Introduction to Microsoft .NET

● Understanding Microsoft .NET 2015

● .NET 2015 Cross platform support

● Advantages of Microsoft .NET 2015

Introduction to .NET Framework

● Understanding .NET Framework

● .NET Framework Version History

● .NET Framework 4.5 Architecture

● Common Language Runtime (CLR)

● CLR Components

● Microsoft Intermediate language (MSIL)

● Common Type System (CTS)

● Common Language Specification (CLS)

● Relationship Between CTS and CLS

● Framework Class Library (FCL)

● Just In Time Compilation (JIT)

IDE Environment: Visual Studio 2015

Introduction to Visual Studio 2015

● Understanding Visual Studio

● Visual Studio Version History

● Understanding Visual Studio Editions

● Advantage of Visual Studio

● Understanding Visual Studio Features

● Visual Studio Explorer – Solution, Team, Server

● Coding and Debugging

● Types of Projects in Visual Studio

● Creating a Project and Solution

● Toolbox

● Property Window

C# Programming Language

Introduction to C#

● History of C# Version

● C# What,

Why & When

● Structure of a C# Program

● Basic Input / Output Operations

● Compiling, Running, and Debugging

Data Type, Variable & Typecasting

● Understanding Data Type

● Types of Data Type – Value Type & Reference Type

● Naming a variable

● Boxing and Unboxing

● Data Conversions – Implicit & Explicit

● Safe Type Casting with IS and AS Operator

Assemblies

● Types of Assemblies

● Shared Assembly, Private Assembly & Satellite Assembly

● Assembly Manifest, Metadata

● Assembly Cache

Namespaces

● Creating Namespace

● Advantages of Namespace

Garbage Collection

● Understanding Garbage Collection (GC)

● Advantage of Garbage Collector

● Generations in Garbage Collector

● Garbage Collector Working Phase

● Garbage Collection Algorithm

● Finalize and Dispose Method

Operators

● Different Types of Operators

● Operators precedence

Conditional Statements

● Introduction to conditional statement

● If statements

● If..else statement

● If..else ladder

● Switch statement

Loops

● Introduction to loop

● do...while loop

● while loop

● Making star patterns

Jump Statements

● break statements

● continue statement

Loops Contd.

● for loop

● foreach loop

● Making number patterns

Arrays

● Introduction to Array

● One Dimensional Array

● MultiDimensional

Array

● Jagged Array

Strings

● Introduction to strings

● Mutable strings

● Immutable strings

● Strings methods

ObjectOriented

Programming Concepts

● Introduction to object oriented programming

● Classes and Objects

● Inheritance

● Polymorphism

● Abstraction

● Encapsulation

Class and Objects

● Creating a class

● Access Modifiers

● Instance members

● Creating an object

Constructor and Destructor

● Introduction to Constructor

● Types of Constructor

● Default Constructor

● Parameterised Constructor

● Introduction to Destructor

● Creating Destructor

Methods

● Introductiofn to Methods

● Methods Categories

● Methods Parameters

● Optional Parameters

● Call By Value

● Call By Reference

Polymorphism

● Introduction to Polymorphism

● Types of Polymorphism

● Methods Overloading

● Methods Overriding

● Abstract, override and new keywords

Inheritance

● Introduction to Inheritance

● Types of inheritance

● Single Level Inheritance

● MultiLevel

Inheritance

● Multiple Inheritance

● Hybrid Inheritance

● Advantage of Inheritance

● Sealed class and private constructor

Property

● Introduction to Property

● Types of Property

● ReadWrite

Property

● Read Only Property

● Write Only Property

● Advantages of Property

Indexer

● Introduction to Indexer

● Types of Indexer

● Advantages of Indexer

Abstract Class

● Introduction to Abstract Class

● Creating Abstract class

● Need of Abstract class

● Advantages of abstract class

Interface

● Introduction to Interface

● Creating Interface

● Need of Interface

● Abstract class vs. Interface

Static Class

● Introduction to static class

● Creating static class and static methods

● Need of static class

● Normal class vs. Static class

Partial Class

● Introduction to partial class

● Creating partial class

● Need of partial class

Structure

● Introduction to structure

● Creating structure

● Structure vs. Class

Enum

● Introduction to enum

● Creating enum

● Need of enum

Exception Handling

● Understanding Exceptions

● Try, Catch and Finally block

● Throw exception

● Handling Exception

● Custom Exception

Delegates

● Introduction to delegates

● Types of delegates

● Single Delegate

● Multicast Delegate

Generic Delegates

● Introduction to generic delegates

● Func

● Action

● Predicate

Events

● Introduction to events

● Creating Events

● Event Handler

Anonymous Method and Lambda Expression

● Anonymous Method

● Lambda Expression

Attributes

● Introduction to attributes

● Creating attribute class

● Need of attributes

Reflection

● Introduction to reflection

● Viewing Metadata

Collections

● Introduction to .NET Collections

● Introduction to Collection Classes – Array List, Hash Table, Dictionary, Stack, Queue

● Introduction to Collection Interfaces – IEnumerable, IQuerable, IList, ICollection

● Relations between Collection Interfaces and Classes

● Using Collection classes – Array List, Hash Table & Dictionary

Collections Contd.

● Using Collection classes – Stack, Queue

● Using Collection Interfaces – IEnumerable, IQuerable, IList, ICollection

Generics

● Understanding .NET Generics

● Using Generics classes

● Using Generics Interfaces

● Generics Advantages

● Collections vs. Generics

File Handling

● File I/O and Streams

● Stream readers and writers

● Creating file

● Writing data to file

● File opening mode

File Class and Operations

● File class

● Directory class

● Path class

● Performing operations on a file

ASP.NET

Introduction to HTML

● Introduction to ASP.NET

● ASP.NET Version History

● Difference Between ASP and ASP.NET

● ASP.NET Architecture

● Inline Technique & CodeBehind

Technique

● Server Controls

● Page Lifecycle

● Postback

● Page Directives

Web Form Control

● Text Box

● Check Box

● List Box

● Drop DownList

● Image Control

● Radio Button

**HTML**

**Introduction to HTML**

HTMl is the standard markup language for creating web page.

**● History of Html**

|  |  |
| --- | --- |
| **Version** | **Year** |
| HTML | 1991 |
| HTML 2.0 | 1995 |
| HTML 3.2 | 1997 |
| HTML 4.01 | 1999 |
| XHTML | 2000 |
| HTML5 | 2014 |

**● Getting Started With HTML**

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| alt | Specifies an alternative text for an image, when the image cannot be displayed |
| disabled | Specifies that an input element should be disabled |
| href | Specifies the URL (web address) for a link |
| id | Specifies a unique id for an element |
| src | Specifies the URL (web address) for an image |
| style | Specifies an inline CSS style for an element |
| title | Specifies extra information about an element (displayed as a tool tip) |

**● HTML Page Structure**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<html>](http://www.w3schools.com/tags/tag_html.asp) | Defines the root of an HTML document |
| [<body>](http://www.w3schools.com/tags/tag_body.asp) | Defines the document's body |
| [<head>](http://www.w3schools.com/tags/tag_head.asp) | A container for all the head elements (title, scripts, styles, meta information, and more) |
| [<h1> to <h6>](http://www.w3schools.com/tags/tag_hn.asp) | Defines HTML headings |
| [<hr>](http://www.w3schools.com/tags/tag_hr.asp) | Defines a thematic change in the content |

**● Html Element**

An HTML element usually consists of a **start** tag and **end** tag, with the content inserted in between:

<tagname>Content goes here...</tagname>

The HTML **element** is everything from the start tag to the end tag:

<p>My first paragraph.</p>

|  |  |  |
| --- | --- | --- |
| **Start tag** | **Element content** | **End tag** |
| <h1> | My First Heading | </h1> |
| <p> | My first paragraph. | </p> |
| <br> |  |  |

HTML elements with no content are called empty elements. Empty elements do not have an end tag, such as the <br> element (which indicates a line break).

HTML elements can be nested (elements can contain elements).

**● Inline and Block Elements**

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The <div> element is a block-level element.

Examples of block-level elements:

* <div>
* <h1> - <h6>
* <p>
* <form>

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline <span> element inside a paragraph.

Examples of inline elements:

* <span>
* <a>
* <img>

**● Images**

HTML images are defined with the **<img>** tag.

The source file (src), alternative text (alt), width, and height are provided as attributes:

Example

<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">

**● Ordered list and Unordered list**

Ordered list:

An ordered list starts with the **<ol>** tag. Each list item starts with the **<li>** tag.

The **type** attribute of the <ol> tag, defines the type of the list item marker:

|  |  |
| --- | --- |
| **Type** | **Description** |
| type="1" | The list items will be numbered with numbers (default) |
| type="A" | The list items will be numbered with uppercase letters |
| type="a" | The list items will be numbered with lowercase letters |
| type="I" | The list items will be numbered with uppercase roman numbers |
| type="i" | The list items will be numbered with lowercase roman numbers |

Unordered list:

An unordered list starts with the **<ul>** tag. Each list item starts with the **<li>** tag.

The CSS **list-style-type** property is used to define the style of the list item marker:

|  |  |
| --- | --- |
| **Value** | **Description** |
| disc | Sets the list item marker to a bullet (default) |
| circle | Sets the list item marker to a circle |
| square | Sets the list item marker to a square |
| none | The list items will not be marked |

**● Tables and Nesting tables**

An HTML table is defined with the **<table>** tag. Each table row is defined with the **<tr>** tag. A table header is defined with the **<th>** tag. By default, table headings are bold and centered. A table data/cell is defined with the **<td>** tag.

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<table>](http://www.w3schools.com/tags/tag_table.asp) | Defines a table |
| [<th>](http://www.w3schools.com/tags/tag_th.asp) | Defines a header cell in a table |
| [<tr>](http://www.w3schools.com/tags/tag_tr.asp) | Defines a row in a table |
| [<td>](http://www.w3schools.com/tags/tag_td.asp) | Defines a cell in a table |
| [<caption>](http://www.w3schools.com/tags/tag_caption.asp) | Defines a table caption |
| [<colgroup>](http://www.w3schools.com/tags/tag_colgroup.asp) | Specifies a group of one or more columns in a table for formatting |
| [<col>](http://www.w3schools.com/tags/tag_col.asp) | Specifies column properties for each column within a <colgroup> element |
| [<thead>](http://www.w3schools.com/tags/tag_thead.asp) | Groups the header content in a table |
| [<tbody>](http://www.w3schools.com/tags/tag_tbody.asp) | Groups the body content in a table |
| [<tfoot>](http://www.w3schools.com/tags/tag_tfoot.asp) | Groups the footer content in a table |

**CSS**

**Introduction to CSS**

**● Introduction to CSS**

* **CSS** stands for **C**ascading **S**tyle **S**heets
* CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
* CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
* External stylesheets are stored in **CSS files**

**● CSS Id and Class**

The id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element should be unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

The style rule below will be applied to the HTML element with id="para1":

Example

#para1 {  
    text-align: center;  
    color: red;  
}

The class Selector

The class selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

In the example below, all HTML elements with class="center" will be red and center-aligned:

.center {  
    text-align: center;  
    color: red;  
}

**● Colors and Backgrounds**

Colors in CSS are most often specified by:

* a valid color name - like "red"
* an RGB value - like "rgb(255, 0, 0)"
* a HEX value - like "#ff0000"

The CSS background properties are used to define the background effects for elements.

