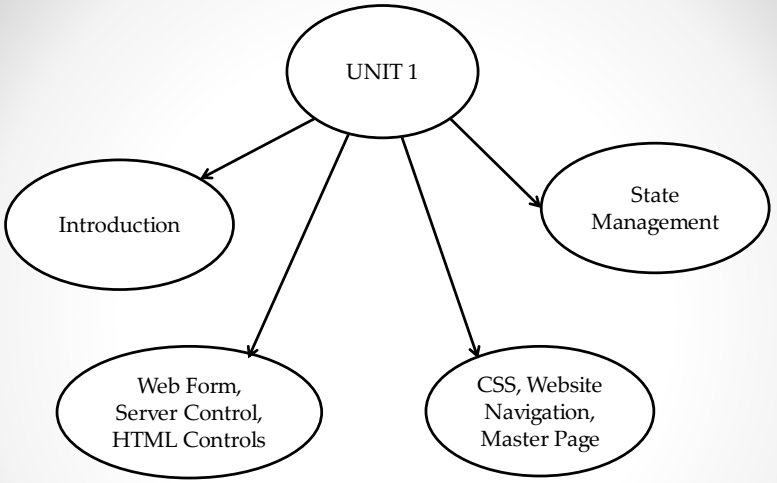
UNIT 1 Introduction of Web Development



Seven Pillars of ASP.NET

1. ***ASP.NET is Integrated with the .NET Framework.***
2. ***ASP.NET is Compiled, not Interpreted.***
3. ***ASP.NET is Multilanguage.***
4. ***ASP.NET is hosted by the Common Language Runtime.***
5. ***ASP.NET is Object-Oriented.***
6. ***ASP.NET Supports all Browser.***
7. ***ASP.NET is easy to Deploy and Configure.***

1. *ASP.NET is Integrated with the .NET Framework*

- The .NET Framework is divided into an almost particular collection of functional parts, with tens of thousands of type.
  - Namespace
    - Number of classes in the .NET Framework is grouped into a logical, hierarchical container is called ***namespace***.
  - Class library
    - .NET namespace offers functionality for nearly every aspects of distributed development from message queuing to security is a ***class library***.

2. *ASP.NET is Compiled, not Interpreted.*

- .NET application go through two stages of compilation.
  - First stage ,the C# code you write is complied into an intermediated language called Microsoft Intermediate Language (MSIL/IL).
    - it is fundamental reason that .NET can be language-interdependent.
      - first compilation step may happen automatically when the page is first requested, or you can perform it in advance (a process known as ***precompiling***). The compiled file with IL code is an ***assembly***.
    - All .NET languages are compiled into virtually identical IL code.
      - second step of compilation happens just before the page is actually executed. At this point, the IL code is compiled into low-level native machine code. This stage is known as **just-in-time (JIT) compilation**, and it takes place in the same way for all .NET applications.

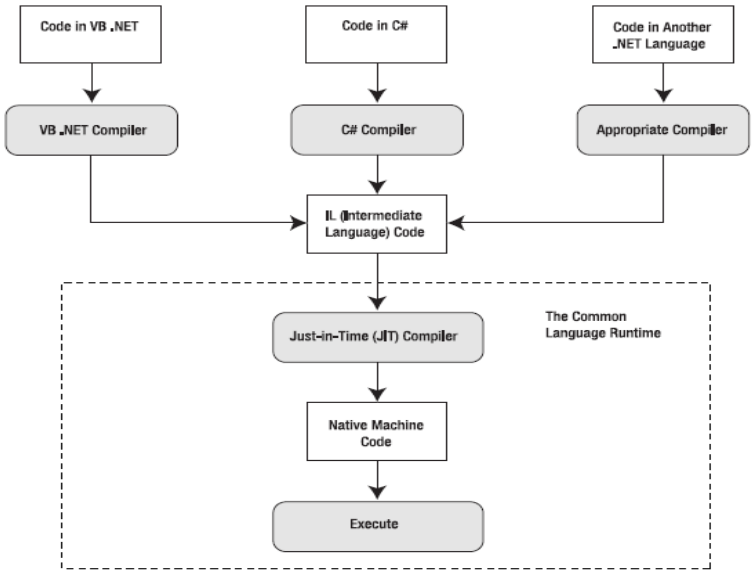


Figure 1-1. Compilation in an ASP.NET web page

3: ASP.NET Is Multilanguage

- You may be use one language over another when you develop an application, that choice won't determine what you can accomplish with your web applications. Because what ever language used for writing code are compiled into IL.
- IL is a stepping stone for every ***managed application***. (A managed application is any application that's written for .NET and executes inside the managed environment of the CLR.)
- Although different compilers can sometimes introduce their own optimizations, as a general rule of thumb no .NET language outperforms any other .NET language, because they all share the same common infrastructure. This infrastructure is formalized in the CLS (Common Language Specification)

Common Language Specification

- The CLR expects all objects to follow to a specific set of rules so that they can interact. The CLS is this set of rules.
- The CLS defines many laws that all languages must follow, such as primitive types, method overloading, and so on. Any compiler that generates IL code to be executed in the CLR must adhere to all rules governed within the CLS

4. ASP.NET Is Hosted by the Common Language Runtime

- ASP.NET engine runs inside the runtime environment of the CLR. The whole of the .NET Framework—that is, all namespaces, applications, and classes—is referred to as ***managed code***.
- Benefits of CLR :-
  1. **Automatic memory management and garbage collection.**
  2. **Type safety.**
  3. **Extensible metadata.**
  4. **Structured error handling.**
  5. **Multithreading.**

1. Automatic memory management and garbage collection

- Every time application instantiates a reference-type object, the CLR allocates space on the managed heap for that object. However, you never need to clear this memory manually.
- As soon as your reference to an object goes out of scope (or your application ends), the object becomes available for garbage collection.
- The garbage collector runs periodically inside the CLR, automatically reclaiming unused memory for inaccessible objects.

