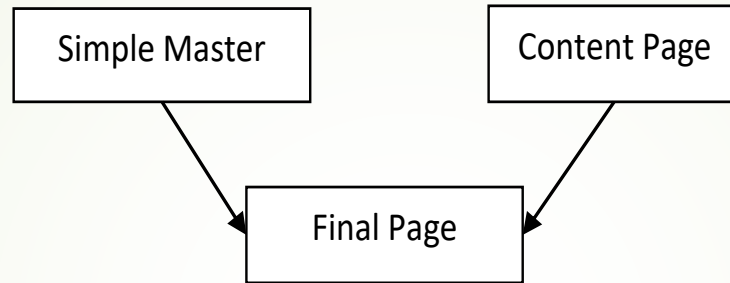
Master Page

Content Page

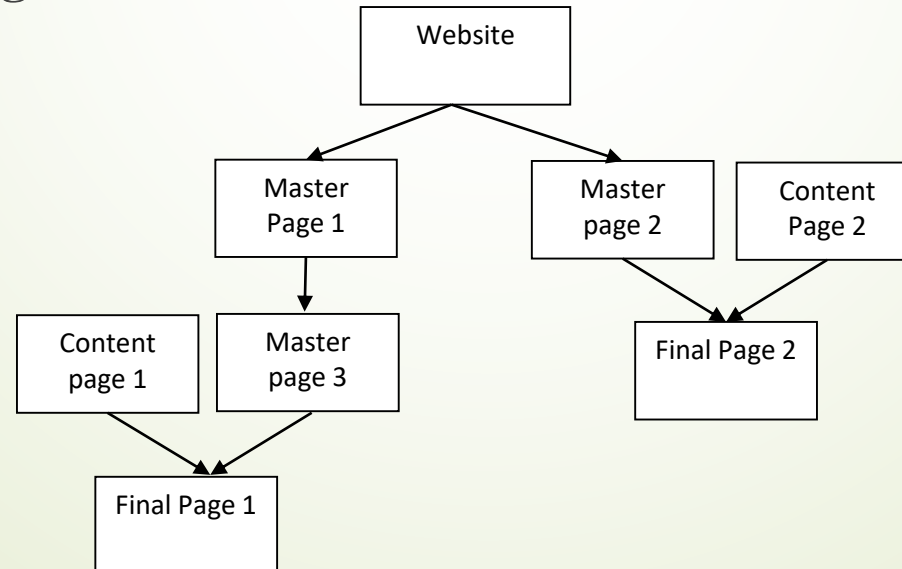


Master Page

Simple Master Page



Nested Master Page



Master Page

- **Advantages of Master Page**
- It allows you to centralize **the common functionality** of your pages so that you can make updates in one place only.
- It makes easy to create one set of controls and code and apply the results to a set of pages. For example, **you can use controls on the master page to create a menu that applies to all pages.**
- It will give you a fine-grained control over the **layout of the final page by allowing you to control how the placeholder controls are rendered.**
- It provides an object model that allows you to **customize the master page from individual content pages.**

Themes, Skins and CSS

- A theme is a collection of property settings that allow you to define the look of pages and controls, and then apply the look consistently across all the pages & entire Web application.
- **Themes and Control Skins**
- Themes are made up of a set of elements:
 - **skins, cascading style sheets (CSS), images, and other resources.** At a minimum, a theme will contain skins.
- **Themes are defined in special directories in your Web site or on your Web server.**

Themes, Skins and CSS

- A page theme is a theme folder with control skins, style sheets, graphics files and other resources created as a subfolder of the \App_Themes folder in your Web site.
- Each theme is a different subfolder of the \App_Themes folder.
- The following example shows a typical page theme, defining two themes named BlueTheme and PinkTheme.

```
MyWebSite
  App_Themes
    BlueTheme
      Controls.skin
      BlueTheme.css
    PinkTheme
      Controls.skin
      PinkTheme.css
```