CSS background properties:

* background-color
* background-image
* background-repeat
* background-attachment
* background-position

**● Text and Fonts**

**Text:**

|  |  |  |
| --- | --- | --- |
| **Property** |  | **Description** |
| [color](http://www.w3schools.com/cssref/pr_text_color.asp) |  | Sets the color of text |
| [direction](http://www.w3schools.com/cssref/pr_text_direction.asp) |  | Specifies the text direction/writing direction |
| [letter-spacing](http://www.w3schools.com/cssref/pr_text_letter-spacing.asp) |  | Increases or decreases the space between characters in a text |
| [line-height](http://www.w3schools.com/cssref/pr_dim_line-height.asp) |  | Sets the line height |
| [text-align](http://www.w3schools.com/cssref/pr_text_text-align.asp) |  | Specifies the horizontal alignment of text |
| [text-decoration](http://www.w3schools.com/cssref/pr_text_text-decoration.asp) |  | Specifies the decoration added to text |
| [text-indent](http://www.w3schools.com/cssref/pr_text_text-indent.asp) |  | Specifies the indentation of the first line in a text-block |
| [text-shadow](http://www.w3schools.com/cssref/css3_pr_text-shadow.asp) |  | Specifies the shadow effect added to text |
| [text-transform](http://www.w3schools.com/cssref/pr_text_text-transform.asp) |  | Controls the capitalization of text |
| [unicode-bidi](http://www.w3schools.com/cssref/pr_text_unicode-bidi.asp) |  | Used together with the [direction](http://www.w3schools.com/cssref/pr_text_direction.asp) property to set or return whether the text should be overridden to support multiple languages in the same document |
| [vertical-align](http://www.w3schools.com/cssref/pr_pos_vertical-align.asp) |  | Sets the vertical alignment of an element |
| [white-space](http://www.w3schools.com/cssref/pr_text_white-space.asp) |  | Specifies how white-space inside an element is handled |
| [word-spacing](http://www.w3schools.com/cssref/pr_text_word-spacing.asp) |  | Increases or decreases the space between words in a text |

Font:

In CSS, there are two types of font family names:

* **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
* **font family** - a specific font family (like "Times New Roman" or "Arial")

|  |  |
| --- | --- |
| **Property** | **Description** |
| [font](http://www.w3schools.com/cssref/pr_font_font.asp) | Sets all the font properties in one declaration |
| [font-family](http://www.w3schools.com/cssref/pr_font_font-family.asp) | Specifies the font family for text |
| [font-size](http://www.w3schools.com/cssref/pr_font_font-size.asp) | Specifies the font size of text |
| [font-style](http://www.w3schools.com/cssref/pr_font_font-style.asp) | Specifies the font style for text |
| [font-variant](http://www.w3schools.com/cssref/pr_font_font-variant.asp) | Specifies whether or not a text should be displayed in a small-caps font |
| [font-weight](http://www.w3schools.com/cssref/pr_font_weight.asp) | Specifies the weight of a font |

**● Lists Styles**

In HTML, there are two main types of lists:

* unordered lists (<ul>) - the list items are marked with bullets
* ordered lists (<ol>) - the list items are marked with numbers or letters

The CSS list properties allow you to:

* Set different list item markers for ordered lists
* Set different list item markers for unordered lists
* Set an image as the list item marker
* Add background colors to lists and list items

|  |  |
| --- | --- |
| **Property** | **Description** |
| [list-style](http://www.w3schools.com/cssref/pr_list-style.asp) | Sets all the properties for a list in one declaration |
| [list-style-image](http://www.w3schools.com/cssref/pr_list-style-image.asp) | Specifies an image as the list-item marker |
| [list-style-position](http://www.w3schools.com/cssref/pr_list-style-position.asp) | Specifies if the list-item markers should appear inside or outside the content flow |
| [list-style-type](http://www.w3schools.com/cssref/pr_list-style-type.asp) | Specifies the type of list-item marker |

**● Tables Styles**

|  |  |
| --- | --- |
| **Property** | **Description** |
| [border](http://www.w3schools.com/cssref/pr_border.asp) | Sets all the border properties in one declaration |
| [border-collapse](http://www.w3schools.com/cssref/pr_border-collapse.asp) | Specifies whether or not table borders should be collapsed |
| [border-spacing](http://www.w3schools.com/cssref/pr_border-spacing.asp) | Specifies the distance between the borders of adjacent cells |
| [caption-side](http://www.w3schools.com/cssref/pr_tab_caption-side.asp) | Specifies the placement of a table caption |
| [empty-cells](http://www.w3schools.com/cssref/pr_tab_empty-cells.asp) | Specifies whether or not to display borders and background on empty cells in a table |
| [table-layout](http://www.w3schools.com/cssref/pr_tab_table-layout.asp) | Sets the layout algorithm to be used for a table |

**● Introduction to CSS3**

CSS3 has been split into "modules". It contains the "old CSS specification" (which has been split into smaller pieces). In addition, new modules are added.

Some of the most important CSS3 modules are:

* Selectors
* Box Model
* Backgrounds and Borders
* Image Values and Replaced Content
* Text Effects
* 2D/3D Transformations
* Animations
* Multiple Column Layout
* User Interface

**● CSS Box Model**

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout. The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

Explanation of the different parts:

* **Content** - The content of the box, where text and images appear
* **Padding** - Clears an area around the content. The padding is transparent
* **Border** - A border that goes around the padding and content
* **Margin** - Clears an area outside the border. The margin is transparent

**● CSS Grouping/Nesting, positions and advance**

CSS Grouping

In CSS, most of the time same style will be applied to multiple elements. The code size can be minimized by grouping the selectors and separating them by commas like shown below.

<style>

h1 {

color:blue;

}

h2 {

color: blue;

}

h3 {

color: blue;

}

h1, h2, h3

{

color: blue;

}

</style>

CSS Nesting

CSS Nesting is used to apply CSS to a selector inside a selector.

In the below example, the CSS style is applied to the paragraphs under the class "para1" instead of all paragraphs.

<style>

p {

color:blue;

}

.para1 p {

color:red;

}

</style>

**Master Page**

**● Introduction to Master Page**

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.

**● Creating Master Page**

The master page is the template for how your pages will look. In this section, you will first create a master page. You will then use a table to lay out the master page with a menu, a logo, and a footer that will appear on each page of your site. You will also work with a content placeholder, which is a region in the master page that can be replaced with information in a content page.

To create the master page

1. In **Solution Explorer**, right-click the name of your Web site, and then click **AddNew Item**.
2. Under **Visual Studio installed templates**, click **Master Page**.
3. In the **Name** box, type **Master1**.
4. Select the **Place code in separate file** check box.
5. In the **Language** list, click the programming language you prefer and then click **Add**.

The new master page opens in **Source** view.

**● Advantages of Master Pages**

* Master pages enable consistent and standardized layout of the website.
* You can make layout changes of the site in the master page instead of making changes in the pages.
* It is very easy to implement.
* It provides an object model which allows you to customize the master page from individual content pages.
* It allows you to centralize the common functionality of your pages so that you can make updates in just one place.

**Forms & Validations**

**● Creating Forms**

**● Creating User SignUp Page**

**● Using Data Validations Controls – Required Field, Regular Expression, Compare and**

**Range Validator**

**State Management**

State management means to preserve state of a control, web page, object/data, and user in the application explicitly because all ASP.NET web applications are stateless, i.e., by default, for each page posted to the server, the state of controls is lost.

Levels of state management

1. Control level: In ASP.NET, by default controls provide state management automatically.
2. Variable or object level: In ASP.NET, member variables at page level are stateless and thus we need to maintain state explicitly.
3. Single or multiple page level: State management at single as well as multiple page level i.e., managing state between page requests.
4. User level: State should be preserved as long as a user is running the application.
5. Application level: State available for complete application irrespective of the user, i.e., should be available to all users.
6. Application to application level: State management between or among two or more applications.

**● Client Side State Management Techniques – Cookies, ViewState, Hidden Field, Query**

**String**

Hidden Field:

Hidden field is a control provided by ASP.NET which is used to store small amounts of data on the client. It store one value for the variable and it is a preferable way when a variable's value is changed frequently. Hidden field control is not rendered to the client (browser) and it is invisible on the browser. A hidden field travels with every request like a standard control’s value.

Let us see with a simple example how to use a hidden field. These examples increase a value by 1 on every "No Action Button" click. The source of the hidden field control is

<asp:HiddenField ID="HiddenField1" runat="server" />

View State:

View state is another client side state management mechanism provided by ASP.NET to store user's data, i.e., sometimes the user needs to preserve data temporarily after a post back, then the view state is the preferred way for doing it. It stores data in the generated HTML using hidden field not on the server.

View State provides page level state management i.e., as long as the user is on the current page, state is available and the user redirects to the next page and the current page state is lost. View State can store any type of data because it is object type but it is preferable not to store a complex type of data due to the need for serialization and deserilization on each post back. View state is enabled by default for all server side controls of ASP.NET with a property EnableviewState set to true.

Let us see how ViewState is used with the help of the following example. In the example we try to save the number of postbacks on button click.

protected void Page\_Load(object sender, EventArgs e)

{

if (IsPostBack)

{

if (ViewState["count"] != null)

{

int ViewstateVal = Convert.ToInt32(ViewState["count"]) + 1;

Label1.Text = ViewstateVal.ToString();

ViewState["count"]=ViewstateVal.ToString();

}

else

{

ViewState["count"] = "1";

}

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

Label1.Text=ViewState["count"].ToString();

}

Cookies:

Cookie is a small text file which is created by the client's browser and also stored on the client hard disk by the browser. It does not use server memory. Generally a cookie is used to identify users.

A cookie is a small file that stores user information. Whenever a user makes a request for a page the first time, the server creates a cookie and sends it to the client along with the requested page and the client browser receives that cookie and stores it on the client machine either permanently or temporarily (persistent or non persistence). The next time the user makes a request for the same site, either the same or another page, the browser checks the existence of the cookie for that site in the folder. If the cookie exists it sends a request with the same cookie, else that request is treated as a new request.

**Types of Cookies**

**1. Persistence Cookie**: Cookies which you can set an expiry date time are called persistence cookies. Persistence cookies are permanently stored till the time you set.

Let us see how to create persistence cookies. There are two ways, the first one is:

Response.Cookies["nameWithPCookies"].Value = "This is A Persistance Cookie";

Response.Cookies["nameWithPCookies"].Expires = DateTime.Now.AddSeconds(10);

And the second one is:

HttpCookie aCookieValPer = new HttpCookie("Persistance");

aCookieValPer.Value = "This is A Persistance Cookie";

aCookieValPer.Expires = DateTime.Now.AddSeconds(10);

Response.Cookies.Add(aCookieValPer);

**2. Non-Persistence Cookie**: Non persistence cookies are not permanently stored on the user client hard disk folder. It maintains user information as long as the user accesses the same browser. When user closes the browser the cookie will be discarded. Non Persistence cookies are useful for public computers.

Let us see how to create a non persistence cookies. There are two ways, the first one is:

Response.Cookies["nameWithNPCookies"].Value = "This is A Non Persistance Cookie";

And the second way is:

HttpCookie aCookieValNonPer = new HttpCookie("NonPersistance");

aCookieValNonPer.Value = "This is A Non Persistance Cookie;

Response.Cookies.Add(aCookieValNonPer);

**● Server Side Statement Management – Session, Application State**

**Session:**

Session management is a very strong technique to maintain state. Generally session is used to store user's information and/or uniquely identify a user (or say browser). The server maintains the state of user information by using a session ID. When users makes a request without a session ID, ASP.NET creates a session ID and sends it with every request and response to the same user.

How to get and set value in Session:

Hide   Copy Code

Session["Count"] = Convert.ToInt32(Session["Count"]) + 1;*//Set Value to The Session*

Label2.Text = Session["Count"].ToString(); *//Get Value from the Sesion*

Let us see an example where we save the count of button clicks in a session, and save the “number of redirects to the same page” button click in a query string. Here I have set the expiry to 10 minutes. After starting the application, the application variable exists till the end of the application. A session variable will expire after 10 minutes (if it is idle). A query string contains the value in URL so it won’t depend on the user idle time and could be used by the server anytime it is passed with a request.

**Session Events in ASP.NET**

To manage a session, ASP.NET provides two events: session\_start and session\_end that is written in a special file called *Global.asax* in the root directory of the project.

**Session\_Start**: The Session\_start event is raised every time a new user makes a request without a session ID, i.e., new browser accesses the application, then a session\_start event raised. Let's see the *Global.asax* file.

Hide   Copy Code

void Session\_Start(object sender, EventArgs e)

{

Session["Count"] = 0; *// Code that runs when a new session is started*

}

**Session\_End**: The Session\_End event is raised when session ends either because of a time out expiry or explicitly by using Session.Abandon(). The Session\_End event is raised only in the case of In proc mode not in the state server and SQL Server modes.

There are four session storage mechanisms provided by ASP.NET:

* *In Proc mode*
* *State Server mode*
* *SQL Server mode*
* *Custom mode*

**In Process mode**: In proc mode is the default mode provided by ASP.NET. In this mode, session values are stored in the web server's memory (in IIS). If there are more than one IIS servers then session values are stored in each server separately on which request has been made. Since the session values are stored in server, whenever server is restarted the session values will be lost.

<configuration>

<sessionstate mode="InProc" cookieless="false" timeout="10"

stateConnectionString="tcpip=127.0.0.1:80808"

sqlConnectionString="Data Source=.\SqlDataSource;User ID=userid;Password=password"/>

</configuration>

**In State Server mode**: This mode could store session in the web server but out of the application pool. But usually if this mode is used there will be a separate server for storing sessions, i.e., stateServer. The benefit is that when IIS restarts the session is available. It stores session in a separate Windows service. For State server session mode, we have to configure it explicitly in the web config file and start the aspnet\_state service.

<configuration><sessionstate mode="stateserver" cookieless="false"

timeout="10" stateConnectionString="tcpip=127.0.0.1:42424"

sqlConnectionString="Data Source=.\SqlDataSource;User ID=userid;Password=password"/> </configuration>

**In SQL Server mode**: Session is stored in a SQL Server database. This kind of session mode is also separate from IIS, i.e., session is available even after restarting the IIS server. This mode is highly secure and reliable but also has a disadvantage that there is overhead from serialization and deserialization of session data. This mode should be used when reliability is more important than performance.

<configuration>

<sessionstate mode="sqlserver" cookieless="false" timeout="10"

stateConnectionString="tcpip=127.0.0.1:4 2424"

sqlConnectionString="Data Source=.\SqlDataSource;User ID=userid;Password=password"/>

</configuration>

**Custom Session mode**: Generally we should prefer in proc state server mode or SQL Server mode but if you need to store session data using other than these techniques then ASP.NET provides a custom session mode. This way we have to maintain everything customized even generating session ID, data store, and also security.