2. Type safety

- When we compile an application, .NET adds information to your assembly that indicates details such as the available classes, their members, their data types, and so on. As a result, other applications can use them without requiring additional support files, and the compiler can verify that every call is valid at runtime. This extra layer of safety completely eliminates whole categories of low-level errors.

3. Extensible metadata

- The information about classes and members is only one of the types of metadata that .NET stores in a compiled assembly.
- *Metadata* describes your code and allows you to provide additional information to the runtime or other services. For example, this metadata might tell a debugger how to trace your code, or it might tell Visual Studio how to display a custom control at design time. You could also use metadata to enable other runtime services, such as transactions or object pooling.

4. Structured error handling

- .NET languages offer structured exception handling, which allows you to organize your error-handling code logically and concisely.
- You can create separate blocks to deal with different types of errors. You can also nest exception handlers multiple layers deep.

5. Multithreading

- The CLR provides a group of threads that various classes can use. For example, you can call methods, read files, or communicate with web services asynchronously, without needing to explicitly create new threads.

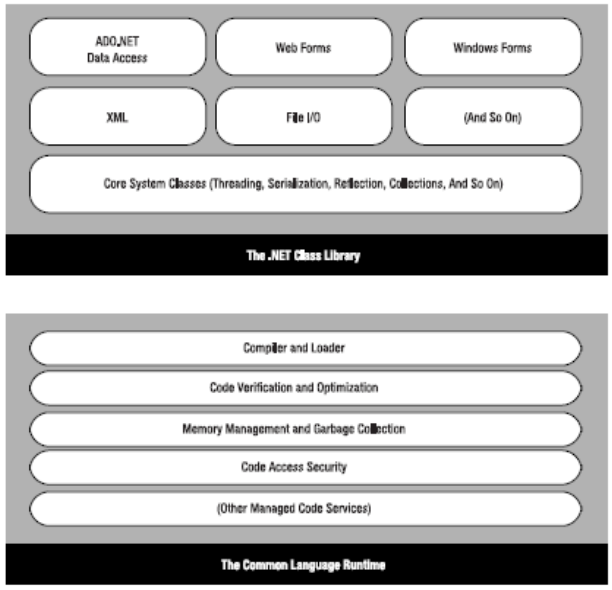


Figure 1-2. The CLR and the .NET Framework

5. ASP.NET Is Object-Oriented

- Developers can manipulate control objects programmatically using code to customize their appearance, provide data to display, and even react to events.
- Ex.

```
<input type="text" id="myText" runat="server" />
void Page_Load(object sender, EventArgs e)
{
myText.Value = "Hello World!";
}
```

6. ASP.NET Supports all Browsers

- One of the greatest challenges web developers face is the wide variety of browsers they need to support.
- Different browsers, versions, and configurations differ in their support of XHTML, CSS, and JavaScript.
- ASP.NET addresses this problem in a extraordinarily intelligent way. Although you can retrieve information about the client browser and its capabilities in an ASP.NET page,

7. ASP.NET Is Easy to Deploy and Configure

- Every installation of the .NET Framework provides the same core classes. As a result, deploying an ASP.NET application is relatively simple.

Evaluation of ASP.NET

- ASP.NET 1.0 and 1.1
  - When ASP.NET 1.0 first hit the scene, its core idea was a model of web page design called ***web forms.***
  - Web form model is simply an abstraction that models your page as a combination of objects.
  - When a browser requests a specific page, ASP.NET instantiates the page object, and then creates objects for all the ASP.NET controls inside that page. The page and its control go through a sequence of life-cycle events, and then—when the page processing is finished-they render the final HTML and are released from memory

Con..

- ASP.NET 2.0
  - It's a testament to the good design of ASP.NET 1.0 and 1.1 that few of the changes introduced in ASP.NET 2.0 were fixes for existing features.
  - Instead, ASP.NET 2.0 kept the same core abstraction (the web form model) and concentrated on adding new, higher-level features. Some of the highlights include:
    - **Master pages**
      - Master pages are reusable page templates.
    - **Themes**
      - Themes allow you to define a standardized set of appearance characteristics for web controls.
    - **Navigation**
      - ASP.NET's navigation framework includes a mechanism for defining site maps that describe the logical arrangement of pages in a website.

Con..

- **Security and membership**
  - ASP.NET 2.0 added a slew of security-related features, including automatic support for storing user credentials, a role-based authorization feature, and prebuilt security controls for common tasks like logging in, registering, and retrieving a forgotten password.
- **Data source controls**
  - The data source control model allows you to define how your page interacts with a data source *declaratively* in your markup, rather than having to write the equivalent data access code by hand.
- **Web parts**
  - Web parts provide a prebuilt portal framework complete with a flow-based layout, configurable views, and even drag-and-drop support.
- **Profiles**
  - Profiles allow you to store user-specific information in a database without writing any database code.

Con..

- ASP.NET 3.5
- Microsoft used the name .NET Framework 3.0 to release new technologies
  - WPF (Windows Presentation Foundation)
    - A glassy new user interface technology for building rich clients
  - WCF (Windows Communication Foundation)
    - a technology for building message-oriented services
  - WF (Windows Workflow Foundation)
    - A technology that allows you to model a complex business process as a series of actions (optionally using a visual flowchart-like designer).
- However, the .NET Framework 3.0 doesn't include a new version of the CLR or ASP.NET. Instead, the next release of ASP.NET was rolled into the .NET Framework 3.5.

Con..

- .NET Framework 3.5 has new features are concentrated in two areas: LINQ and Ajax.
- LINQ
  - LINQ (Language Integrated Query) is a set of extensions for the C# and Visual Basic languages. It allows you to write C# or Visual Basic code that manipulates in-memory data in much the same way you query a database.
- ASP.NET AJAX
  - Because traditional ASP.NET code does all its work on the web server, every time an action occurs in the page the browser needs to post some data to the server, get a new copy of the page, and refresh the display. This process, though fast, introduces a noticeable flicker.
  - It also takes enough time that it isn't practical to respond to events that fire frequently, such as mouse movements or key presses.
  - Web developers work around these sorts of limitations using JavaScript, the only broadly supported client-side scripting language. In ASP.NET, many of the most powerful controls use a healthy bit of JavaScript.