Themes, Skins and CSS

- **How to Apply Theme in a Web Page?**
- You can define the name of the theme and wish to make use of in the Page directives of the aspx page.
- This can be done as below:

```
<%@ Page Theme="Theme1" Language="C#" AutoEventWireup="true"  
CodeFile="Default.aspx.cs" Inherits="_Default"  
EnableTheming="true"%>
```

Themes, Skins and CSS

➤ Skins

- A skin file has the file name extension .skin and contains property settings for individual controls such as Button, Label, TextBox, or Calendar controls.
- Control skin settings are like the control markup itself, but contain only the properties you want to set as part of the theme.
- For example, the following is a control skin for a Button control:
- `<asp:button runat="server" BackColor="lightblue" ForeColor="black" />`
- You create .skin files in the Theme folder.
- A .skin file can contain one or more control skins for one or more control types.
- You can define skins in a separate file for each control or define all the skins for a theme in a single file.

Themes, Skins and CSS

- There are two Types of Skins:
 - Default Skin
 - Named Skin
- A default skin automatically applies to all controls of the same type when a theme is applied to a page.
- A control skin is a default skin if it does not have a SkinID attribute.
- For example, if you create a default skin for a Calendar control, the control skin applies to all Calendar controls on pages that use the theme.
- (Default skins are matched exactly by control type, so that a Button control skin applies to all Button controls, but not to LinkButton controls or to controls that derive from the Button object.)

Themes, Skins and CSS

- A named skin is a control skin with a SkinID property set. Named skins do not automatically apply to controls by type.
- Instead, you explicitly apply a named skin to a control by setting the control's SkinID property.
- Creating named skins allows you to set different skins for different instances of the same control in an application.

Themes, Skins and CSS

- **Cascading Style Sheets (CSS)**
- A theme can also includes a cascading style sheet (.css file).
- When you put a .css file in the theme folder, the style sheet is applied automatically as part of the theme.
- You define a style sheet using the file name extension .css in the theme folder.

Configuration Overview

- The behavior of an ASP.NET application is affected by different settings in the configuration files:
 - machine.config
 - web.config
- The machine.config file contains default and the machine-specific value for all supported settings.
- **The machine settings are controlled by the system administrator and applications are generally not given access to this file.**
- An application however, can override the default values by creating **web.config files in its roots folder**. The **web.config file is a subset of the machine.config file**.
- Visual Studio generates a default web.config file for each project. An application can execute without a web.config file, however, we cannot debug an application without a web.config file.

Configuration Overview

- web.config file is an XML-based configuration file used in ASP.NET-based applications to manage various settings that are concerned with the configuration of our website.
- Hierarchy of the Web.config file.

```
<configuration>
  <configSections>
    <sectionGroup></sectionGroup>
  </configSections>

  <system.web></system.web>

  <connectionStrings></connectionStrings>
  <appSettings></appSettings>
  ....
</configuration>
```

Configuration Overview

- There are number of important settings that can be stored in the configuration file. Some of the most frequently used configurations, stored conveniently inside **web.config** file are:
 - Database Connections
 - Caching Settings
 - Session States
 - Error Handling
 - Security
- Benefits of XML-based Configuration files
 - ASP.NET Configuration system is **extensible and application specific** information can be stored and retrieved easily. It is human readable.
 - **We need not restart the web server when the settings are changed in configuration file.** ASP.NET automatically detects the changes and applies them to the running ASP.NET application.
 - We can use any standard text editor or XML parser to create and edit ASP.NET configuration files.