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Cookieless true/false | Indicates that the session is used with or without cookie. cookieless set to true indicates sessions without cookies is used and cookieless set to false indicates sessions with cookies is used. cookieless set to false is the default set. |
| timeout | Indicates the session will abound if it is idle before session is abounded explicitly (the default time is 20 min). |
| StateConnectionString | Indicates the session state is stored on the remote computer (server). This attribute is required when session mode is StateServer |
| SqlConnectionString | Indicates the session state is stored in the database. This attribute is required when session mode is SqlServer. |

**Application State:**

Application state is a server side state management technique. The date stored in application state is common for all users of that particular ASP.NET application and can be accessed anywhere in the application. It is also called application level state management. Data stored in the application should be of small size.

How to get and set a value in the **application object**:

Hide   Copy Code

Application["Count"] = Convert.ToInt32(Application["Count"]) + 1; *//Set Value to The Application Object*

Label1.Text = Application["Count"].ToString(); *//Get Value from the Application Object*

**Application events in ASP.NET**

There are three types of events in ASP.NET. Application event is written in a special file called *Global.asax*. This file is not created by default, it is created explicitly by the developer in the root directory. An application can create more than one *Global.asax* file but only the root one is read by ASP.NET.

Application\_start: The Application\_Start event is raised when an app domain starts. When the first request is raised to an application then the Application\_Start event is raised. Let's see the *Global.asax* file.

Hide   Copy Code

void Application\_Start(object sender, EventArgs e)

{

Application["Count"] = 0;

}

Application\_Error: It is raised when an unhandled exception occurs, and we can manage the exception in this event.

Application\_End: The Application\_End event is raised just before an application domain ends because of any reason, may IIS server restarting or making some changes in an application cycle.

**Bootstrap**

**Introduction to Bootstrap**

* Bootstrap is a free front-end framework for faster and easier web development
* Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
* Bootstrap also gives you the ability to easily create responsive designs

**● Introduction to Responsive Design**

Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

**● Need of Bootstrap**

Advantages of Bootstrap:

* **Easy to use:** Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
* **Responsive features:** Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
* **Mobile-first approach:** In Bootstrap 3, mobile-first styles are part of the core framework
* **Browser compatibility:** Bootstrap is compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Safari, and Opera)

**● Bootstrap Fundamentals**

* **Scaffolding** − Bootstrap provides a basic structure with Grid System, link styles, and background. This is is covered in detail in the section**Bootstrap Basic Structure**
* **CSS** − Bootstrap comes with the feature of global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system. This is covered in detail in the section **Bootstrap with CSS**.
* **Components** − Bootstrap contains over a dozen reusable components built to provide iconography, dropdowns, navigation, alerts, pop-overs, and much more. This is covered in detail in the section **Layout Components**.
* **JavaScript Plugins** − Bootstrap contains over a dozen custom jQuery plugins. You can easily include them all, or one by one. This is covered in details in the section **Bootstrap Plugins**.
* **Customize** − You can customize Bootstrap's components, LESS variables, and jQuery plugins to get your very own version.

**● Bootstrap Grid System**

Bootstrap's grid system allows up to 12 columns across the page.

If you do not want to use all 12 columns individually, you can group the columns together to create wider columns:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 |
| span 4 | | | | span 4 | | | | span 4 | | | |
| span 4 | | | | span 8 | | | | | | | |
| span 6 | | | | | | span 6 | | | | | |
| span 12 | | | | | | | | | | | |

Bootstrap's grid system is responsive, and the columns will re-arrange automatically depending on the screen size

The Bootstrap grid system has four classes:

* xs (for phones)
* sm (for tablets)
* md (for desktops)
* lg (for larger desktops)

The classes above can be combined to create more dynamic and flexible layouts.

**● Bootstrap Components**

**● Header, Footer, Navigation Bar**

Header:

The page header is a nice little feature to add appropriate spacing around the headings on a page. This is particularly helpful on a web page where you may have several post titles and need a way to add distinction to each of them. To use a page header, wrap your heading in a <div> with a class of **.page-header** −

<div class = "page-header">

<h1>

Example page header

<small>Subtext for header</small>

</h1>

</div>

<p>This is a sample text.This is a sample text.This is a sample text. This is a sample text.</p>

Navigation Bar:

The navbar is one of the prominent features of Bootstrap sites. Navbars are responsive 'meta' components that serve as navigation headers for your application or site. Navbars collapse in mobile views and become horizontal as the available viewport width increases. At its core, the navbar includes styling for site names and basic navigation.

To create a default navbar −

* Add the classes **.navbar, .navbar-default** to the <nav> tag.
* Add **role = "navigation"** to the above element, to help with accessibility.
* Add a header class **.navbar-header** to the <div> element. Include an <a> element with class **navbar-brand**. This will give the text a slightly larger size.
* To add links to the navbar, simply add an unordered list with the classes of **.nav, .navbar-nav**.

**● Bootstrap Modals**

A modal is a child window that is layered over its parent window. Typically, the purpose is to display content from a separate source that can have some interaction without leaving the parent window. Child windows can provide information, interaction, or more.

If you want to include this plugin functionality individually, then you will need**modal.js**.

Here are some useful methods that can be used with modal().

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| **Options** − .modal(options) | Activates your content as a modal. Accepts an optional options object. | $('#identifier').modal({  keyboard: false  }) |
| **Toggle** − .modal('toggle') | Manually toggles a modal. | $('#identifier').modal('toggle') |
| **Show** − .modal('show') | Manually opens a modal. | $('#identifier').modal('show') |
| **Hide** − .modal('hide') | Manually hides a modal. | $('#identifier').modal('hide') |

**● Bootstrap Form Elements**

Bootstrap provides you with following types of form layouts −

* Vertical (default) form
* In-line form
* Horizontal form

Vertical or Basic Form

The basic form structure comes with Bootstrap; individual form controls automatically receive some global styling. To create a basic form do the following −

* Add a role *form* to the parent <form> element.
* Wrap labels and controls in a <div> with class *.form-group*. This is needed for optimum spacing.
* Add a class of *.form-control* to all textual <input>, <textarea>, and <select> elements.

Inline Form

To create a form where all of the elements are inline, left aligned and labels are alongside, add the class *.form-inline*to the <form> tag.

Horizontal Form

Horizontal forms stands apart from the others not only in the amount of markup, but also in the presentation of the form. To create a form that uses the horizontal layout, do the following −

* Add a class of *.form-horizontal* to the parent <form> element.
* Wrap labels and controls in a <div> with class *.form-group*.
* Add a class of *.control-label* to the labels.

**● Bootstrap Icons**

Bootstrap provides a few different options for icons as fonts. sample examples of icons are as shown below −

|  |  |  |
| --- | --- | --- |
| **Example** | **Description** | **Download Link** |
| Round Icon Badges | This example indicates about Round Icon Badges in Bootstrap | [**Download**](https://www.tutorialspoint.com/bootstrap/src/zip/icons.zip) |
| Rotation and Flip | This example indicates about icons with animation in Bootstrap | [**Download**](https://www.tutorialspoint.com/bootstrap/src/zip/icons2.zip) |
| Ui-Kit-Metro Like Nav | This example indicates about icons with UI animation in Bootstrap | [**Download**](https://www.tutorialspoint.com/bootstrap/src/zip/icons3.zip) |
| Fat and Flat blue button | This example indicates about Fat and Flat blue button in Bootstrap | [**Download**](https://www.tutorialspoint.com/bootstrap/src/zip/icons4.zip) |

**● Typography**

Bootstrap uses Helvetica Neue, Helvetica, Arial, and sans-serif in its default font stack. Using typography feature of Bootstrap you can create headings, paragraphs, lists and other inline elements.

<h1> - <h6>, <small>, <mark>, <abbr>, <blackquote>, <dl>, <code>, <kbd>, <pre>,

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | **Example** |
| .lead | Makes a paragraph stand out | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_txt_lead&stacked=h) |
| .small | Indicates smaller text (set to 85% of the size of the parent) | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_txt_small&stacked=h) |
| .text-left | Indicates left-aligned text | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_text-left&stacked=h) |
| .text-center | Indicates center-aligned text | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_text-left&stacked=h) |
| .text-right | Indicates right-aligned text | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_text-left&stacked=h) |
| .text-justify | Indicates justified text | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_text-left&stacked=h) |
| .text-nowrap | Indicates no wrap text | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_text-left&stacked=h) |
| .text-lowercase | Indicates lowercased text | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_text-lowercase&stacked=h) |
| .text-uppercase | Indicates uppercased text | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_text-lowercase&stacked=h) |
| .text-capitalize | Indicates capitalized text | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_text-lowercase&stacked=h) |
| .initialism | Displays the text inside an <abbr> element in a slightly smaller font size | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_txt_abbr2&stacked=h) |
| .list-unstyled | Removes the default list-style and left margin on list items (works on both <ul>  And <ol>). This class only applies to immediate children list items (to remove the default list-style from any nested l lists, apply this class to any nested lists as well) |  |
| .list-inline | Places all list items on a single line | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_txt_list-inline&stacked=h) |
| .dl-horizontal | Lines up the terms (<dt>) and descriptions (<dd>) in <dl> elements side-by-side. Starts off like default <dl>s, but when the browser window expands, it will line up side-by-side | [Try it](http://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_ref_txt_dl-horizontal&stacked=h) |
| .pre-scrollable | Makes a <pre> element scrollable |  |

**● Customizing Bootstrap**

If you want to customize its components under a new class name thats perhaps more contextual to your project, you can try utilizing SASS/LESS' **mixins** or using **extend** to inherit Bootstrap's class styles into your new class component. Otherwise, you can just override the component you need using the same selector.

When adding additional classes to certain assets, try to keep a similar naming convention. eg: .btn-rounded .btn-newcolor

**● Panels, Sliders, Tooltips**

Panels:

Panel components are used when you want to put your DOM component in a box. To get a basic panel, just add class **.panel** to the <div> element. Also add class **.panel-default** to this element as shown in the following example −

<div class = "panel panel-default">

<div class = "panel-body">

This is a Basic panel

</div>

</div>

Sliders:

Bootstrap provides a few different options for styling Navigation slider. Some of examples are as shown below

|  |  |  |
| --- | --- | --- |
| **Example** | **Description** | **Download Link** |
| Slide menu on over | This example indicates about Slider Menu structure in Bootstrap | [**Download**](https://www.tutorialspoint.com/bootstrap/src/zip/slider.zip) |
| Sub Menu | This example indicates about Slider Sub Menu structure in Bootstrap | [**Download**](https://www.tutorialspoint.com/bootstrap/src/zip/slider3.zip) |
| Sidebar With Tabs | This example indicates about Slider With Tabs structure in Bootstrap | [**Download**](https://www.tutorialspoint.com/bootstrap/src/zip/slider2.zip) |
| Tabs | This example indicates about Tabs structure in Bootstrap | [**Download**](https://www.tutorialspoint.com/bootstrap/src/zip/slider4.zip) |

Tooltips:

If you want to include this plugin functionality individually, then you will need **tooltip.js**.

Usage

The tooltip plugin generates content and markup on demand, and by default places tooltips after their trigger element. You can add tooltips in the following two ways −

* **Via data attributes** − To add a tooltip, add **data-toggle = "tooltip"** to an anchor tag. The title of the anchor will be the text of a tooltip. By default, tooltip is set to top by the plugin.

<a href = "#" data-toggle = "tooltip" title = "Example tooltip">Hover over me</a>

* **Via JavaScript** − Trigger the tooltip via JavaScript −

$('#identifier').tooltip(options)

**The following are some useful methods for tooltips −**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| **Options** − .tooltip(options) | Attaches a tooltip handler to an element collection. | $().tooltip(options) |
| **Toggle** − .tooltip('toggle') | Toggles an element's tooltip. | $('#element').tooltip('toggle') |
| **Show** − .tooltip('show') | Reveals an element's tooltip. | $('#element').tooltip('show') |
| **Hide** − .tooltip('hide') | Hides an element's tooltip. | $('#element').tooltip('hide') |
| **Destroy** − .tooltip('destroy') | Hides and destroys an element's tooltip. | $('#element').tooltip('destroy') |

SQL Server

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database.

Why SQL?

 Allows users to access data in relational database management systems.

 Allows users to describe the data.

 Allows users to define the data in database and manipulate that data.

 Allows to embed within other languages using SQL modules, libraries & pre-compilers.

 Allows users to create and drop databases and tables.