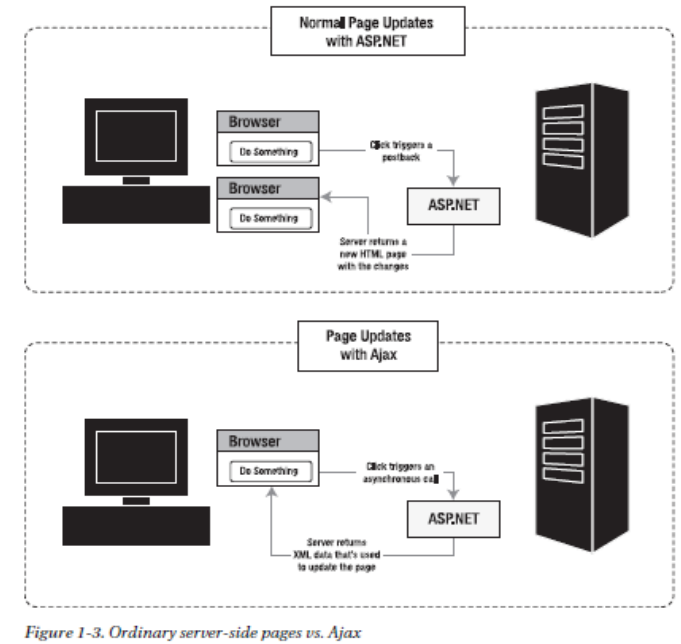


Figure 1-3. Ordinary server-side pages vs. Ajax

Con..

- Ajax and similar client-side scripting techniques are nothing new, but in recent years they've begun to play an increasingly important role in web development. One of the reasons is that the XMLHttpRequest object—the plumbing that's required to support asynchronous client requests—is now present in the majority of modern browsers, including the following:
- • Internet Explorer 5 and newer
- • Netscape 7 and newer
- • Opera 7.6 and newer
- • Safari 1.2 and newer
- • Firefox (any version)
- • Google Chrome (any version)

Con..

- ASP.NET 4
  - ASP.NET continues to plug in new enhancements and refinements. The most significant ones include :
    - **Consistent XHTML rendering**
    - **Updated browser detection**
    - **Session state compression**
    - **Opt-in view state**
    - **Extensible caching**
    - **The Chart control**
    - **Revamped Visual Studio**
    - **Routing**
    - **Better deployment tools**
    - **ASP.NET MVC**
    - **ASP.NET Dynamic Data**

ASP.NET MVC

- ASP.NET MVC (which stands for Model-View-Controller) offers a dramatically different way to build web pages than the standard web forms model.
- The core idea is that your application is separated into three logical parts.
  - The *model* includes the application-specific business code—for example, data-access logic and validation rules.
  - The *view* creates a suitable representation of the model by rendering it to HTML pages.
  - The *controller* coordinates the whole show, handling user interactions, updating the model, and passing the information to the view.

Con..

- Following points are consider in ASP.NET MVC
  - **Test-driven development**
  - **Control over HTML markup**
  - **Control over URLs**

ASP.NET Dynamic Data

- ASP.NET Dynamic Data is a scaffolding framework that allows you to quickly build a data-driven application.

Silverlight

- Silverlight is all about client code—quite simply, it allows you to create richer pages than you could with HTML, DHTML, and JavaScript alone.

Introducing Visual Studio

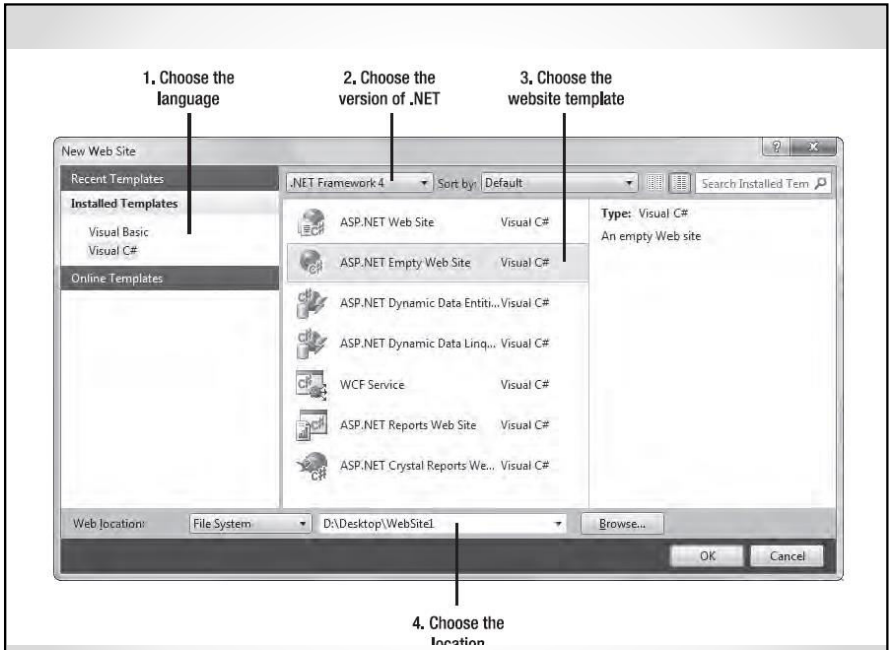
- Advantages of VS:-
  - An integrated web server
  - Multilanguage development
  - Less code to write
  - Intuitive coding style
  - Faster development time
  - Debugging

Website and Web Projects

- Project-based development
  - When you create a web project, Visual Studio generates a .csproj project file (assuming you're coding in C#) that records the files in your project and stores a few debugging settings. When you run a web project, Visual Studio compiles all your code into a single assembly before launching your web browser.
- Projectless development
  - Create a simple website without any project file. In this case, Visual Studio assumes that every file in the website directory (and its subdirectories) is part of your web application. In this scenario, Visual Studio doesn't need to precompile your code. Instead, ASP.NET compiles your website the first time you request a page. (Of course, you can use pre compilation to remove the first-request overhead for a deployed web application. Chapter 18 explains how.)