Configuration Overview

➤ Global.asax

- The Global.asax is also known as the ASP.NET application file and is used to serve **application-level and session-level events**.
- It allows us to write code that response to global application events raised **by ASP.NET or by HttpModules**.
- These events fire at various points during the lifetime of a **web application, including when the application domain is first created**.
- The Global.asax file resides in the root directory of an ASP.NET-based application
- At run time, Global.asax is parsed and compiled into a dynamically generated .NET Framework class derived from the `HttpApplication` base class.
- The Global.asax file is optional. If you do not define the file, the ASP.NET page framework assumes that you have not defined any application or session event handlers

Configuration Overview

Basic Application Events of Global.asax

Event Handling Method	Description
Application_Start()	event occurs when the application starts, which is the first time it receives a request from any user. It doesn't occur on subsequent requests. This event is commonly used to create or cache some initial information that will be reused later.
Application_End()	event occurs when the application is shutting down, generally because the web server has restarted. You can insert cleanup code here.
Application_BeginRequest()	event occurs with each request the application receives, just before the page code is executed.
Application_EndRequest()	event occurs with each request the application receives, just after the page code is executed.
Session_Start()	event occurs whenever a new user request is received and a session is started.
Session_End()	event occurs when a session times out or is programmatically ended. This event is only raised if you are using in-process session state storage.

.Net State Management

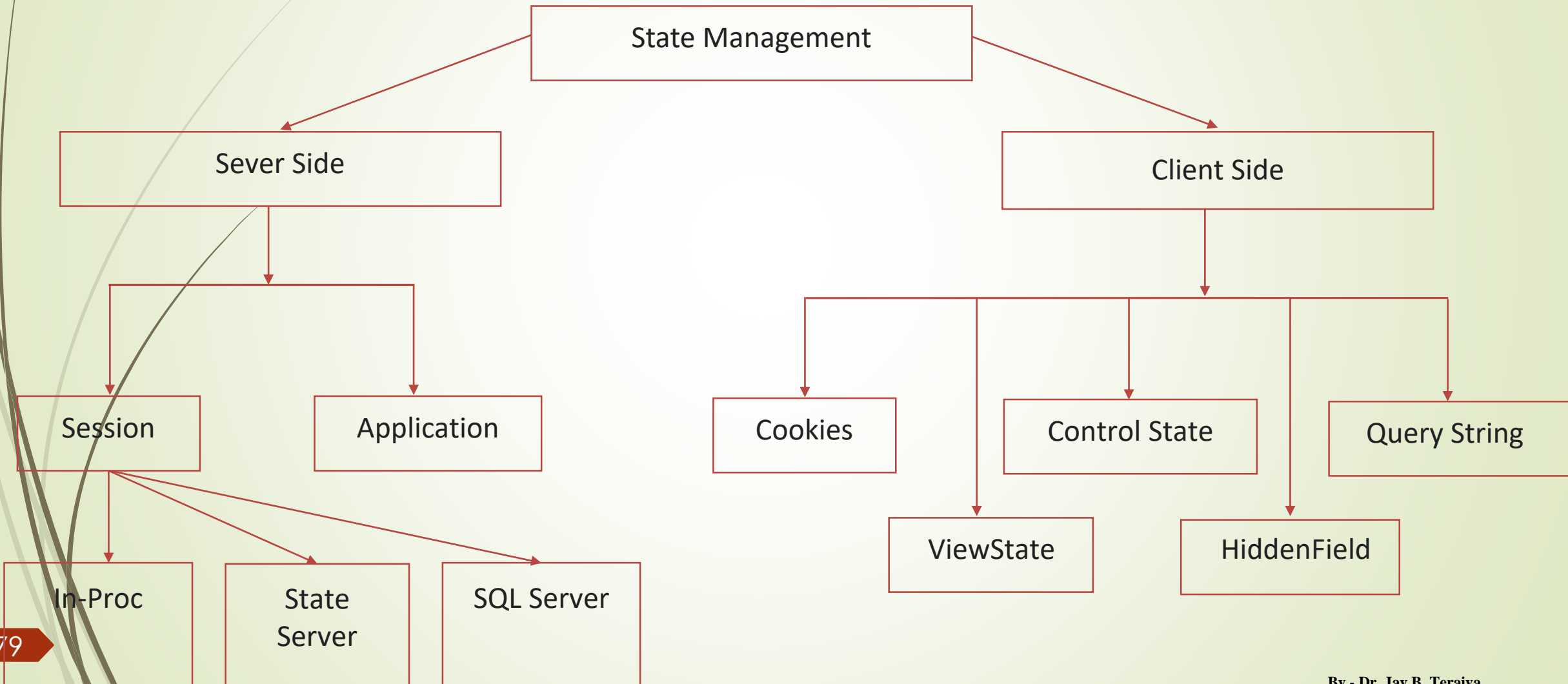
- Whenever we visit a website, our browser communicates with the respective server using the HTTP or HTTPs protocol.
- As HTTP is a stateless protocol. It just cleans up or we can say removes all the resources/references that were serving a specific request in the past.
- These resources can be:
 - Objects
 - Allocated Memory
 - Sessions ID's
 - Some URL info
 - and so on.
- In traditional Web programming, this would typically mean that all information associated with the page and the controls on the page would be lost with each round trip.
- For example, if a user enters information into a text box, that information would be lost in the round trip from the browser or client device to the server.