**● Understanding SQL Server**

**SQL A**rchitecture:



● SQL Server Version History

|  |  |  |
| --- | --- | --- |
| **Version** | **Year** | **Code Name** |
| 6.0 | 1995 | SQL95 |
| 6.5 | 1996 | Hydra |
| 7.0 | 1998 | Sphinx |
| 8.0 (2000) | 2000 | Shiloh |
| 9.0 (2005) | 2005 | Yukon |
| 10.0 (2008) | 2008 | Katmai |
| 10.5 (2008 R2) | 2010 | Kilimanjaro |
| 11.0 (2012) | 2012 | Denali |
| 12 (2014) | 2014 | Hekaton (initially), SQL 14 (current) |

● RDBMS VS DBMS VS NoSQL

DBMS provides a systematic and organized way of storing, managing and retrieving from collection of logically related information. RDBMS also provides what DBMS provides but above that it provides relationship integrity. So in short we can say ***RDBMS = DBMS + REFERENTIAL INTEGRITY***

**● Advantage of SQL Server**

SQL Server offers many advantages over other types of databases. Some of those advantages include the following:

* **Security**: SQL Server provides two security models including integration with Windows Authentication to provide robust security platform for managing your organizations information. Security can be managed across all database objects including tables, views, stored procedures and at a very granular level, to provide complete information security. Database encryption is now available as well to secure SQL Server database at rest.
* **Scalability**: SQL Server is a highly scalable enterprise database platform capabable of managing petabytes of data while delivery high performance. This allows organizations to deploy SQL Server databases with the confidence that SQL Server can continue to deliver superior performance even as their needs and businesses grow.
* **High Availability/Failover**: SQL Server provides a variety of high availablity features including fail over clustering availale on the Standard and Enterprise editions, database mirroring, log shipping, and database snapshots to meet your organizations' needs.
* **Business Intelligence**: SQL Server Standard and Enterprise Editions come with extensive business intelligence capabilities through SQL Server Analysis Services. Analytical models can be developed to breakdown business information to provide business insights for managing and forecasting business needs. SQL Server also comes with integrated reporting capabilities from SQL Server Reporting Services which allows organizations to access business critical information across the organization. In more recent versions of SQL Server, PowerPivot for Microsoft Excel has been added to allow users another familiar tool for analyzing business data.
* **Integration**: SQL Server provides a robust set of integration tools call SQL Server Integration Services (SSIS) which can be used considate data from multiple heterogenous data sources for holistic data view and comprehensive data analysis. SSIS can also be used to migrate data from other database formats (Oracle, DBase, Paradox, legacy formats, Microsoft Access) to SQL Server.

**T SQL**

**● Types of SQL Commands**

**● Data Definition Language (DDL)**

Data definition statement are used to define the database structure or table.

|  |  |
| --- | --- |
| Statement | Description |
| CREATE | Create new database/table. |
| ALTER | Modifies the structure of database/table. |
| DROP | Deletes a database/table. |
| TRUNCATE | Remove all table records including allocated table spaces. |
| RENAME | Rename the database/table. |

● Data Manipulation Language (DML)

Data manipulation statement are used for managing data within table object.

|  |  |
| --- | --- |
| Statement | Description |
| SELECT | Retrieve data from the table. |
| INSERT | Insert data into a table. |
| UPDATE | Updates existing data with new data within a table. |
| DELETE | Deletes the records rows from the table. |
| MERGE | MERGE (also called UPSERT) statements to INSERT new records or UPDATE existing records depending on condition matches or not. |
| LOCK TABLE | LOCK TABLE statement to lock one or more tables in a specified mode. Table access denied to a other users for the duration of your table operation. |
| CALL EXPLAIN PLAN | Statements are supported in PL/SQL only for executed dynamically. CALL a PL/SQL program or EXPLAIN PATH access the data path. |

**● Data Query Language (DQL)**

|  |  |
| --- | --- |
| **Command** | **Description** |
| SELECT | Retrieves certain records from one or more tables |

**● Data Control Language (DCL)**

Data control statement are use to give privileges to access limited data.

|  |  |
| --- | --- |
| Statement | Description |
| GRANT | Gives privileges to user for accessing database data. |
| REVOKE | Take back for given privileges. |
| ANALYZE | ANALYZE statement to collect statistics information about index, cluster, table. |
| AUDIT | To track the occurrence of a specific SQL statement or all SQL statements during the user sessions. |
| COMMENT | Write comment to the data table. |

**● Transaction Control Language (TCL)**

Transaction control statement are used to apply the changes permanently save into database.

|  |  |
| --- | --- |
| Statement | Description |
| COMMIT | Permanent work save into database. |
| ROLLBACK | Restore database to original form since the last COMMIT. |
| SAVEPOINT | Create SAVEPOINT for later use ROLLBACK the new changes. |
| SET TRANSACTION | SET TRANSACTION command set the transaction properties such as read-write/read only access. |

**Session Control Statements (SCS)**

Session control statement are managed properties dynamically of a user session.

|  |  |
| --- | --- |
| Statement | Description |
| ALTER SESSION | ALTER SESSION statement to modify conditions or parameters that are affect to your database connection. |
| SET ROLE | SET ROLE statement to enable or disable the roles that are currently enabled for the session. |

● **Creating Database**

Database is a collection of objects such as table, view, stored procedure, function, trigger, etc.

In MS SQL Server, two types of databases are available.

* System databases
* User Databases

**System Databases**

System databases are created automatically when we install MS SQL Server. Following is a list of system databases −

* Master
* Model
* MSDB
* Tempdb
* Resource (Introduced in 2005 version)
* Distribution (It’s for Replication feature only)

**User Databases**

User databases are created by users (Administrators, developers, and testers who have access to create databases).

Create database dbname;

**● Altering Database**

**ALTER DATABASE { database\_name | CURRENT }**

**{**

**MODIFY NAME = new\_database\_name**

**| COLLATE collation\_name**

**| <file\_and\_filegroup\_options>**

**| <set\_database\_options>**

**}**

**● Deleting Database**

DROP DATABASE DatabaseName;

**● Creating Table**

The CREATE TABLE statement is used to create a table in a database. Tables are organized into rows and columns; and each table must have a name.

CREATE TABLE *table\_name*  
(  
*column\_name1 data\_type*(*size*),  
*column\_name2 data\_type*(*size*),  
*column\_name3 data\_type*(*size*),  
....  
);

**● Altering Table**

**ALTER TABLE table\_name RENAME TO new\_table\_name;**

The basic syntax of **ALTER TABLE** to add a new column in an existing table is as follows:

ALTER TABLE table\_name ADD column\_name datatype;

The basic syntax of ALTER TABLE to **DROP COLUMN** in an existing table is as follows:

ALTER TABLE table\_name DROP COLUMN column\_name;

The basic syntax of ALTER TABLE to change the **DATA TYPE** of a column in a table is as follows:

ALTER TABLE table\_name MODIFY COLUMN column\_name datatype;

The basic syntax of ALTER TABLE to add a **NOT NULL** constraint to a column in a table is as follows:

ALTER TABLE table\_name MODIFY column\_name datatype NOT NULL;

The basic syntax of ALTER TABLE to **ADD UNIQUE CONSTRAINT** to a table is as follows:

ALTER TABLE table\_name

ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);

The basic syntax of ALTER TABLE to **ADD CHECK CONSTRAINT** to a table is as follows:

ALTER TABLE table\_name

ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);

The basic syntax of ALTER TABLE to **ADD PRIMARY KEY** constraint to a table is as follows:

ALTER TABLE table\_name

ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);

The basic syntax of ALTER TABLE to **DROP CONSTRAINT** from a table is as follows:

ALTER TABLE table\_name

DROP CONSTRAINT MyUniqueConstraint;

If you're using MySQL, the code is as follows:

ALTER TABLE table\_name

DROP INDEX MyUniqueConstraint;

The basic syntax of ALTER TABLE to **DROP PRIMARY KEY** constraint from a table is as follows:

ALTER TABLE table\_name

DROP CONSTRAINT MyPrimaryKey;

**● Deleting Table**

**DROP TABLE table\_name;**

**SQL Keys**

A key is a single or combination of multiple fields in a table. It is used to fetch or retrieve records/data-rows from data table according to the condition/requirement. Keys are also used to create relationship among different database tables or views.

**● Types of SQL Keys**

**● Super Key**

Super key is a set of one or more than one keys that can be used to identify a record uniquely in a table. **Example :** Primary key, Unique key, Alternate key are subset of Super Keys.

**● Primary Key**

The PRIMARY KEY constraint uniquely identifies each record in a database table.

Primary keys must contain UNIQUE values. A primary key column cannot contain NULL values.

Most tables should have a primary key, and each table can have only ONE primary key.

The following SQL creates a PRIMARY KEY on the "P\_Id" column when the "Persons" table is created:

SQL Server / Oracle / MS Access:

CREATE TABLE Persons  
(  
P\_Id int NOT NULL PRIMARY KEY,  
LastName varchar(255) NOT NULL,  
FirstName varchar(255),  
Address varchar(255),  
City varchar(255)  
)

**DROP a PRIMARY KEY**

ALTER TABLE Persons

DROP PRIMARY KEY

**● Unique Key**

Unique key is a set of one or more fields/columns of a table that uniquely identify a record in database table. It is like Primary key but it can accept only one null value and it cannot have duplicate values.

**● Foreign Key**

A foreign key is a key used to link two tables together. This is sometimes called a referencing key. Foreign Key is a column or a combination of columns whose values match a Primary Key in a different table. The relationship between 2 tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

**● Candidate Key**

A Candidate Key is a set of one or more fields/columns that can identify a record uniquely in a table. There can be multiple Candidate Keys in one table. Each Candidate Key can work as Primary Key.

**Example:** In below diagram ID, RollNo and EnrollNo are Candidate Keys since all these three fields can be work as Primary Key.

**● Alternate Key**

Alternate key is a key that can be work as a primary key. Basically it is a candidate key that currently is not primary key.

**Example:** In below diagram RollNo and EnrollNo becomes Alternate Keys when we define ID as Primary Key.

**● Composite Key**

Composite Key is a combination of more than one fields/columns of a table. It can be a Candidate key, Primary key.

**SQL Commands & Predicates**

**● Insert**

The INSERT INTO statements is used to insert new records in a table.

***INSERT INTO ColorTable (code, colorvalue) VALUES ('b1', 'Brown’)***

● Update

The update statement is used to update records in a table .

***UPDATE ColorTable  SET colorvalue =’Black’ where code=’bl’***

● Delete

The DELETE statement is used to delete rows in a table.

***DELETE FROM ColorTable  WHERE code = ‘b1'***

● Truncate

**TRUNCATE TABLE** command is used to delete complete data from an existing table.

The basic syntax of **TRUNCATE TABLE** is as follows:

TRUNCATE TABLE table\_name;

● Delete VS Truncate

Following are difference between them:

* DELETE TABLE syntax logs the deletes thus making the delete operation slow. TRUNCATE table does not log any information but it logs information about deallocation of data page of the table. So TRUNCATE table is faster as compared to delete table.
* DELETE table can be rolled back while TRUNCATE cannot be.
* DELETE table can have criteria while TRUNCATE cannot.
* TRUNCATE table cannot have trigger

● Distinct

In a table, a column may contain many duplicate values; and sometimes you only want to list the different (distinct) values. The DISTINCT keyword can be used to return only distinct (different) values.

SELECT DISTINCT *column\_name*,*column\_name* FROM *table\_name*;

● Between…And

The BETWEEN operator selects values within a range. The values can be numbers, text, or dates.

SELECT *column\_name(s)*  
FROM *table\_name*  
WHERE *column\_name*BETWEEN *value1* AND *value2;*

**Between …In**

SELECT \* FROM Products  
WHERE (Price BETWEEN 10 AND 20)  
AND NOT CategoryID IN (1,2,3);

● **In**

The IN operator allows you to specify multiple values in a WHERE clause.

SELECT *column\_name(s)*  
FROM *table\_name*  
WHERE *column\_name* IN (*value1*,*value2*,...);

● **Like**

The LIKE operator is used to search for a specified pattern in a column.

SELECT *column\_name(s)*  
FROM *table\_name*  
WHERE *column\_name* LIKE *pattern*;

● Is Null

|  |  |
| --- | --- |
| IS NULL | The NULL operator is used to compare a value with a NULL value. |

SQL Clauses

● Select

The SELECT statement is used to select data from a database. The result is stored in a result table, called the result-set.

SELECT *column\_name*,*column\_name* FROM *table\_name*;

● Where

The WHERE clause is used to extract only those records that fulfill a specified criterion.

SELECT *column\_name*,*column\_name* FROM *table\_name*  
WHERE *column\_name operator value*;

Operators in where: =, < >, >, <, < =, > =, between, like, in

● Order By

Using the “ORDER BY” clause, you either sort the data in ascending manner or descending manner.

***select \* from sales.salesperson order by salespersonid asc***

***select \* from sales.salesperson order by salespersonid desc***

● Group By

SELECT SUM(column\_name)

FROM table\_name

WHERE CONDITION

GROUP BY column\_name;

● Having

SELECT SUM(column\_name)

FROM table\_name

WHERE CONDITION

GROUP BY column\_name

HAVING (arithematic function condition);

Query & SubQuery

● Understanding Query and SubQuery

● Writing and Executing Query

Here is our SQL query:

SELECT  
     first\_name,  
     last\_name  
FROM  
     people\_massachusetts  
WHERE  
     hair\_color = "red"  
AND  
     birth\_date BETWEEN '2003-01-01' AND '2003-12-31'  
ORDER BY  
     last\_name  
;

**SQL Joins**

An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

● Types of Joins

* **INNER JOIN**: Returns all rows when there is at least one match in BOTH tables
* **LEFT JOIN**: Return all rows from the left table, and the matched rows from the right table
* **RIGHT JOIN**: Return all rows from the right table, and the matched rows from the left table
* **FULL JOIN**: Return all rows when there is a match in ONE of the tables

● Inner Join

The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.

SELECT *column\_name(s)*  
FROM *table1*  
INNER JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

● Outer Join

**LEFT OUTER JOIN**

The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table The result is NULL in the right side when there is no match.

SELECT *column\_name(s)*  
FROM *table1*  
LEFT JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

**RIGHT OUTER JOIN**

The RIGHT JOIN keyword returns all rows from the right table, with the matching rows in the left table. The result is NULL in the left side when there is no match.

SELECT *column\_name(s)*  
FROM *table1*  
RIGHT JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

FULL OUTER JOIN

The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table. The FULL OUTER JOIN keyword combines the result of both LEFT and RIGHT joins.

SELECT *column\_name(s)*  
FROM *table1*  
FULL OUTER JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

● **Cross Join**

“CROSS JOIN” or “CARTESIAN PRODUCT” combines all rows from both tables. Number of rows will be product of the number of rows in each table. In real life scenario I cannot imagine where we will want to use a Cartesian product. But there are scenarios where we would like permutation and combination probably Cartesian would be the easiest way to achieve it.

● **Self-Join**

If you want to join wo instances of the same table, you can use self-join.

SELECT a.column\_name, b.column\_name...

FROM table1 a, table1 b

WHERE a.common\_filed = b.common\_field;

**Views**

**● Understanding Views**

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

**● Need of Views**

Views, which are kind of virtual tables, allow users to do the following:

* Structure data in a way that users or classes of users find natural or intuitive.
* Restrict access to the data such that a user can see and (sometimes) modify exactly what they need and no more.
* Summarize data from various tables which can be used to generate reports.

**● Creating, Altering and Dropping Views**

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables, or another view

CREATE VIEW view\_name AS

SELECT column1, column2.....

FROM table\_name WHERE [condition];

**● Simple and Complex Views**

|  |  |  |
| --- | --- | --- |
| Feature | Simple Views | Complex Views |
| Number of tables | One | One or more |
| Contain functions | No | Yes |
| Contain groups of data | No | Yes |
| DML operations through a view | Yes | Not always |

**Functions**

**● Built-In Functions**

SQL has many built-in functions for performing processing on string or numeric data. Following is the list of all useful SQL built-in functions:

* SQL COUNT Function - The SQL COUNT aggregate function is used to count the number of rows in a database table.
* SQL MAX Function - The SQL MAX aggregate function allows us to select the highest (maximum) value for a certain column.
* SQL MIN Function - The SQL MIN aggregate function allows us to select the lowest (minimum) value for a certain column.
* SQL AVG Function - The SQL AVG aggregate function selects the average value for certain table column.
* SQL SUM Function - The SQL SUM aggregate function allows selecting the total for a numeric column.
* SQL SQRT Functions - This is used to generate a square root of a given number.
* SQL RAND Function - This is used to generate a random number using SQL command.
* SQL CONCAT Function - This is used to concatenate any string inside any SQL command.
* SQL Numeric Functions - Complete list of SQL functions required to manipulate numbers in SQL.
* SQL String Functions - Complete list of SQL functions required to manipulate strings in SQL.

**● Scalar Functions**

SQL scalar functions return a single value, based on the input value.

Useful scalar functions:

* UCASE() - Converts a field to upper case
* LCASE() - Converts a field to lower case
* MID() - Extract characters from a text field
* LEN() - Returns the length of a text field
* ROUND() - Rounds a numeric field to the number of decimals specified
* NOW() - Returns the current system date and time
* FORMAT () - Formats how a field is to be displayed

**● Aggregate Functions**

* AVG: Computes the average of a specific set of values, which can be an expression list or a set of data records in a table.
* SUM: Returns the sum of a specific set of values, which can be an expression list or a set of data records in a table.
* COUNT: Computes the number of data records in a table.
* MAX: Returns the maximum value from a specific set of values, which can be an expression list or a set of data records in a table.
* MIN: Returns the minimum value from a specific set of values, which can be an expression list or a set of data records in a table

**● Creating, Altering and Dropping Functions**

CREATE FUNCTION fn\_myfunc ()

ALTER FUNCTION [ schema\_name. ] function\_name

( [ { @parameter\_name [ AS ][ type\_schema\_name. ] parameter\_data\_type

[ = default ] }

[ ,...n ]

]

)

DROP FUNCTION [ IF EXISTS ] { [ schema\_name. ] function\_name } [ ,...n ]

[;]

**Stored Procedures**

**Types of Stored Procedures**

User-defined

A user-defined procedure can be created in a user-defined database or in all system databases except the **Resource** database. The procedure can be developed in either Transact-SQL or as a reference to a Microsoft .NET Framework common runtime language (CLR) method.

Temporary

Temporary procedures are a form of user-defined procedures. The temporary procedures are like a permanent procedure, except temporary procedures are stored in **tempdb**. There are two types of temporary procedures: local and global. They differ from each other in their names, their visibility, and their availability. Local temporary procedures have a single number sign (#) as the first character of their names; they are visible only to the current user connection, and they are deleted when the connection is closed. Global temporary procedures have two number signs (##) as the first two characters of their names; they are visible to any user after they are created, and they are deleted at the end of the last session using the procedure.

System:

System procedures are included with SQL Server. They are physically stored in the internal, hidden **Resource** database and logically appear in the **sys**schema of every system- and user-defined database

**● Creating , Altering and Dropping Procedure**

USE AdventureWorks

GO

CREATE PROCEDURE dbo.uspGetAddress

AS

SELECT \* FROM Person.Address

GO

ALTER PROCEDURE Purchasing.uspVendorAllInfo

@Product varchar(25)

DROP PROCEDURE <stored procedure name>;

GO

**● Optional Parameters**

ALTER PROCEDURE [dbo].[getcustomers]

{@lastname varchar(30)= null,

@firstname varchar(30) = null

}

**● Input and Output Parameters**

**Output:**

CREATE PROCEDURE dbo.uspGetAddressCount @City nvarchar(30), @AddressCount int OUTPUT

AS

SELECT @AddressCount = count(\*)

FROM AdventureWorks.Person.Address

WHERE City = @City

**Triggers**

**● Understanding Triggers**

For trigger you have to select trigger template. You have to specify on which object and which event will this method fire. The first attribute specifies the name, target (on which the trigger will fire) and event ( insert ,update or delete).

***[SqlTrigger (Name="Trigger1", Target="Table1", Event="FOR INSERT")]***

**● Stored Procedures VS Functions VS Triggers**

a strored procedure  is a assembly of lines of sql request  to do a treatment for a web application (asp.net) or window form etc ... it contain variables ,for, if etc

a trigger is like a behaviour made by a table after insert or after update etc

a function is a treatment that return a value to use it after that anywhere like conversion of dates etc

**● Types of Triggers**

There are two types of triggers: -

**INSTEAD OF triggers**

INSTEAD OF triggers fire in place of the triggering action. For example, if an INSTEADOF UPDATE trigger exists on the Sales table and an UPDATE statement is executed against the Sales table, the UPDATE statement will not change a row in the sales table. Instead, the UPDATE statement causes the INSTEAD OF UPDATE trigger to be executed, which may or may not modify data in the Sales table.

**AFTER triggers**

AFTER triggers execute following the SQL action, such as an insert, update, or delete. This is the traditional trigger which existed in SQL SERVER.INSTEAD OF triggers gets executed automatically before the Primary Key and the Foreign Key constraints are checked, whereas the traditional AFTER triggers gets executed after these constraints are checked. Unlike AFTER triggers, INSTEAD OF triggers can be created on views

**● Creating, Altering and Dropping Triggers**

Create the trigger on the instance of SQL Server.

tr.Create();

ALTER TRIGGER *trigger-name*

[ACTIVE | INACTIVE]

{BEFORE | AFTER} *<actions>*

[POSITION *number*]

AS

*<trigger\_body>*

Remove the trigger.

tr.Drop();

ADO.NET

Introduction to ADO.NET

● Understanding ADO.NET

● ADO VS ADO.NET

Following are some major differences between both

* As in classic ADO we had client and server side cursors they are no more present in ADO.NET. Note it's a disconnected model so they are no more applicable.
* Locking is not supported due to disconnected model.
* All data is persisted in XML as compared to classic ADO where data was persisted in Binary format also.

**● Advantage of ADO.NET**

**● ADO.NET Version History**

**● ADO.NET Architecture**

The most important section in ADO.NET architecture is “Data Provider”. Data Provider provides access to data source (SQL SERVER, ACCESS , ORACLE).In short it provides object to achieve functionalities like opening and closing connection , retrieve data and update data. In the below figure you can see the four main sections of a data provider :-

* Connection.
* Command object (This is the responsible object to use stored procedures)
* Data Adapter (This object acts as a bridge between data store and dataset).
* Data reader (This object reads data from data store in forward only mode). Dataset object represents disconnected and cached data. If you see the diagram it is not indirect connection with the data store (SQL SERVER, ORACLE etc) rather it talks with Data adapter, who is responsible for filling the dataset. Dataset can have one or more Data table and relations.

“DataView” object is used to sort and filter data in Datatable.

**● Connected and Disconnected Architecture**

**● ADO.NET Objects**

**● .NET Data Providers**

**Connection & Command**

**● Connection Classes**

**● Command Objects**

**● Creating Commands**

**● Executing Commands**

**● Parameterized Queries**

**● Command Types**

**Data Readers & Connected Access**

**● Understanding DataReader**

**● Reading Data using DataReader**

**● ExecuteReader Options**

**Data Sets & Disconnected Access**

**● Understanding DataSet**

**● Understanding DataAdapter**

**● Understanding DataTable, DataColumn, DataRow**

**Transaction**

**● Understanding Transaction**

**● Transaction ACID properties**

**● Implementing Transaction**

**● Commit and Rollback**

**Data Controls**

**● Data Source Controls**

**● Choosing Data Binding Controls**

**● GridView**

**● DataList**

**● Paging, Sorting**

**AJAX**

**Introduction to AJAX**

**● Understanding AJAX**

* Update a web page without reloading the page
* Request data from a server - after the page has loaded
* Receive data from a server - after the page has loaded
* Send data to a server - in the background

AJAX is based on internet standards, and uses a combination of:

* XMLHttpRequest object (to retrieve data from a web server)

The XMLHttpRequest object is used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

* JavaScript/DOM (to display/use the data)

XMLHttpRequest Object Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| new XMLHttpRequest() | Creates a new XMLHttpRequest object |
| abort() | Cancels the current request |
| getAllResponseHeaders() | Returns header information |
| getResponseHeader() | Returns specific header information |
| open(*method,url,async,user,psw*) | Specifies the request  *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password |
| send() | Sends the request to the server Used for GET requests |
| send(*string*) | Sends the request to the server. Used for POST requests |
| setRequestHeader() | Adds a label/value pair to the header to be sent |

|  |  |
| --- | --- |
| **Property** | **Description** |
| onreadystatechange | Defines a function to be called when the readyState  property changes |
| readyState | Holds the status of the XMLHttpRequest. 0: request not initialized  1: server connection established 2: request received  3: processing request  4: request finished and response is ready |
| responseText | Returns the response data as a string |
| responseXML | Returns the response data as XML data |
| status | Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" |
| statusText | Returns the status-text (e.g. "OK" or "Not Found") |

XML Http Request Object Properties:

* Script manager  
  Manages ASP.NET Ajax script libraries and script files, partial-page rendering, and client proxy class generation for Web and application services.

**Namespace:**  System.Web.UI  
**Assembly:** System.Web.Extensions (in System.Web.Extensions.dll)

Syntax: public class ScriptManager : Control, IPostBackDataHandler, IPostBackEventHandler

 the ScriptManager control enables you to do the following:

* Register script that is compatible with partial-page updates. In order to manage dependencies between your script and the core library, any script that you register is loaded after the Microsoft Ajax Library script.
* Specify whether release or debug scripts are sent to the browser.
* Provide access to Web service methods from script by registering Web services with the ScriptManager control.
* Provide access to ASP.NET authentication, role, and profile application services from client script by registering these services with theScriptManager control.
* Enable culture-specific display of ECMAScript (JavaScript) , , and functions in the browser.
* Access localization resources for embedded script files or for stand-alone script files by using the ResourceUICultures property of theScriptReference control.
* Register server controls that implement the IExtenderControl or IScriptControl interfaces with the ScriptManager control so that script required by client components and behaviors is rendered.

● Update Panel  
Enables sections of a page to be partially rendered without a postback.

**Namespace:**  [System.Web.UI](https://msdn.microsoft.com/en-us/library/system.web.ui(v=vs.110).aspx)  
**Assembly:** System.Web.Extensions (in System.Web.Extensions.dll)

Syntax: public class UpdatePanel : Control, IAttributeAccessor  
UpdatePanel controls are a central part of AJAX functionality in ASP.NET. They are used with the [ScriptManager](https://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager(v=vs.110).aspx) control to enable partial-page rendering. Partial-page rendering reduces the need for synchronous postbacks and complete page updates when only part of the page has to be updated. Partial-page rendering improves the user experience because it reduces the screen flicker that occurs during a full-page postback and improves Web page interactivity.

The <asp:UpdatePanel> tag has two childtags - the **ContentTemplate** and the **Triggers** tags. The ContentTemplate tag is required, since it holds the content of the panel. The content can be anything that you would normally put on your page, from literal text to web controls. The Triggers tag allows you to define certain triggers which will make the panel update it's content.

You can also use UpdatePanel controls in the following scenarios:

* In user controls.
* On master and content pages.
* Nested inside other UpdatePanel controls.
* Inside templated controls such as the [GridView](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview(v=vs.110).aspx) or [Repeater](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.repeater(v=vs.110).aspx) controls.

● Remote Method Calls

● AJAX Control Toolkit

Accordion

AlwaysVisibleControl

Animation

AutoComplete

Calendar

CascadingDropDown

CollapsiblePanel

ConfirmButton

DragPanel

DropDown

DropShadow

DynamicPopulate

FilteredTextBox

HoverMenu

ListSearch

MaskedEdit

ModalPopup

MutuallyExclusiveCheckBox

NoBot

NumericUpDown

PagingBulletedList

PasswordStrength

PopupControl

Rating

ReorderList

ResizableControl

RoundedCorners

Slider

SlideShow

Tabs

TextBoxWatermark

ToggleButton

UpdatePanelAnimation

ValidatorCallout

WCF

Introduction to WCF

● Introduction to WCF

WCF stands for Windows Communication Foundation. It is a framework for building, configuring, and deploying network-distributed services.

A WCF application consists of three components:

* WCF service,
* WCF service host, and
* WCF service client.

WCF platform is also known as the Service Model.

**Metadata**

This is a significant concept of WCF, as it facilitates easy interaction between a client application and a WCF service. Normally, metadata for a WCF service is generated automatically when enabled, and this is done by inspection of service and its endpoints.

**Advantages of WCF**

* It is interoperable with respect to other services. This is in sharp contrast to .NET Remoting in which both the client and the service must have .Net.
* WCF services offer enhanced reliability as well as security in comparison to ASMX (Active Server Methods) web services.
* Implementing the security model and binding change in WCF do not require a major change in coding. Just a few configuration changes is required to meet the constraints.
* WCF has built-in logging mechanism whereas in other technologies, it is essential to do the requisite coding.
* WCF has integrated AJAX and support for JSON (JavaScript object notation).
* It offers scalability and support for up-coming web service standards.

**Some major differences that exist between WCF and a Web service which are listed below.**

* **Attributes** - WCF service is defined by ServiceContract and OperationContract attributes, whereas a web service is defined by WebService and WebMethod attributes.
* **Protocols** - WCF supports a range of protocols, i.e., HTTP, Named Pipes, TCP, and MSMQ, whereas a web service only supports HTTP protocol.
* **Hosting Mechanisms** - Various activation mechanisms are there for WCF hosting, i.e., IIS (Internet Information Service), WAS (Windows Activation Service), Self-hosting and Windows Service, but a web service is hosted only by IIS.
* **Services** - WCF supports a robust security, trustworthy messaging, transaction and interoperability, while a web service only supports security services.
* **Serializer** - WCF Supports DataContract serializer by employing System.Runtime.Serialization, whereas a web service supports XML serializer by making use of System.Xml.Serialization.
* **Tools** - ServiceMetadata tool (svcutil.exe) is used for client generation for a WCF service, while WSDL.EXE tool is used for generating the same for a web service.
* **Exception Handling** - In WCF, unhandled exceptions are handled in a better way by making use of FaultContract. They do not return to the client like in a web service as SOAP faults.
* **Hash Table** - It is possible to serialize a Hash Table in WCF, but this is not the case in a web service.
* **Bindings** - WCF supports several types of bindings like BasicHttpBinding, WSDualHttpBinding, WSHttpBinding, etc., while a web service supports only SOAP or XML.
* **Multithreading** - WCF supports multithreading by using the ServiceBehavior Class, whereas this is not supported in a web service.
* **Duplex Service Operations** - WCF supports duplex service operations apart from supporting one-way and request-response service operations, whereas a web service does not support duplex service operations.

For developing a WCF service application, there are mainly two tools – Microsoft Visual Studio and CodePlex. Microsoft Visual Studio is a complete package of development tools, necessary for developing a large number of diverse applications like ASP.NET web applications, desktop applications, mobile applications, and many more.

Microsoft Visual Studio uses the .NET framework functionality. CodePlex on the other hand is an open-source project hosting site of Microsoft that offers several free tools for WCF service application development.

Tools offered by codeplex are wscf.blue, WCFProxyGenerator, WCF Mock.

● Understanding WCF ABC (Address, Binding, Contract)

● WCF Addresses

Address specifies the exact location to receive the messages and is specified as a Uniform Resource Identifier (URI). It is expressed as scheme://domain [: port]/[path]. Take a look at the address mentioned below:

net.tcp://localhost:9000/ServiceA

Here, 'net.tcp' is the scheme for the TCP protocol. The domain is 'localhost' which can be the name of a machine or a web domain, and the path is 'ServiceA'.

● WCF Bindings

It defines the way an endpoint communicates. It comprises of some binding elements that make the infrastructure for communication. For example, a binding states the protocols used for transport like TCP, HTTP, etc., the format of message encoding, and the protocols related to security as well as reliability.

Basic binding is offered by the BasicHttpBinding class. It uses the HTTP protocol to transport and represent a WCF service as an ASP.NET web service (ASMX web service), so that old clients who use ASMX web services can consume the new services conveniently.

Basic binding is set as default binding in a WCF web service enabled by Silverlight and is a standard binding for communications in web service style. It does not support reliable messaging.

Other Types of Service Bindings

* **TCP Binding** - Provided by the NetTCPBinding class, this binding makes use of the TCP protocol for communication within the same network and does the message encoding in binary format. This binding is considered as the most reliable in contrast to others.
* **WS Dual Binding** - This type of binding is more like WSHttpBinding with the only exception that it facilitates bidirectional communication, i.e., messages can be sent and received by both clients and services. It is offered by the WSDualHttpBinding class.
* **Web binding** - Web binding is designed to represent WCF services in the form of HTTP requests by the use of HTTP-GET, HTTP-POST, etc. It is offered by the WebHttpBinding class and is used commonly with social networks.
* **MSMQ Binding** - It is offered by the NetMsmqBinding class and is used to provide solutions in case the service processes a message at a distinct time than that sent by the client. MSMQ binding makes use of MSMQ for transportation and provides support to detached message queued. MSMQ is an implementation for message queuing offered by Microsoft.
* **Federated WS Binding** - It is a specific form of WS binding and offers support for federated security. It is offered by the WSFederationHttpBinding class.
* **Peer Network Binding** - Offered by the NetPeerTCPBinding class, it is mainly used in file sharing systems. It uses TCP protocol but makes use of peer networking as transport. In this networking, each machine (node) acts as a client and a server to the other nodes. Peer network binding is used in file sharing systems like torrent.
* **MSMQ Integration Binding** - Offered by the Msmq Integration Binding class, it helps communicate with existing systems that communicate via MSMQ (Microsoft Message Queuing).

● WCF Contracts

It is a collection of operations that specifies what functionality the endpoint exposes to the clinet. It generally consists of an interface name.

The contracts layer is just next to the application layer and contains information similar to that of a real-world contract that specifies the operation of a service and the kind of accessible information it will make. Contracts are basically of four types discussed below in brief:

* **Service contract** - This contract provides information to the client as well as to the outer world about the offerings of the endpoint, and the protocols to be used in the communication process.
* **Data contract** - The data exchanged by a service is defined by a data contract. Both the client and the service has to be in agreement with the data contract.
* **Message contract** - A data contract is controlled by a message contract. It primarily does the customization of the type formatting of the SOAP message parameters. Here, it should be mentioned that WCF employs SOAP format for the purpose of communication. SOAP stands for Simple Object Access Protocol.
* **Policy and Binding** - There are certain pre-conditions for communication with a service, and such conditions are defined by policy and binding contract. A client needs to follow this contract.

● WCF Endpoints

It defines the address where a message is to be sent or received. It also specifies the communication mechanism to describe how the messages will be sent along with defining the set of messages. A structure of an endpoint comprises of the following parts:

Address

Binding

Contracts

There are three diff types of endpoints: **Standard endpoints, infrastructure endpoints, application endpoints.**

Building & Consuming WCF Service

● Creating WCF Service

1. On the **File** menu, point to **New** and then click **Project**.
2. In the **New Project** dialog box, expand the **Visual Basic** or **Visual C#** node and click **WCF**, followed by **WCF Service Library**. Click **OK** to open the project.

This creates a working service that can be tested and accessed. The following two steps demonstrate how you might modify the default method to use a different data type. In a real application, you would also add your own functions to the service.

1. In **Solution Explorer**, double-click IService1.vb or IService1.cs and find the following line:

C# [OperationContract]

string GetData(int value);

Change the type for the value parameter to String:

C# [OperationContract]

string GetData(string value);

In the above code, note the <OperationContract()> or [OperationContract] attributes. These attributes are required for any method exposed by the service.

4. In **Solution Explorer**, double-click Service1.vb or Service1.cs and find the following line:

C# public string GetData(int value)

{

return string.Format("You entered: {0}", value);

}

Change the type for the value parameter to String:

C# public string GetData(string value)

{

return string.Format("You entered: {0}", value);

}

#### **To test a WCF service**

1. Press **F5** to run the service. A **WCF Test Client** form will be displayed and it will load the service.
2. In the **WCF Test Client** form, double-click the **GetData()** method under **IService1**. The **GetData** tab will be displayed.
3. In the **Request** box, select the **Value** field and type Hello.
4. Click the **Invoke** button. If a **Security Warning** dialog box is displayed, click **OK**. The result will be displayed in the **Response** box.
5. On the **File** menu, click **Exit** to close the test form.

Accessing the Service

**To reference a WCF service**

1. On the **File** menu, point to **Add** and then click **New Project**.
2. In the **New Project** dialog box, expand the **Visual Basic** or **Visual C#** node and select **Windows**, and then select **Windows Forms Application**. Click **OK** to open the project.
3. Right-click **WindowsApplication1** and click **Add Service Reference**. The **Add Service Reference** dialog box will appear.
4. In the **Add Service Reference** dialog box, click **Discover**.

**Service1** will be displayed in the **Services** pane.

1. Click **OK** to add the service reference.

#### **To build a client application**

1. In **Solution Explorer**, double-click **Form1.vb** or **Form1.cs** to open the Windows Forms Designer if it is not already open.
2. From the **Toolbox**, drag a TextBox control, a Label control, and a Button control onto the form.
3. Double-click the Button, and add the following code in the Click event handler:

C# private void button1\_Click(System.Object sender, System.EventArgs e)

{

ServiceReference1.Service1Client client = new

ServiceReference1.Service1Client();

string returnString;

returnString = client.GetData(textBox1.Text);

label1.Text = returnString;

}

1. In **Solution Explorer**, right-click **WindowsApplication1** and click **Set as StartUp Project**.
2. Press **F5** to run the project. Enter some text and click the button. The label will display "You entered:" and the text that you entered.

● **Configuring Endpoints**

Endpoints provide clients with access to the functionality a Windows Communication Foundation (WCF) service offers. You can define one or more endpoints for a service by using a combination of relative and absolute endpoint addresses, or if you do not define any service endpoints, the runtime provides some by default for you. An endpoint address can be created in code with the Endpoint Address class. The URI specified for the endpoint address can be a fully-qualified path or a path that is relative to the service's base address.

● **Creating Service Client**

A client application uses the WCF client proxy to communicate with the service. Client applications usually import a service's metadata to generate WCF client code that can be used to invoke the service.

The basic steps for creating a WCF client include the following:

1. Compile the service code.
2. Generate the WCF client proxy.
3. Instantiate the WCF client proxy.

● **Consuming WCF Service**

WCF services allow other applications to access or consume them. A WCF service can be consumed by many ways depending on the hosting type. Here, we are explaining the step-by-step method to consume a WCF service for each of the following popular hosting options:

* Consuming WCF Service hosted in IIS 5/6
* Consuming WCF Service that is self-hosted
* Consuming WCF Service hosted in Windows Activation Service
* Consuming WCF Service hosted in Windows Service

● **WCF Hosting**

After creating a WCF service, the next step is to host it so that the client applications can consume it. This is known as WCF service hosting. A WCF service can be hosted by using any of the four ways given below:

* **IIS Hosting** - IIS stands for Internet Information Services. Its working model is similar to that of ASP.NET while hosting a WCF service. The best feature of IIS hosting is that the service activation is processed automatically. IIS hosting also offers process health monitoring, idle shutdown, process recycling, and many more features to facilitate a WCF service hosting.
* **Self-Hosting** - When a WCF service is hosted in a managed application, it is known as self-hosting. It requires a developer to write the requisite coding for ServiceHost initialization. In self-hosting, a WCF service can be hosted in a variety of applications like Console application, Windows form, etc.
* **WAS Hosting** - Hosting a WCF service in Windows Activation Service (WAS) is most advantageous because of its features such as process recycling, idle time management, common configuration system, and support for HTTP, TCP, etc.
* **Windows Service Hosting** - For local system clients, it is best to host WCF service as a window service and this is known as Window Service Hosting. All the versions of Windows support this type of hosting and here, the Service Control Manager can control the process lifecycle of the WCF service.

ASP.NET MVC

Introduction to ASP.NET MVC

● Understanding ASP.NET MVC

The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the **model**, the **view**, and the **controller**. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects. The ASP.NET MVC framework is a lightweight, highly testable presentation framework that (as with Web Forms-based applications) is integrated with existing ASP.NET features, such as master pages and membership-based authentication.

Razor View Engine:

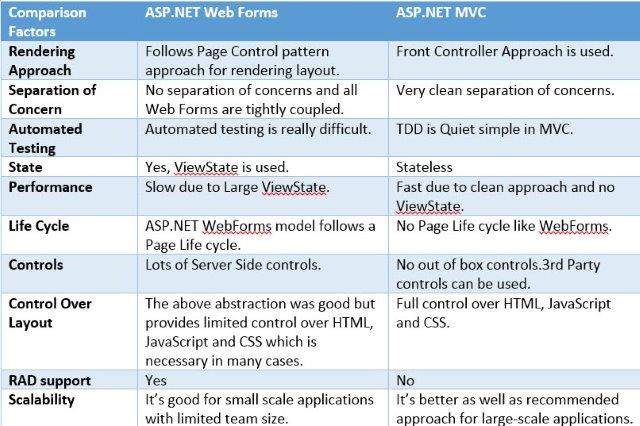
Razor is the first major update to render HTML in ASP.Net MVC 3. Razor was designed specifically for view engine syntax. Main focus of this would be to simplify and code-focused templating for HTML generation. Below is the sample of using Razor:

@model ASP.Net MVCMusicStore.Models.Customer

@{ViewBag.Title = "Get Customers";}

< div class="cust"> <h3><em>@Model.CustomerName</<em> </<h3><div>

● ASP.NET VS MVC VS ASP.NET Web Form VS ASP.NET MVC



● ASP.NET MVC Version History

Mobile templates Added ASP.NET Web API template for creating REST based services. Asynchronous controller task support. Bundling of the java scripts. Segregating the configs for ASP.Net MVC routing, Web API, Bundle etc.

● Understanding Model, View and Controller

Model: It is basically a business entity which is used to represent the application data. Controller: The Request which is sent by the user always scatters through controller and its responsibility is to redirect to the specific view using View () method. View: it's the presentation layer of ASP.Net MVC.

**Model:** The Model component corresponds to all the data related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

**View:** The View component is used for all the UI logic of the application. For example, the Customer view would include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

**Controller:** Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller would handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller would be used to view the Customer data.

● Advantages of ASP.NET MVC

* Provides a clean separation of concerns among UI (Presentation layer), model (Transfer objects/Domain Objects/Entities) and Business Logic (Controller).
* Easy to UNIT Test.
* Improved reusability of model and views. We can have multiple views which can point to the same model and vice versa.
* Improved structuring of the code.

● Design Pattern VS Architectural Pattern

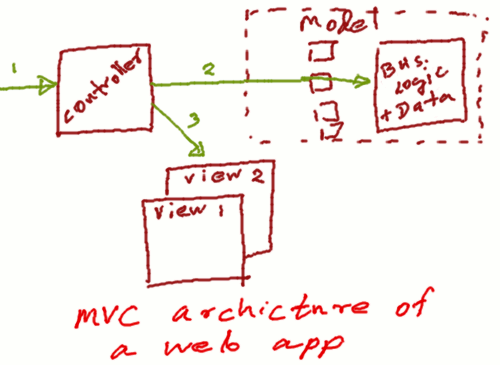
**Architectural pattern:** Concerned with the subsystems of an application with their relationships and collaborations with each other.

**Design pattern:** Concerned with the (further small) components of a subsystem with their relationships and collaborations with each other.

**MVC as an architectural pattern**

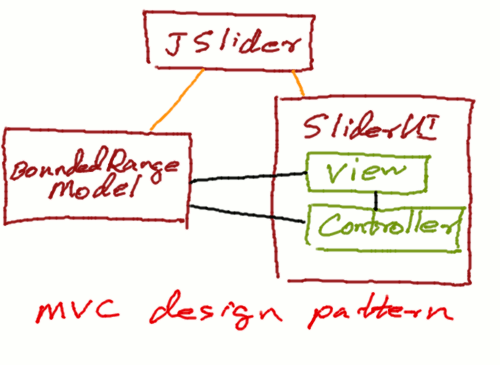
Here the core functionality and data of an application are abstracted as the model. You have one or more views and one or more controllers collectively representing the user interface. A view is responsible for displaying information to the user. A controller is responsible for accepting user inputs, interpret them to understand what should be done and act accordingly (for example update data or invoke a functionality of the model).

A real world example is the implementation of a web application that follows the MVC architectural pattern. Many such web application frameworks exists. The key is that an application is broken down into components (call them subsystems) by the MVC pattern.



**MVC as a design pattern**

Consider the many UI widget libraries available out there where MVC is used as a design pattern to implement widgets. Take for example the JSlider class (in Java Swing library). A JSlider instance represents a slider widget where a user can move the knob of the slider to change its position. JSlider is implemented as follows:



● Layer VS Tier

● MVC pattern VS 3Tier Architecture

**The MVC pattern:**

* **Models:** These represent "stuff" in your application. This layer has gotten a little fuzzy in recent years, as I will explain later.
* **Views:** The user interface. The thing the user interacts with.
* **Controllers:** The programming code that responds to the user and to changes in the model layer

**The 3-Tier Architecture:**

With the 3-tier architecture, you have layers with different responsibilities.

* **User Services:** (or "services" in general) This layer is more about coordinating the retrieval and modifications of the "model" layer. Complex, multi-step actions get performed here
* **Business Layer:** This represents the business rules etched into programming code. What "The Business" wants is enforced in this layer.
* **Data Access Layer:** One or more classes responsible for accessing a persistent data store.

Visual Studio Project Templates

**● Understanding Visual Studio ASP.NET MVC Templates**

|  |  |
| --- | --- |
| **Project Template** | **More Information** |
| ASP.NET Web Forms Application | Use this project template to create a web application that is based on ASP.NET Web Forms pages and that includes the following functionality. You can choose not to use any of these features when they are not required for your application.   * A master page. * A cascading style sheet. * Login security that uses the ASP.NET membership system. * Ajax scripting that uses jQuery. * Navigation that uses a menu control.   By default, the ASP.NET Web Application project template includes the following:   * Folders to contain membership pages, client script files, and cascading style sheet files. * A data folder (App\_Data), which is granted permissions that allow ASP.NET to read and write to it at run time. * A master page (the Site.master file). * Web pages named Default.aspx, Contact.aspx, and About.aspx. These content pages are based on the default master. * A global application class (Global.asax file). * A Web.config file. * A Packages.config file. |
| ASP.NET MVC 3 Web Application | Use this project template to create web applications that use a model-view-controller pattern, using the ASP.NET MVC 3 release. The MVC pattern helps separate the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements. In addition, this project template promotes test-driven development (TDD). |
| ASP.NET MVC 4 Web Application | Use this project template to create web applications that use a model-view-controller pattern, using the ASP.NET MVC 4 release. |
| ASP.NET Empty Web Application | This template creates an ASP.NET web application that includes a Web.config file, but no other files. Use this project template when you do not require the functionality built into the standard template. |
| ASP.NET Dynamic Data Entities Web Application | Use this project template to create a Dynamic Data web application that uses the ADO.NET Entity Framework. This type of Dynamic Data web application can target any relational database.  Dynamic Data automatically generates a functional web application for viewing and editing data, based on the schema of the data. |
| ASP.NET AJAX Server Control | Use this project template to create a web server control for ASP.NET Web Forms pages that incorporates Ajax functionality. The control consists of server and client code that work together to produce rich client behavior. |
| ASP.NET AJAX Server Control Extender | Use this project template to create an Ajax component that enhances the client capabilities of standard ASP.NET web server controls. |
| ASP.NET Server Control | Use this project template to create a custom ASP.NET web server control that can be used in ASP.NET Web Forms pages. |

**● Creating an ASP.NET MVC project**

Steps to create first MVC Application

**Step 1:** Start your Visual Studio and select File->New->Project. Select Web->ASP.NET MVC Web Application and name this project as FirstMVCApplication. Select the Location as C:\MVC. Click OK.

**Step 2:** This will open the Project Template option. Select Empty template and View Engine as Razor. Click OK.

By this, Visual Studio will create our first MVC project like this (shown in screenshot):

**Step 3:** Now we will create the first Controller in our application. Controllers are just simple C# classes which contains multiple public methods, known as action methods. To add a new Controller, right click the Controllers folder in our project and select Add->Controller. Name the Controller as HomeContoller and click Add.

This will create a class file HomeController.cs under the Controllers folder with the following default code.

using System;

using System.Web.Mvc;

namespace FirstMVCApplication.Controllers

{

public class HomeController : Controller

{

public ViewResult Index()

{

return View();

}

}

}

The above code basically defines a public method Index inside our HomeController and returns a ViewResult object. In the next steps, we will learn how to return a View using the ViewResult object.

**Step 4:** Now we will add a new View to our Home Controller. To add a new View, right click view folder and click Add->View.

Name the new View as Index and View Engine as Razor (SCHTML). Click Add.

This will add a new cshtml file inside Views/Home folder with the following code:

@{

Layout = null;

}

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<div>

</div>

</body>

</html>

**Step 5:** Modify the above View's body content with the following code:

<body>

<div>

Welcome to My First MVC Application (<b>From Index View</b>)

</div>

</body>

**Step 6:** Now run the application. This will give you the following output in the browser. This output is rendered based on the content in our View file. The application first calls the Controller which in turn calls this View and gives the output.

**Step 7:** In Step 6, the output we received was based on the content of our View file and had no interaction with the Controller. Moving a step forward, now we will now create a small example to display a Welcome message with current time using interaction of View and Controller.

MVC uses the ViewBag object to pass data between Controller and View. Open the HomeController.cs and edit the Index function to the following code.

public ViewResult Index()

{

int hour = DateTime.Now.Hour;

ViewBag.Greeting =

hour < 12

? "Good Morning. Time is" + DateTime.Now.ToShortTimeString()

: "Good Afternoon. Time is " + DateTime.Now.ToShortTimeString();

return View();

}

In the above code, we set set the value of the Greeting attribute of the ViewBag object. The code checks the current hour and returns the Good Morning/Afternoon message accordingly using return View() statement. Note that here Greeting is just an example attribute that we have used with ViewBag object. You can use any other attribute name in place of Greeting.

**Step 8:** Now open the Index.cshtml and copy the following code in the body section:

<body>

<div>

@ViewBag.Greeting (<b>From Index View</b>)

</div>

</body>

In the above code, we are accessing the value of Greeting attribute of the ViewBag object using @ (which would be set from the Controller).

**Step 9:** Now run the application again. This time our code will run the Controller first, set the ViewBag and then render it using the View code.

● Understanding ASP.NET MVC project folder structure

In your visual studio, open File->New->Project and select ASP.NET MVC Application. Name it as MVCFolderDemo.

Click OK. In the next window, select Internet Application as the Project Template and click OK.

This will create a sample MVC application as shown below:

Note that the filers present in this project are coming out of the default template that we have selected. These may change slightly as per different versions.

Controllers Folder

* This folder will contain all the Controller classes. MVC requires name of all the controller files to end with Controller.
* In our example, the Controllers folder contains two class files: AccountController and HomeController.

Models Folder

* This folder will contain all the Model classes which are used to work on application data.
* In our example, the Models folder contains AccountModels. You can open and look at the code in this file to see how the data model is created for managing accounts in our example.

Views Folder

* This folder stores the HTML files related to application display and user interface.
* It contains one folder for each controller.
* In our example, you will see three sub-folders under Views namely Account, Home and Shared which contains html files specific to that view area.

App\_Start Folder

* This folder contains all the files which are needed during the application load.
* For e.g., the RouteConfig file is used to route the incoming URL to the correct Controller and Action

Content Folder

* This folder contains all the static files such as css, images, icons, etc.
* The Site.css file inside this folder is the default styling that the application applies.

Scripts Folder

* This folder stores all the JS files in the project. By default, Visual Studio adds MVC, jQuery and other standard JS libraries.

● Understanding configuration files

The web.config file exists in the Views folders to prevent access to your views by any means other than your controller. In the MVC design pattern, controllers are supposed to route requests and return a rendered view to the calling client.

means *localhost9999://Home/Index.cshtml* should not be directly accessible.

Controller & Actions

**● Understanding Controller**

Asp.net MVC Controllers are responsible for controlling the flow of the application execution. When you make a request (means request a page) to MVC applications, a controller is responsible for returning the response to that request. A controller can have one or more actions. A controller action can return different types of action results to a particular request.

Controller is responsible for controlling the application logic and acts as the coordinator between the View and the Model. The Controller receives input from users via the View, then process the user's data with the help of Model and passing the results back to the View.

**● Creating Controller**

For creating a Controller, create an MVC Empty Application and then right-click on the Controller folder in your MVC application and select the menu option Add->Controller. After selection the Add Controller dialog is being displayed. Name the Controller as DemoController.

**● Understanding Actions**

Actions are the methods in Controller class which is responsible for returning the view or json data. Action will mainly have return type : "ActionResult" and it will be invoked from method : "InvokeAction()" called by controller.

**● Actions and Non Actions Methods**

By default, the MVC framework treats all public methods of a controller class as action methods. If your controller class contains a public method and you do not want it to be an action method, you must mark that method with the NonAction Attribute attribute.

**● Understanding Action Results**

Most action methods return an instance of a class that derives from [ActionResult](https://msdn.microsoft.com/en-us/library/system.web.mvc.actionresult(v=vs.98).aspx). The **Action Result** class is the base for all action results. However, there are different action result types, depending on the task that the action method is performing. For example, the most common action is to call the [View](https://msdn.microsoft.com/en-us/library/system.web.mvc.controller.view(v=vs.98).aspx) method. The **View** method returns an instance of the [ViewResult](https://msdn.microsoft.com/en-us/library/system.web.mvc.viewresult(v=vs.98).aspx) class, which is derived from **ActionResult**.

The following table shows the built-in action result types and the action helper methods that return them.

|  |  |  |
| --- | --- | --- |
| **Action Result** | **Helper Method** | **Description** |
| ViewResult | View | Renders a view as a Web page. |
| PartialViewResult | PartialView | Renders a partial view, which defines a section of a view that can be rendered inside another view. |
| RedirectResult | Redirect | Redirects to another action method by using its URL. |
| RedirectToRouteResult | RedirectToAction  RedirectToRoute | Redirects to another action method. |
| ContentResult | Content | Returns a user-defined content type. |
| JsonResult | Json | Returns a serialized JSON object. |
| JavaScriptResult | JavaScript | Returns a script that can be executed on the client. |
| FileResult | File | Returns binary output to write to the response. |
| EmptyResult | (None) | Represents a return value that is used if the action method must return a **null** result (void). |

View:

View Model is a plain class with properties, which is used to bind it to strongly typed view. Views are the components involved with application's User Interface. These Views are generally bind from the model data and have extensions like html, aspx, cshtml, vbhtml, etc. In our First MVC Application, we had used Views with Controller to display data to the final user. For rendering these static and dynamic content to the browser, MVC Framework utilizes View Engines. View Engines are basically markup syntax implementation which are responsible for rendering the final HTML to the browser.

● Types of Views

The MVC Framework comes with two built-in view engines:

1. **Razor Engine:** Razor is a markup syntax that enables the server side C# or VB code into web pages. This server side code can be used to create dynamic content when the web page is being loaded. Razor is an advanced engine as compared to ASPX engine and was launched in the later versions of MVC.

**System.**web.mvc.razor is namespace for razor view

|  |  |
| --- | --- |
| **Class** | **Description** |
| System_CAPS_pubclass | [MvcCSharpRazorCodeParser](https://msdn.microsoft.com/en-us/library/system.web.mvc.razor.mvccsharprazorcodeparser(v=vs.118).aspx) | Compiles ASP.NET Razor views into classes. |
| System_CAPS_pubclass | [MvcVBRazorCodeParser](https://msdn.microsoft.com/en-us/library/system.web.mvc.razor.mvcvbrazorcodeparser(v=vs.118).aspx) | Extends the **VBCodeParser** class by adding support for the **@model** keyword. |
| System_CAPS_pubclass | [MvcWebPageRazorHost](https://msdn.microsoft.com/en-us/library/system.web.mvc.razor.mvcwebpagerazorhost(v=vs.118).aspx) | Configures the ASP.NET Razor parser and code generator for a specified file. |

2. **ASPX Engine:** ASPX or the Web Forms engine is the default view engine that is included in the MVC Framework since the beginning. Writing code with this engine is very similar to writing code in ASP.NET Web Forms.

Following are small code snippets comparing both Razor and ASPX engine.

**Razor:**

@Html.ActionLink("Create New", "UserAdd")

**ASPX:**

<% Html.ActionLink("SignUp", "SignUp") %>

Out of these two, Razor is more advanced View Engine as it comes with compact syntax, test driven development approaches, and better security features. We will use Razor engine in all our examples since it is the most dominantly used View engine.

These View Engines can be coded and implemented in following two types:

* Strongly typed
* Dynamic typed

Strongly typed Views is considered as a better approach since we already know what data is being passed as the Model unlike dynamic typed Views in which the data gets bind at runtime and may lead to runtime errors if something changes in the linked model.

● Creating Standard View

● Creating Layout Page

Layouts are used in MVC to provide consistent look and feel on all the pages of our application. It is same as defining the Master Pages but MVC gives some more functionalities. In this layout, we are using an HTML helper method and some other system defined methods so let's see these methods one by one.

* **Url.Content():** This method specifies the path of any file that we are using in our View code. It takes the virtual path as input and returns absolute path.
* **Html.ActionLink():** Used to render HTML links that links to action of some controller. The first parameter specifies the display name, the second parameter specifies the Action name and the third parameter specifies the Controller name.
* **RenderSection():** Specifies the name of section that we want to display at that location in the template
* **RenderBody():** Renders the actual body of the associated View.

**● Communication between Controller and View**

Some technologies, such as C#, use MVVM because there is no link between the View and any other, everything goes through the service locator, binding the variables.

On pure MVC, the View talks directly with the Model and vice-versa. The Controller is only there when any change arises.

And then, there is the one called PAC (Presentation Abstraction Control). In this architecture, the View and the Model don't talk to each other. The Controller is the only one allowed to do anything with either the View or the Model. People often confuse this with MVC.

Model & Validation

The model is responsible for managing the data of the application. It responds to the request from the view and it also responds to instructions from the controller to update itself.

Model classes can either be created manually or generated from database entities.

Model objects are the parts of the application that implement the domain logic, also known as business logic. Domain logic handles the data that is passed between the database and the UI. For example, in an inventory system, the model keeps track of the items in storage and the logic to determine whether an item is in stock. In addition, the model would be the part of the application that updates the database when an item is sold and shipped out of the warehouse. Often, the model also stores and retrieves model state in a database.

● Creating Model  
        Step1:- Create a simple class file  
        Step2:- Define the controller with action  
        Step3:- Create strongly typed view using the class  
        Step 4 :- Run your application

● Understanding ASP.NET MVC Validation

● Need of Server Side and Client Side Validation

In the server-side validation (ASP.NET MVC Server-Side Validation), the page must be submitted via a postback to be validated on the server and if the model data is not valid, then the server sends a response back to the client. With client-side validation, the input data is checked as soon as they are submitted, so there is no postback to the server and there is no page refresh.

● Validation with Data Annotation

ASP.NET MVC uses DataAnnotations attributes to implement validations. DataAnnotations includes built-in validation attributes for different validation rules, which can be applied to the properties of model class. ASP.NET MVC framework will automatically enforce these validation rules and display validation messages in the view.

The DataAnnotations attributes included in *System.ComponentModel.DataAnnotations* namespace. The following table lists DataAnnotations validation attributes.

| **Attribute** | **Description** |
| --- | --- |
| Required | Indicates that the property is a required field |
| StringLength | Defines a maximum length for string field |
| Range | Defines a maximum and minimum value for a numeric field |
| RegularExpression | Specifies that the field value must match with specified Regular Expression |
| CreditCard | Specifies that the specified field is a credit card number |
| CustomValidation | Specified custom validation method to validate the field |
| EmailAddress | Validates with email address format |
| FileExtension | Validates with file extension |
| MaxLength | Specifies maximum length for a string field |
| MinLength | Specifies minimum length for a string field |
| Phone | Specifies that the field is a phone number using regular expression for phone numbers |

**What you mean by Routing in ASP.Net MVC?**

Routing is a pattern matching mechanism of incoming requests to the URL patterns which are registered in route table. Class : "UrlRoutingModule" is used for the same process.

**What is Attribute Routing in ASP.Net MVC?**

ASP.NET Web API supports this type routing. This is introduced in ASP.Net MVC5. In this type of routing, attributes are being used to define the routes. This type of routing gives more control over classic URI Routing. Attribute Routing can be defined at controller level or at Action level like :

[Route("{action = TestCategoryList}")] - Controller Level

[Route("customers/{TestCategoryId:int:min(10)}")] - Action Level

**What is Dependency Injection in ASP.Net MVC?**

it's a design pattern and is used for developing loosely couple code. This is greatly used in the software projects. This will reduce the coding in case of changes on project design so this is vastly used.

**Below are the advantages of DI:**

* Reduces class coupling
* Increases code reusing
* Improves code maintainability
* Improves application testing

**Explain Test Driven Development (TDD) ?**

TDD is a methodology which says, write your tests first before you write your code. In TDD, tests drive your application design and development cycles. You do not do the check-in of your code into source control until all of your unit tests pass.

**What is Representational State Transfer (REST) mean?**

REST is an architectural style which uses HTTP protocol methods like GET, POST, PUT, and DELETE to access the data. ASP.Net MVC works in this style. In ASP.Net MVC 4 there is a support for Web API which uses to build the service using HTTP verbs.

**How to use Jquery Plugins in ASP.Net MVC validation?**

We can use dataannotations for validation in ASP.Net MVC. If we want to use validation during runtime using Jquery then we can use Jquery plugins for validation. Eg: If validation is to be done on customer name textbox then we can do as :

$('#CustomerName').rules("add", {

required: true,

minlength: 2,

messages: {

required: "Please enter name",

minlength: "Minimum length is 2"

}

});

**What are the differences between Partial View and Display Template and Edit Templates in ASP.Net MVC?**

* Display Templates: These are model centric. Meaning it depends on the properties of the view model used. It uses convention that will only display like divs or labels.
* Edit Templates: These are also model centric but will have editable controls like Textboxes.
* Partial View: These are view centric. These will differ from templates by the way they render the properties (Id's) Eg : CategoryViewModel has Product class property then it will be rendered as Model.Product.ProductName but in case of templates if we CategoryViewModel has List then @Html.DisplayFor(m => m.Products) works and it renders the template for each item of this list.

**Can I use Razor code in Javascript in ASP.Net MVC?**

Yes. We can use the razor code in JavaScript in cshtml by using <text> element.

**How we can handle the exception at controller level in ASP.Net MVC?**

Exception Handling is made simple in ASP.Net MVC and it can be done by just overriding "OnException" and set the result property of the filtercontext object (as shown below) to the view detail, which is to be returned in case of exception.

protected overrides void OnException(ExceptionContext filterContext)

{

}

**What is PartialView in ASP.Net MVC?**

PartialView is similar to UserControls in traditional web forms. For re-usability purpose partial views are used. Since it's been shared with multiple views these are kept in shared folder. Partial Views can be rendered in following ways :

* Html.Partial()
* Html.RenderPartial()

**Explain Bundle.Config in ASP.Net MVC4?**

"BundleConfig.cs" in ASP.Net MVC4 is used to register the bundles by the bundling and minification system. Many bundles are added by default including jQuery libraries like - jQuery. validate, Modernizer, and default CSS references.

Bundling and Minification:

Bundling and Minification are two performance improvement techniques that improves the request load time of the application. Most of the current major browsers limit the number of simultaneous connections per hostname to six. It means that at a time, all the additional requests will be queued by the browser.

Enabling Bundling and Minification

To enable bundling and Minification in your MVC application, open the Web.config file inside your solution. In this file search for compilation settings under system.web:

<system.web>

<compilation debug = "true" />

</system.web>

By default, you will see the debug parameter set to true which means that bundling and minification is disabled. Set this parameter to false.

Bundling

To improve the performance of the application, ASP.NET MVC provides inbuilt feature to bundle multiple files into a single file which in turn improves the page load performance because of fewer HTTP requests.

Bundling is a simple logical group of files that could be referenced by unique name and loaded with a single HTTP request.

By default, the MVC application's BundleConfig (located inside App\_Start folder) comes with the following code −

public static void RegisterBundles(BundleCollection bundles) {

// Following is the sample code to bundle all the css files in the project

// The code to bundle other javascript files will also be similar to this

bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(

"~/Content/themes/base/jquery.ui.core.css",

"~/Content/themes/base/jquery.ui.tabs.css",

"~/Content/themes/base/jquery.ui.datepicker.css",

"~/Content/themes/base/jquery.ui.progressbar.css",

"~/Content/themes/base/jquery.ui.theme.css"));

}

The above code basically bundles all the CSS files present in Content/themes/base folder into a single file.

Minification

Minification is another such performance improvement technique in which it optimizes the javascript, css code by shortening the variable names, removing unnecessary white spaces, line breaks and comments, etc. This in turn reduces the file size and helps the application to load faster.

Exceptional Handling:

In ASP.NET, error handling is done using the standard try catch approach or using application events. ASP.NET MVC comes with built-in support for exception handling using a feature known as exception filters. We are going to learn two approaches here: one with overriding the onException method and another by defining the HandleError filters.

Override OnException method:

This approach is used when we want to handle all the exceptions across the Action methods at the controller level.

To understand this approach, create an MVC application (follow the steps covered in previous chapters). Now add a new Controller class and add the following code which overrides the onException method and explicitly throws an error in our Action method:

Now let us create a common View named Error which will be shown to the user when any exception happens in the application. Inside the Views folder, create a new folder called Shared and add a new View named Error.

The advantage of this approach is that multiple actions within the same controller can share this error handling logic. However, the disadvantage is that we cannot use the same error handling logic across multiple controllers.

HandleError Attribute

The HandleError Attribute is one of the action filters that we studied in Filters and Action Filters chapter. The HandleErrorAttribute is the default implementation of IExceptionFilter. This filter handles all the exceptions raised by controller actions, filters and views.

To use this feature, first of all turn on the customErrors section in web.config. Open the web.config and place the following code inside system.web and set its value as On.