Creating a Project less Website

- File
  - New
    - Web site
      - Choose language
        - Choose web site template
          - Give website name
            - Location
              - ok

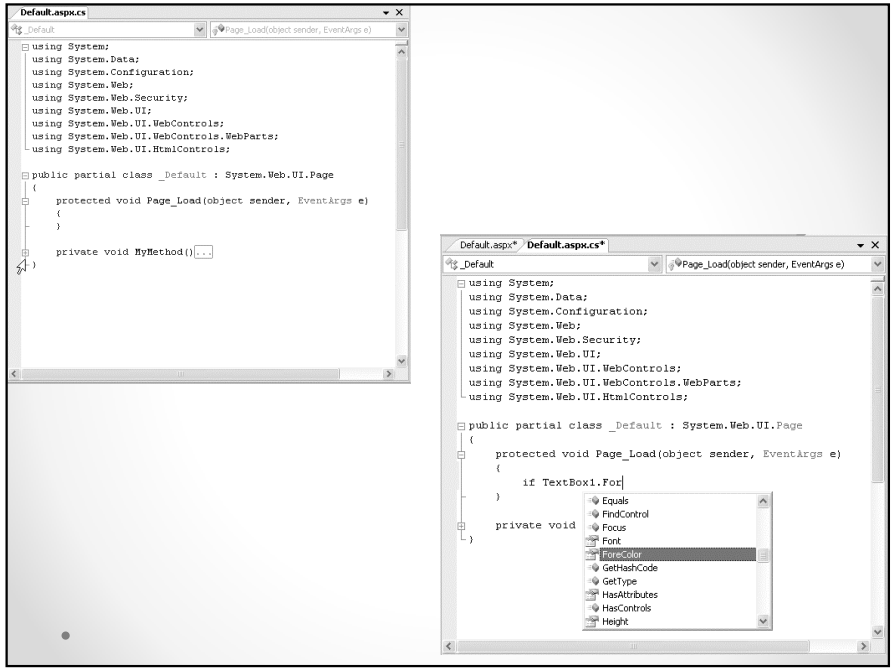


The Template

- ASP.NET Web Site
- ASP.NET Empty Web Site
- ASP.NET Dynamic Data Entites Web Site
- WCF Service
- ASP.NET Reports Web Site

IntelliSense and Outlining

- Outlining
  - Outlining allows Visual Studio to "collapse" a subroutine, block structure, or region to a single line. It allows you to see the code that interests you, while hiding unimportant code.
- Member List
  - Visual Studio makes it easy for you to interact with controls and classes. When you type a period (.) after a class or object name, Visual Studio pops up a list of available properties and methods



The Code Model

- Visual Studio supports two models for coding web pages:
- **Inline code:**
  - This model is the closest to traditional ASP.
  - All the code and HTML markup is stored in a single .aspx file.
  - The code is embedded in one or more script blocks. However, even though the code is in a script block, it doesn't lose IntelliSense or debugging support, and it doesn't need to be executed linearly from top to bottom (like classic ASP code).
  - Instead, you'll still react to control events and use subroutines.
  - This model is handy because it keeps everything in one neat package, and it's popular for coding simple web pages.

Con..

- **Code-behind:**
  - This model separates each ASP.NET web page into two files:
    - an .aspx markup file with the HTML and control tags,
    - a .cs code file with the source code for the page (assuming you're using C# as your web page programming language).
  - This model provides better organization, and separating the user interface from programmatic logic is keenly important when building complex pages.

Web Projects

- The advantage of projectless is that it's simple and straightforward. When you create a projectless website, you don't need to deploy any extra support files. Instead, every file in your web folder is automatically considered part of the web application.
- Projectless development remains popular for the following reasons:
  - **Projectless development simplifies deployment**
  - **Projectless development simplifies file management**
  - **Projectless development simplifies team collaboration**
  - **Projectless development simplifies debugging**
  - **Projectless development allows you to mix languages**

Creating a Web Project

- File
  - New
    - Web project
      - Choose language
        - Choose web project template
          - Give web project name
            - Location
              - ok

Web Project V/S Website

Difference	Web Project	Website
Compilation	Compiled by Visual Studio.	Visual Studio does not need to pre compiled it.
Code Model	It always follow code-behind model	It does not follow any model
Assembly references	assembly references in a web project are stored in a project file	all the assembly references are recorded in the web.config file,

Web Forms

- By using HTML Form Tag

The ASP.NET Event Model

- Classic ASP uses a linear processing model.
  - That means code on the page is processed from start to finish and is executed in order.
  - Because of this model, classic ASP developers need to write a considerable amount of code even for simple pages.
- A classic example is a web page that has three different submit buttons for three different operations. In this case, your script code has to carefully decide which button was clicked when the page is submitted and then execute the right action using conditional logic.

Con..

- ASP.NET provides a refreshing change with its *event-driven* model.
  - In this model, you add controls to a web form and then decide what events you want to respond to.
  - Each event handler is a discrete method, which keeps the page code tidy and organized. This model is nothing new, but until the advent of ASP.NET it has been the exclusive domain of windowed UI programming in rich client applications.

Con..