.Net State Management

- To overcome this inherent limitation of traditional Web programming, ASP.NET includes several options that help you preserve data on both a per-page basis and an application-wide basis.
- The state of a web application helps you to store the runtime changes that have been made to the web application.
- For example, a user selects and saves some products in the shopping cart of an online shopping websites.
- For that you have to store it to state, otherwise the changes are discarded.

.Net State Management

State Management Techniques



.Net State Management

➤ Client-Side State Management

- In Client-Side State Management, the state of the page is maintained at the client side.
- The following are the various techniques to do Client-Side State Management.
 - View State
 - Cookies
 - Query String
- In Client-Side State Management, data is stored at the client side and sent to the browser every time.
- This increases bandwidth usage.
- If the size of data is greater then the application will be less responsive. Performance issues would occur.

.Net State Management

➤ Server-Side State Management

- In the Server-Side State Management, the state of the page is maintained at the server side.
- The following are the various techniques to do Server-Side State Management:
 - Application State
 - Session State
 - SQL Server database
- Server-Side State Management provides better security.
- The state is saved on the server and therefore isn't delivered to the client.
- The server side reduces the traffic to and from the client because the data is not sent to the browser.

Caching in ASP.Net

- **What is Caching?**
- **Caching is a technique of storing frequently used data/information in memory**, so that, when the same data/information is needed next time, it could be directly retrieved from the memory instead of being generated by the application.
- Caching is **extremely important for performance boosting in ASP.NET**, as the pages and controls are dynamically generated here. It is especially important for data related transactions, as these are expensive in terms of response time.
- Caching places frequently used data in quickly accessed media such as the random access memory of the computer. The ASP.NET runtime includes a key-value map of CLR objects called cache. This resides with the application and is available via the **HttpContext** and **System.Web.UI.Page**.

Caching in ASP.Net

- **ASP.NET** provides the following different types of caching:
- **Output Caching** : Output cache stores a copy of the **finally rendered HTML pages or part of pages sent to the client**. When the next client requests for this page, instead of regenerating the page, a cached copy of the page is sent, thus saving time.
- **Data Caching** : Data caching means **caching data from a data source**. As long as the cache is not expired, a request for the data will be fulfilled from the cache. When the cache is expired, fresh data is obtained by the data source and the cache is refilled.
- **Object Caching** : Object caching is caching the **objects on a page, such as data-bound controls**. The cached data is stored in server memory.

Caching in ASP.Net

- **The data will not be available in the following cases:**
 - If its lifetime expires,
 - If the application releases its memory,
 - If caching does not take place for some reason.
- You can access items in the cache using an indexer and may control the lifetime of objects in the cache and set up links between the cached objects and their physical sources.