1. Your page runs for the first time. ASP.NET creates page and control objects, the initialization code executes, and then the page is rendered to HTML and returned to the client. The page objects are also released from server memory.
2. At some point, the user does something that triggers a postback, such as clicking a button. At this point, the page is submitted with all the form data.
3. ASP.NET intercepts the returned page and re-creates the page objects, taking care to return them to the state they were in the last time the page was sent to the client.
4. Next, ASP.NET checks what operation triggered the postback, and it raises the appropriate events (such as Button.Click), which your code can react to. Typically, at this point you'll perform some server-side operation (such as updating a database or reading data from a file) and then modify the control objects to display new information.
5. The modified page is rendered to HTML and returned to the client. The page objects are released from memory. If another postback occurs, ASP.NET repeats the process in steps 2 through 4.

Automatic Postbacks

- In ASP.NET, client actions happen on the client side, and server processing takes place on the web server. This means a certain amount of overhead is always involved in responding to an event. For this reason, events that fire rapidly (such as a mouse move event) are completely unfeasible in the world of ASP.NET.
- ASP.NET web controls extend with an automatic postback feature. With this feature, input controls can fire different events, and your server-side code can respond immediately.
- For example,
  - you can trigger a postback when the user clicks a check box, changes the selection in a list, or changes the text in a text box and then moves to another field.

Automatic Postbacks "Under the Hood"

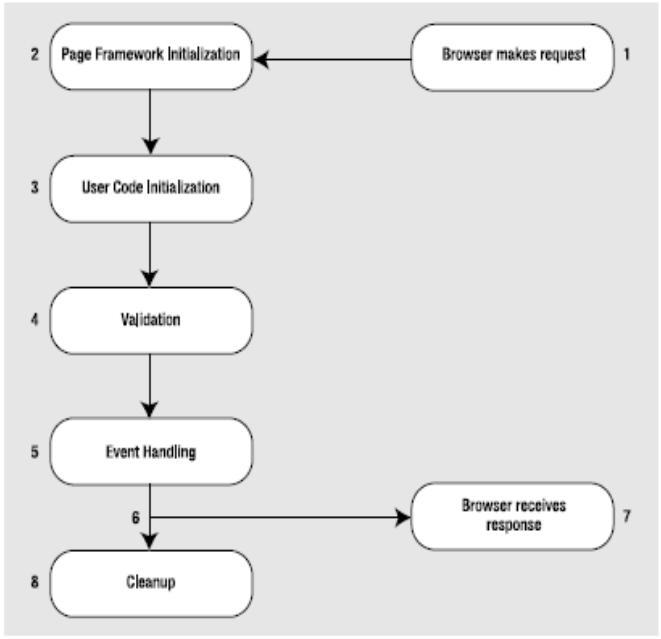
- To use automatic postback, you simply need to set the AutoPostBack property of a web control to true
  - the default is false, which ensures optimum performance if you don't need to react to a change event.
- When you do, ASP.NET uses the client-side abilities of JavaScript to bridge the gap between client-side and server-side code.
- Example,
  - if you create a web page that includes one or more web controls that are configured to use AutoPostBack, ASP.NET adds a JavaScript function to the rendered HTML page named __doPostBack().
  - When called, it triggers a postback, posting the page back to the web server with all the form information.
  - ASP.NET also adds two hidden input fields that the __doPostBack() function uses to pass information back to the server. This information consists of the ID of the control that raised the event and any additional information that might be relevant.

Con..

- any control that has its AutoPostBack property set to true is connected to the __doPostBack() function using the onclick or onchange attribute. These attributes indicate what action the browser should take in response to the client-side JavaScript events onclick and onchange.

Web Forms Processing Stages

- On the server side, processing an ASP.NET web form takes place in stages.At each stage, various events are raised.
- This allows your page to plug into the processing flow at any stage and respond however you would like.
- The following list shows the major stages in the process flow of an ASP.NET page:
  - Page framework initialization
  - User code initialization
  - Validation
  - Event handling
  - Automatic data binding
  - Cleanup



Page Framework Initialization

- This is the stage in which ASP.NET first creates the page. It generates all the controls you have defined with tags in the .aspx web page.
- In addition, if the page is not being requested for the first time (in other words, if it's a postback), ASP.NET deserializes the view state information and applies it to all the controls.
- At this stage, the **Page.Init** event fires. However, this event is rarely handled by the web page, because it's still too early to perform page initialization. That's because the control objects may not be created yet and because the view state information isn't loaded.

User Code Initialization

- At this stage of the processing, the **Page.Load** event is fired. Most web pages handle this event to perform any required initialization (such as filling in dynamic text or configuring controls).
- The Page.Load event *always* fires, regardless of whether the page is being requested for the first time or whether it is being requested as part of a postback.

Validation

- ASP.NET includes validation controls that can automatically validate other user input controls and display error messages.
- These controls fire after the page is loaded but before any other events take place.
- However, the validation controls are for the most part self-sufficient, which means you don't need to respond to the validation events.

Event Handling

- At this point, the page is fully loaded and validated. ASP.NET will now fire all the events that have taken place since the last postback. For the most part, ASP.NET events are of two types:
  - **Immediate response events:**
    - These include clicking a submit button or clicking some other button, image region, or link in a rich web control that triggers a postback by calling the __doPostBack() JavaScript function.
  - **Change events:**
    - These include changing the selection in a control or the text in a text box. These events fire immediately for web controls if AutoPostBack is set to true. Otherwise, they fire the next time the page is posted back.

Con..

- For example,
  - imagine you have a page with a submit button and a text box that doesn't post back automatically. You change the text in the text box and then click the submit button. At this point, ASP.NET raises all of the following events (in this order):
- Page.Init
- Page.Load
- TextBox.TextChanged
- Button.Click
- Page.PreRender
- Page.Unload

Cleanup

- At the end of its life cycle, the page is rendered to HTML. After the page has been rendered, the real cleanup begins, and the Page.Unload event is fired.
- At this point, the page objects are still available, but the final HTML is already rendered and can't be changed.
- Remember, the .NET Framework has a garbage collection service that runs periodically to release memory tied to objects that are no longer referenced.
- If you have any unmanaged resources to release, you should make sure you do this explicitly in the cleanup stage or, even better, before.
- When the garbage collector collects the page, the Page.Disposed event fires.

Server Control

- Types of server control
  - **HTML server controls**
  - **Web controls**
  - **Rich controls**
  - **Validation controls**
  - **Data controls**
  - **Navigation controls**
  - **Login controls**
  - **Web parts controls**
  - **ASP.NET AJAX controls**
  - **ASP.NET Dynamic Data controls**

CSS

- CSS (Cascading Style Sheets)
  - CSS provides a cross-platform solution for formatting web pages that works in conjunction with HTML or XHTML and is supported by virtually all modern browsers.
  - Stylesheets consist of rules. Each rule defines how a single ingredient in your web page should be formatted
- Creating a Stylesheet
  - Website
    - Add New
      - select Style Sheet, edit the filename
        - click OK

Con..

- Each rule name has two parts.
  - The portion before the period indicates the HTML element to which the rule applies.
  - The portion after the period is a unique name (called the CSS *class name*) that you choose to identify your rule.
- CSS class names are case-sensitive
- Types of CSS
  - Inline CSS
  - Internal CSS
  - External CSS

Applying CSS

- Inline CSS

<asp:TextBox ID="txt_adrs" runat="server“ Rows="5" Columns ="30" BackColor ="Red“ />

- Internal CSS

1. <link href="StyleSheet.css" rel="stylesheet" type="text/css" />

2

.txt {BackColor :"Red“ }

<asp:TextBox ID="txt_adrs" runat="server“ Rows="5" Columns ="30" CssClass =“txt “ />

- External CSS
  - Web link resource

CSS Selector

- The Element Selector

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
text-align: center;
color: red;
}
</style>
</head>
<body>
<p>Every paragraph will be affected by the style.</p>
<p id="para1">Me too!</p>
```

Con..

- The id Selector

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
text-align: center;
color: red;
}
</style>
</head>
<body>
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>
```

Con..

- The class Selector

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
text-align: center;
color: red;
}
</style>
</head>
<body>
<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>
```

Con..

```
<!DOCTYPE html>
        <html>
                <head>
                        <style>
                                p.center {
                                    text-align: center;
                                    color: red;
                                }
                        </style>
                </head>
                <body>
                        <h1 class="center">This heading will not be
affected</h1>
                        <p class="center">This paragraph will be
red and center-aligned.</p>
```

Con..

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
    text-align: center;
    color: red;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>
```

Themes

- The problem is that CSS rules are limited to a fixed set of style attributes. They allow you to reuse specific formatting details (fonts, borders, foreground and background colors, and so on), but they obviously can't control other aspects of ASP.NET controls.
- you might want to define part of the behavior of the control along with the formatting.

con..

- *Themes* fill this gap. Like CSS, themes allow you to define a set of style attributes that you can apply to controls in multiple pages.
- unlike CSS, themes aren't implemented by the browser. Instead, they're a native ASP.NET solution that's implemented on the server.
- themes don't replace styles, they have some features that CSS can't provide.

Con..

- Here are the key differences:
- **Themes are control-based, not HTML-based:**
  - As a result, themes allow you to define and reuse almost any control property. For example, themes allow you to specify a set of common node pictures and use them in numerous TreeView controls or to define a set of templates for multiple GridView controls. CSS is limited to style attributes that apply directly to HTML.
- **Themes are applied on the server:**
  - When a theme is applied to a page, the final styled page is sent to the user. When a stylesheet is used, the browser receives both the page and the style information and then combines them on the client side.

Con..

- **Themes can be applied through configuration files:**
  - This lets you apply a theme to an entire folde or your whole website without modifying a single web page.
- **Themes don't cascade in the same way as CSS:**
  - Essentially, if you specify a property in a theme *and* in the individual control, the value in the theme overwrites the property in the control. However, you have the choice of changing this behavior and giving precedence to the properties in the page, which makes themes behave more like stylesheets.

Creating Theme

- All themes are application-specific. To use a theme in a web application, you need to create a folder that defines it. You need to place this folder in a folder named App_Themes, which must be inside the toplevel directory for your web application.
- To add a theme to your project
  - select Website
    - Add New Item (or Project ➢ Add New Item)
      - choose Skin File.
- Visual Studio will warn you that skin files need to be placed in a subfolder of the App_Themes folder and will ask you if that's what you intended. If you choose Yes, Visual Studio will create a folder with the same name as your theme file

Applying Theme

- <%@ Page Language="C#" AutoEventWireup="true" ... **Theme="FunkyTheme"** %>
- <asp:Button ID="Button1" runat="server" ... **EnableTheming="false"** />
- <asp:Button runat="server" ForeColor="White" BackColor="DarkOrange"
- Font-Bold="True" **SkinID="Dramatic"** />

Web Site Navigation

- Navigation is a fundamental component of any website.
- **The MultiView and Wizard controls:**
  - These let you boil down a series of steps into a single page. With the help of these controls, you can combine several pages of work into one place, simplifying your navigation needs.
- **The site map model:**
  - This lets you define the navigation structure of your website and bind it directly to rich controls. You'll also learn how to extend this framework to support different types of controls and different site map storage locations.

Con..

- **The rich navigational controls:**
  - These include the TreeView and Menu. Although these controls aren't limited to navigation, they're an ideal match.

Pages with Multiple Views

- Most websites split tasks across several pages.
- For example,
  - if you want to add an item to your shopping cart and take it to the checkout in an e-commerce site, you'll need to jump from one page to another.
- In other situations, you might want to embed the code for several different pages inside a single page.
- For example,
  - you might want to provide several views of the same data (such as a grid-based view and a chart-based view) and allow the user to switch from one view to the other without leaving the page.
  - you might want to handle a small multistep task (such as supplying user information for an account sign-up process), without worrying about how to transfer the relevant information between web pages.

Con..

- In ASP.NET 1.x, the only way to model a page with multiple views was to add several Panel controls to a page so that each panel represents a single view or a single step.
- You could then set the Visible property of each Panel so that you see only one at a time.
- The problem with this approach is that,
  - it clutters your page with extra code for managing the panels.
  - it's not very robust—with a minor mistake, you can end up with two panels showing at the same time.
- With ASP.NET 4, there's no need to design your own multiple view system from scratch.
- Instead, you can use one of two higher-level controls that make these designs much easier—the MultiView and the Wizard

The MultiView Control

- The MultiView is the simpler of the two multiple view controls.
- It gives a way to declare multiple views and show only one at a time.
- It has no default user interface
  - you get only whatever HTML and controls you add.
- The MultiView is equivalent to the custom panel approach.

Con..

- Creating a MultiView is suitably straightforward. You add the <asp:MultiView> tag to your .aspx page file and then add one <asp:View> tag inside it for each separate view.
- The MultiView.ActiveViewIndex determines what view will be shown. This is the only view that's rendered in the page.
- The default ActiveViewIndex value is -1, which means no view is shown.

The Wizard Control

- The Wizard control is a more trendyversion of the MultiView control.
- It also supports showing one of several views at a time, but it includes a fair bit of built-in yet customizable behavior, including navigation buttons, a sidebar with step links, styles, and templates.

Con..

- Wizards represent a single task, and the user moves linearly through them, moving from the current step to the one immediately following it (or the one immediately preceding it in the case of a correction).
- The ASP.NET Wizard control also supports nonlinear navigation.
- By default, the Wizard control supplies navigation buttons and a sidebar with links for each step on the left.
- You can hide the sidebar by setting the Wizard.DisplaySideBar property to false.

Con..

- To create a wizard in ASP.NET, you simply define the steps and their content using <asp:WizardStep> tags. Each step takes a few basic pieces of information like,
  - Title
  - StepType
  - AllowReturn

Menu Control

- The Menu control is another rich control that supports hierarchical data.
- By <asp:Menu> you can add menu in the web page.
- To add menu list in that we need to add <menuItem>.

Master Page

- The ability to define a portion of a page separately and reuse it on multiple pages.
- The ability to create a locked-in layout that defines editable regions. Pages that reuse this template are then constrained to adding or modifying content in the allowed regions.
- The ability to allow some customization of the elements you reuse on each page.
- The ability to bind a page to a page template declaratively (with no code) or to bind to a page dynamically at runtime.
- The ability to design a page that uses a page template with a tool such as Visual Studio.

Con..

- ASP.NET defines two specialized types of pages:
  - master pages
    - A master page is a page template.
    - master pages can include *content placeholders*—defined regions that can be modified.
  - content pages
    - Each *content page* references a single master page and obtains its layout and content.
    - content page can add page-specific content in any of the placeholders. In other words, the content page fills in the missing pieces that the master page doesn't define.
- For example, in a typical website, a master page might include a fixed element such a header and a content placeholder for the rest of the page. The content page then acquires the header for free and supplies additional content.

Con..

- Create a Master Page
  - Select Website
    - Add New Item from the menu
      - Select Master Page, give it a filename (such as SiteTemplate.master)
        - click Add.

Con..

Master Page	Web Form
master page starts with a Master Directive	web forms start with the Page directive
master pages can use the ContentPlaceHolder control	Web form can not use the ContentPlaceHolder control

Con..

- When you create a new master page you start with a blank page that includes two ContentPlaceHolder controls.
  - One is defined in the <head> section
    - which gives content pages the ability to add page metadata, such as search keywords and stylesheet links.
  - The second defined in the <body> section,
    - represents the displayed content of the page.

Con..

- Create a content page
  - Select Website
    - Add New Item from the menu
      - Select web form
        - Master page
          - click Add.

State Management

- ASP.NET includes a variety of options for state management. You choose the right option based on following option :
  - the data you need to store
  - the length of time you want to store it
  - the scope of your data (whether it's limited to individual users or shared across multiple requests)
  - additional security
  - performance considerations.
- The different state management options in ASP.NET are complementary, which means you'll almost always use a combination of them in the same web application.

Con..

- Following options are includes in state management:
  - View State
  - Query String
  - Custom Cookies
  - Session State
  - Application State
  - Profiles
  - Caching

Con..

	View State	Query String	Custom Cookies
Allowed data types	All serializable .NET data types.	A limited amount of string data.	String data.
Storage location	A hidden field in the current web page.	The browser's URL string.	The client's computer (in memory or a small text file, depending on its lifetime settings).
Lifetime	Retained permanently for postbacks to a single page.	Lost when the user enters a new URL or closes the browser. However, can be stored and can persist between visits.	Lost when the user enters a new URL or closes the browser. However, can be stored and can persist between visits.
Scope	Limited to the current page.	Limited to the target page.	The whole ASP.NET application.

	View State	Query String	Custom Cookies
Security	Tamper-proof by default but easy to read. You can use the Page directive to enforce encryption.	Clearly visible and easy for the user to modify	Insecure and can be modified by the user.
Performance Implications	Storing a large amount of information will slow transmission but will not affect server performance.	None, because the amount of data is trivial.	None, because the amount of data is trivial.
Typical use	Page-specific settings	Sending a product ID from a catalog page to a details page.	Personalization preferences for a website

Con..

	Session State	Application State
Allowed data types	All serializable .NET data types. Nonserializable types are supported if you are using the default in-process state service.	All .NET data types.
Storage location	Server memory (by default), or a dedicated database, depending on the mode you choose.	Server memory
Lifetime	Times out after a predefined period (usually 20 minutes but can be altered globally or programmatically).	The lifetime of the application (typically, until the server is rebooted).

	Session State	Application State
Scope	The whole ASP.NET application.	The whole ASP.NET application. Unlike most other types of methods, application data is global to all users.
Security	Secure, because data is never transmitted to the client. However, subject to session hijacking if you don't use SSL.	Very secure, because data is stored on the server.
Performance Implications	Storing a large amount of information can slow down the server severely, especially if there are a large number of users at once, because each user will have a separate set of session data.	Storing a large amount of information can slow down the server, because this data will never time out and be removed.
Typical use	Store items in a shopping basket.	Store items in a shopping basket.

Con..

	Profiles	Caching
Allowed data types	All serializable .NET data types.	All .NET data types. Nonserializable types are supported if you create a custom profile.
Storage location	A back-end database.	Server memory
Lifetime	Permanent.	Depends on the expiration policy you set, but may possibly be released early if server memory becomes scarce.
Scope	The whole ASP.NET application. May also be accessed by other applications.	The same as application state (global to all users and all pages).

	Profiles	Caching
Security	Fairly secure, because although data is never transmitted, it is stored without encryption in a database that could be compromised.	Very secure, because the cached data is stored on the server.
Performance implications	Large amounts of data can be stored easily, but there may be a nontrivial overhead in retrieving and writing the data for each request.	Storing a large amount of information may force out other, more useful cached information. However, ASP.NET has the ability to remove items early to ensure optimum performance.
Typical use	Store customer account information.	Storing data retrieved from a database.

View State

- View state is used natively by the ASP.NET web controls.
- View state allows to retain their properties between postbacks.
- You can add your own data to the view state collection using a built-in page property called ViewState.
- You can store simple data types and your own custom objects.
- view state relies on a dictionary collection, where each item is indexed with a unique string name.

Con..

- Syntax:-
  - ViewState["Variable Name"] = Value;
- Example :-
  - ViewState["Counter"] = 1;
- Use :-
  - Int counter
  - Counter=(int) ViewState["Counter"]

Evaluating View State

- You need to store critical data that the user cannot be allowed to damage with.
- In this case, consider session state. Alternatively, consider using the countermeasures described in the next section. They aren't bulletproof, but they will greatly increase the effort an attacker would need in order to read or modify view state data.
- You need to store information that will be used by multiple pages. In this case, consider session state, cookies, or the query string.
- You need to store an extremely large amount of information, and you don't want to slow down page transmission times. In this case, consider using a database, or possibly session state.

Transferring Information Between Pages

- One of the most significant limitations with view state is that it's tightly bound to a specific page.
- If the user navigates to another page, this information is lost.
- This problem has several solutions, and the best approach depends on your requirements.
  - Query String
  - Cross page posting

The Query String

- Pass information using a query string in the URL.
- The query string is the portion of the URL after the question mark.
  - This approach is commonly used in search engines.
  - For example,
    http://www.google.ca/search?q=organic+gardening
  - In this case, it defines a single variable named q, which contains the "organic+gardening" string.

Con..

- The advantage of the query string are ,
  - it's lightweight.
  - doesn't exert any kind of burden on the server.
  - the query string can easily transport the same information from page to page.
- Query String has some limitations, however :
  - Information is limited to simple strings, which must contain URL-legal characters.
  - Information is clearly visible to the user and to anyone else who cares to eavesdrop on the Internet.
  - The enterprising user might decide to modify the query string and supply new values, which your program won't expect and can't protect against.
  - Many browsers impose a limit on the length of a URL (usually from 1 to 2 KB). For that reason, you can't place a large amount of information in the query string and still be assured of compatibility with most browsers.

Con..

- Syntax :
  - URL ? VariableName = Value
- Example :
  - newpage.aspx?recordID=10
- Use :
  - string VariableName = Request.QueryString[" VariableName "];
- Example :
  - string ID = Request.QueryString["recordID"];

Cookies

- Custom cookies provide another way you can store information for later use.
- Cookies are small files that are created on the client's hard drive or, if they're temporary, in the web browser's memory.
- Advantages of cookies are
  - they work transparently without the user being aware that information needs to be stored.
  - They can be easily used by any page in your application and even retained between visits, which allows for truly long-term storage.

Con..

- Dis-advantages of cookies are
  - they're limited to simple string information, and they're easily accessible and readable if the user finds and opens the corresponding file.
  - This is poor choice for complex or private information or large amounts of data.
- Cookies are fairly easy to use. Both the Request and Response objects provide a Cookies collection.
- You can retrieve cookies from the Request object, and you set cookies using the Response object.

Con..

- Create the cookie object.
  - HttpCookie cookie = new HttpCookie("Preferences");
- Set a value in it.
  - cookie["LanguagePref"] = "English";
- Add another value.
  - cookie["Country"] = "US";
- Add it to the current web response.
  - Response.Cookies.Add(cookie);

Con..

- A cookie persist until the user closes the browser and will be sent with every request.
- To create a longer-lived cookie, you can set an expiration date for that cookie
  - cookie.Expires = DateTime.Now.AddYears(1);
- Cookies are retrieved by cookie name using the Request.Cookies collection:
  - HttpCookie cookie = Request.Cookies["Preferences"];

Session State

- It allows information to be stored in one page and accessed in another, and it supports any type of object, including your own custom data types.
- session state uses the same collection syntax as view state. The only difference is the name of the built-in page property, which is Session.
- Session state is ideal for storing information such as the items in the current user's shopping basket when the user browses from one page to another. I
- It force the web server to store additional information in memory. This extra memory requirement, even if it is small, can quickly grow to performance-destroying levels as thousands of clients access the site.

Con..

- Syntax :
  - Session[“variable Name"] = Object Name;
- Example :
  - Session["ProductsDataSet"] = dsProducts;
- Use:
  - Varabile Name = (data type)Session[“variable name"];
- Example:
  - dsProducts = (DataSet)Session["ProductsDataSet"];

Con..

- Session state is global to your entire application for the current user. Session state can be lost in several ways:
  - If the user closes and restarts the browser.
  - If the user accesses the same page through a different browser window, although the session will still exist if a web page is accessed through the original browser window. Browsers differ on how they handle this situation.
  - If the session times out because of inactivity. By default, a session times out after 20 idle minutes.
  - If the programmer ends the session by calling Session.Abandon().