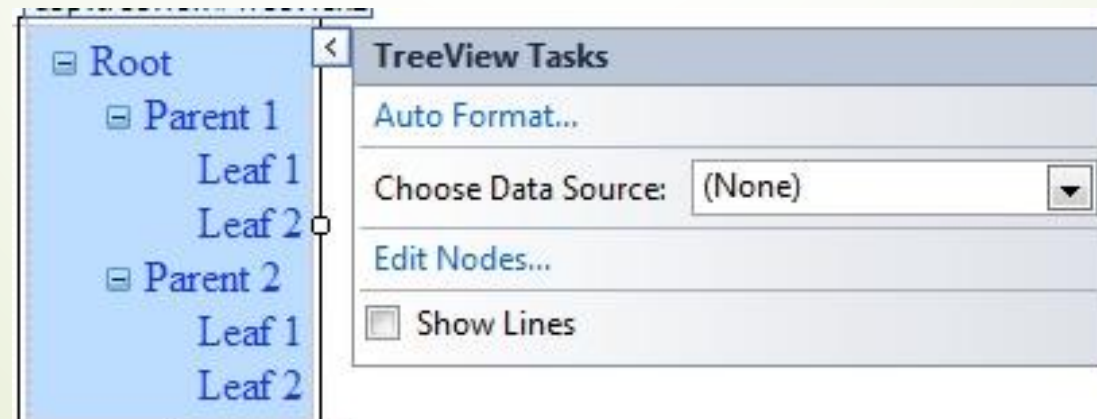
Navigation Control in ASP.NET

- Navigation controls are very important for websites. Navigation controls are basically **used to navigate the user through webpage**. It is more helpful for making the navigation of pages easier. There are three controls in ASP.NET, which are used for Navigation on the webpage.
 - TreeView control
 - Menu Control
 - SiteMapPath control
- There are some Namespaces, which are used for above Navigation controls which are given below:
 - `Using.System.Web.UI.WebControls.TreeView ;`
 - `Using.System.Web.UI.WebControls.Menu ;`
 - `Using.System.Web.UI.WebControls.SiteMapPath ;`

Navigation Control in ASP.NET

➤ The TreeView Control

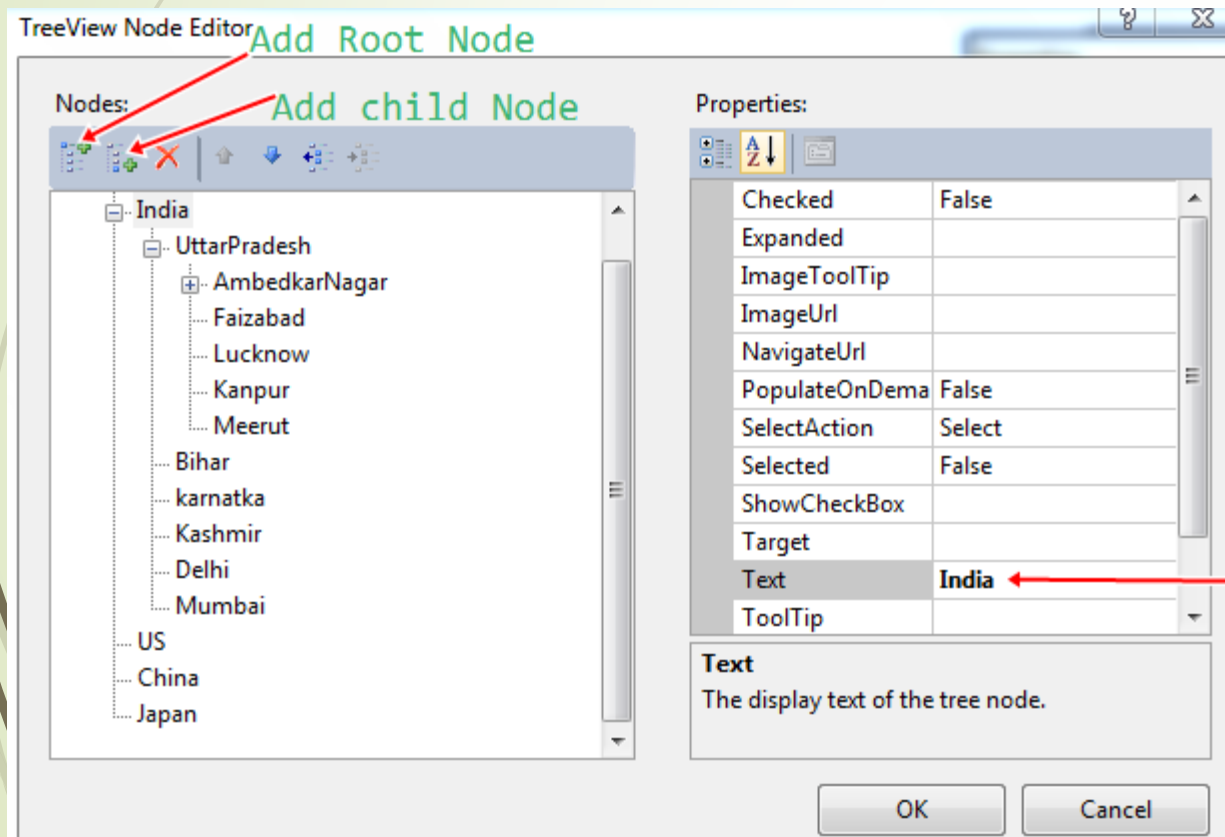
- The TreeView control is used for logically displaying the data in a hierarchical structure. We can use this navigation control for displaying the files and folders on the webpage. We can easily display the XML document, WebSite Map files and Database records in a tree structure.
- First open your visual studio → File → New → Website → Select ASP.NET Empty Website. open solution explorer → Add New Web Form → Drag and Drop TreeView control from Toolbox as shown below:



Navigation Control in ASP.NET

► The TreeView Control

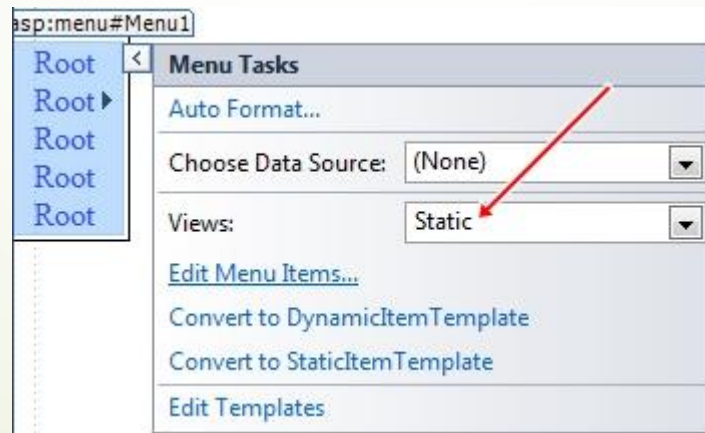
- Now go properties of TreeView control → Click Nodes → Add Root and child Node as shown below:



Navigation Control in ASP.NET

➤ The Menu Control

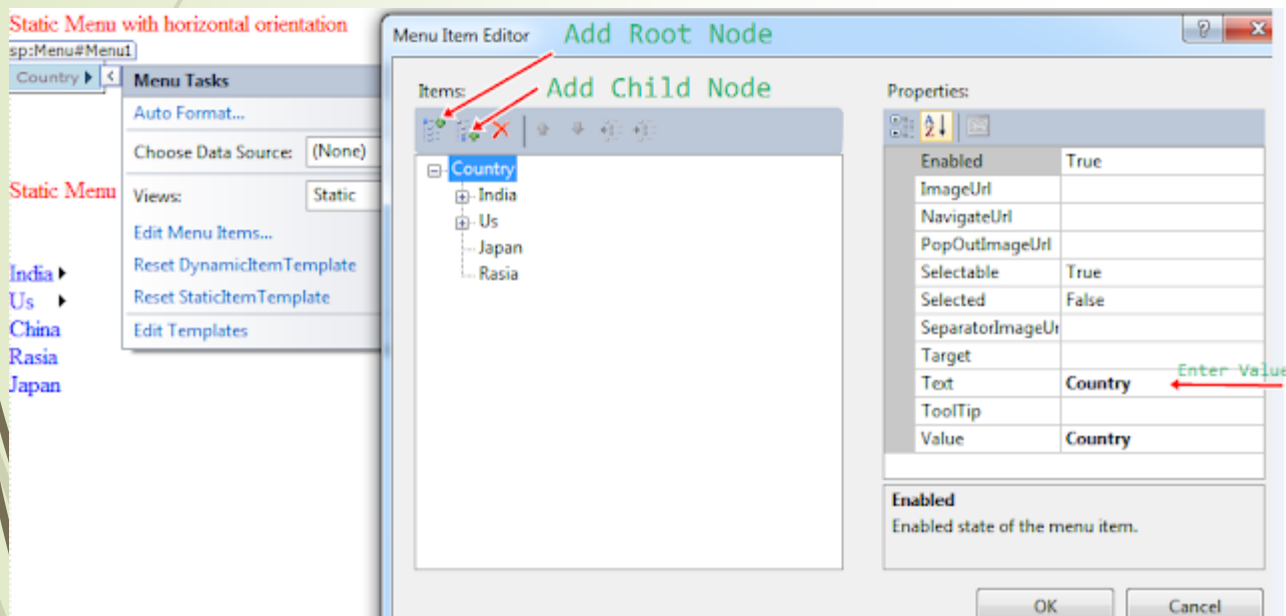
- The menu control is a Navigation control, which is used to display the site navigation information. This can be used to display the site data structure vertically and horizontally.
- It can be used as a binding control as TreeView control. Means we can easily bind the XML and SiteMap data in menu control.
- First Add a New Web Form in solution Explorer → drag and drop menu control on the Form → now select Views = static as shown below:



Navigation Control in ASP.NET

➤ The Menu Control

➤ Now click Edit Menu Items... → Add parent and child nodes as shown below:



Navigation Control in ASP.NET

- **The SiteMapPath Control**
- The SiteMapPath control is also used to display the Navigation information on the site. It display the current page's context within the entire structure of a website.
- There are some steps to implement the SiteMapPath control on the web page. Which are given below:
- **Step 1:** First open your visual studio → File → New → Website → Select ASP.NET Empty Website → Open solution Explorer → Add a Web Form (SiteMap.aspx) → Now again Add Site Map File in solution Explorer → **open web.sitemap file** → write the following codes , which are given below:

Navigation Control in ASP.NET

➤ The SiteMapPath Control

➤ `<?xml version="1.0" encoding="utf-8" ?>`

`<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >`

`<siteMapNode url="SiteMap.aspx" title="Country" description="">`

`<siteMapNode url="page1.aspx" title="India" description="" />`

`<siteMapNode url="page2.aspx" title="China" description="" />`

`<siteMapNode url="page3.aspx" title="US" description="" />`

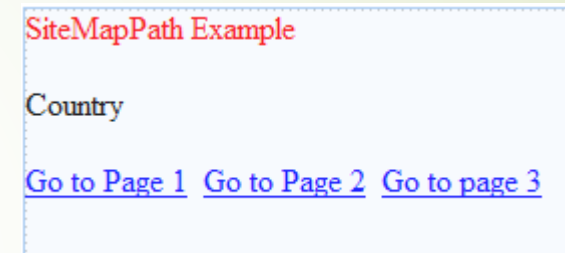
`</siteMapNode>`

`</siteMap>`

Navigation Control in ASP.NET

➤ The SiteMapPath Control

- **Step 2:** Now drag and drop **SiteMapPath** control on the web Form (SiteMap.aspx) → Now drag and drop **HyperLink** control on the Form as shown below:



- **Step 3:** Now Add three more Web Form (page1.aspx ,page2.aspx, page3.aspx) in Solution Explorer → Go properties of HyperLink Button control → set NavigateUrl → Write Text Information ,as shown below:

