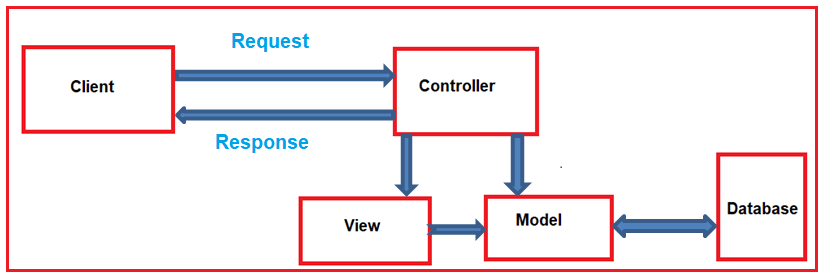
**What is MVC?**

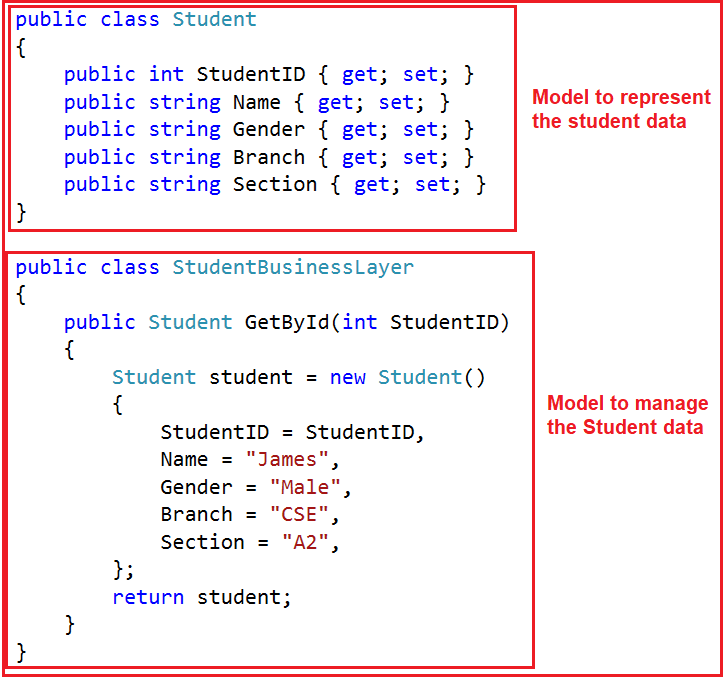
* It’s just a three-layer architecture where M stands for MODEL, V stands for VIEW, and the most important part in this architecture is CONTROLLER, like a Hero of any film
* The **ASP.NET MVC** is an open-source framework built on the top of the Microsoft .NET Framework to develop the web application that enables a clean separation of code
* ASP.net MVC offers an alternative to ASP.net web forms for building web applications. It is a part of the .Net platform for building, deploying and running web apps. You can develop web apps and website with the help of HTML, CSS, jQuery, JavaScript, etc.
* You need to remember that ASP .net MVC is NOT a replacement of ASP.Net web forms based applications.



* The controller is the component who actually receives the incoming HTTP request and then handle that request. In order to handle the incoming HTTP request, the controller does several things are as follows.
* The controller creates the model object if that is required by a view. The model is the component in MVC design, which contains a set of classes to represent the domain data or business data as well as logic to manage the data.
* The controller then selects a view to display the domain data or business data. The point that you need to remember is, while selecting a view, it is the responsibility of the controller to pass the model data.
* In the MVC, the one and only responsibility of a view to display the model data. Therefore, the responsibility of view is to generate the necessary HTML in order to render the model data. Once the HTML is generated by the view, then that HTML is then sent to the client who initially made the request

**Model**

* The Model is the component in the MVC design pattern that manages that data i.e. state of the application in memory.
* The Model contains a set of classes that represent the data as well as logic to manage the data.
* So, in our example, the model is consists of Student class to represent the student data as well as StudentBusinessLayer class to retrieve the student data from any persistent medium like a database



* In ASP.NET MVC is a C# or VB.net class to represent the data as well as to manage the data.
* It is accessible by both controller and view.
* It can be used to pass data from controller action methods to a view.
* It can also be used by a view to display data in a page (HTML output)

**View**

* The view is the component in MVC Design Pattern, which displays the model data as the user interface with which the end-user can interact.
* The View creates the user interface with data from the model. In our example, we want to display the Student information in a web page.
* So here, the student model carried the student data to the view. This is the student model, which should be supplied by the controller to the view

@model FirstMVCApplication.Models.Student

**<html>**

**<head>**

**<title>**Student Details**</title>**

**</head>**

**<body>**

**<br** **/>**

**<br** **/>**

**<table>**

**<tr>**

**<td>**Student ID: **</td>**

**<td>**@Model.StudentID**</td>**

**</tr>**

**<tr>**

**<td>**Name: **</td>**

**<td>**@Model.Name**</td>**

**</tr>**

**<tr>**

**<td>**Gender: **</td>**

**<td>**@Model.Gender **</td>**

**</tr>**

**<tr>**

**<td>**Branch: **</td>**

**<td>**@Model.Branch**</td>**

**</tr>**

**<tr>**

**<td>**Section: **</td>**

**<td>**@Model.Section **</td>**

**</tr>**

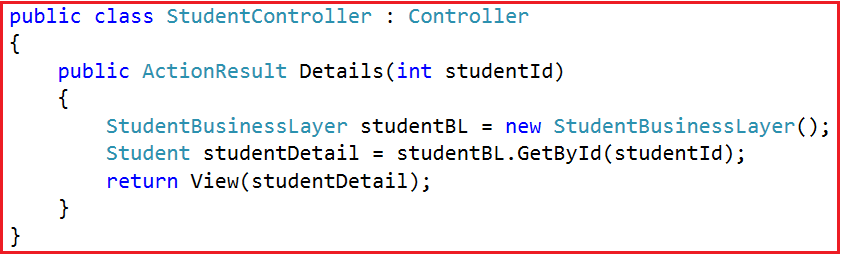
**</table>**

**</body>**

**</html>**

* In ASP.NET MVC is a cshtml page.
* It contains all page specific HTML generation and formatting code.
* A request to a view can only be made from a controller’s action method.
* The one and only responsibility of a view to render the domain data

**Controller**

* This is the one will interact with both models and views to control the flow of application execution.
* The controller is the component in MVC Design Pattern, which will handle the incoming HTTP Request.
* Based on the user actions, the respective controller will work with the model and view and then sends the response back to the user who initially made the request
* 
* The request is going to be mapped to the Details action method of the Student Controller. Following is the code of our Controller class with the Details action method
* It is a C# or VB.NET class that is inherited from the System.Web.Mvc.Controller.
* Is the component, which will interact with both Models and views.
* It contains action methods that are responsible for handling the incoming URL.
* Can access and use the model class to pass the data to the views

**Understanding Model, View and Controller**

Explain in practical

Create MVC project

Explain about Route config & how the startup page is coming and how to configure and where to change the startup page

Explain the theoretical MVC pattern in project

**Key benefits of ASP.NET MVC**

* It is lightweight because it does not use view state or server-based forms or server controls.
* Each developer based on his expertise or experience can work on different parts of the application. For example, one developer may work on the view while the second developer can work on the controller logic and the third developer may work on the business logic.
* Clean HTML and easy integration with JavaScript and jQuery.
* It provides better support for test-driven development (TDD). This is because we can focus on one aspect at a time i.e. we can focus on the view without worrying about business logic.
* ASP.NET MVC Framework divides the application into three main aspects such as Model, View, and Controller, which make it easier to manage the application complexity.
* Another important advantage of the ASP.NET MVC framework is its components are designed to be extensible and pluggable and therefore they are easily replaced or customized.
* The MVC framework is built on top of the ASP.NET Framework and hence we can use most of the ASP.NET features such as authentication and authorization scenarios, membership and roles, caching, session and many more.
* ASP.NET MVC framework supports a powerful URL routing mechanism (i.e. attribute routing) which helps to build a more user-friendly and SEO friendly URLs for our application
* **Development**: The MVC components such as (Model, View and Controller) can be developed in isolation. Because they are not directly depended on each other
* **Testing** The MVC components are not directly depend on each other this make it easy to test components separately.
* **Maintenance**: All components logic is separated it make it easy for Maintenance in future.
* **Control over HTML**: You want full control over the HTML that is rendered in the browser and you can afford the development time and overhead to do all of your own markup.

**Difference between ASP.NET webforms and ASP.NET MVC**

|  |  |
| --- | --- |
| ASP.net web form | ASP.net MVC |
| Page controller pattern that means an implicit controller (code behind) would process the request | Front controller pattern that means an explicitcontroller would be there to process the request |
| web form has user controls for code reusability | Partial views has code reusability |
| view and controllers are not separated | view and controller are handled separately |
| stateful and view state is used to maintain state | Stateless, so view state is not used |
| Has server control | Has html helper |
| views are tightly coupled with business logic | views and logics are separately managed |
| Master pages for constant look and feel | MVC layouts for constant look and feel |
| File based urls and it needs physical files | Route based urls and does not need a physical file. It depends on controller |
| Recommended for small scale applications | Recommended for large scale applications |

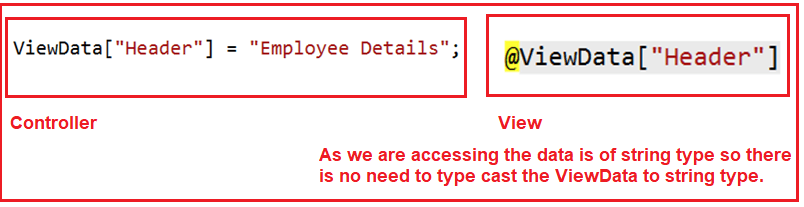
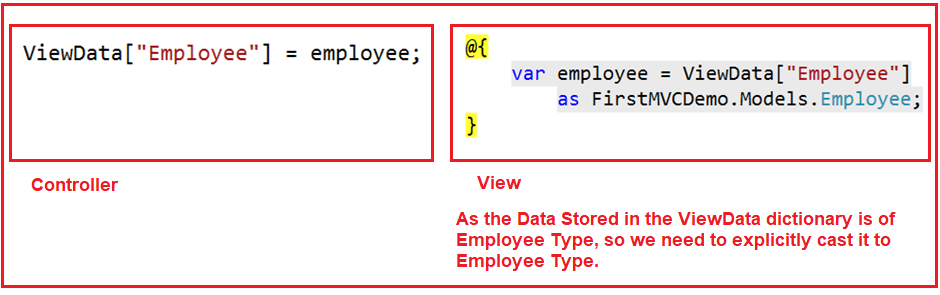
**HTML HELPERS in ASP.NET MVC**

* ASP.NET web forms application, as a developer, we generally use the toolbox for adding controls on any particular web page. However, coming to ASP.NET MVC application there is no such toolbox available to drag and drop HTML controls on to the view
* So, to overcome the above problem, the ASP.NET MVC Framework provides Html Helper classes which contain different extension methods.
* We can use those extension methods to create HTML controls programmatically within a view.
* All the HtmlHelper methods that are present within the HtmlHelper class generate HTML and return the result as an HTML string
* An HTML Helper in MVC is an extension method of the HTML Helper class which is used to generate HTML content in a view.  For example, if you want to generate a textbox with id=”firstname” and name=”firstname” then you can type all the required HTML in a view as shown below
* <input type=”text” name=”firtsname” id=”firstname” />
* @Html.TextBox(“firstname”)
* @Html.TextBox(“firstname”, “Pranaya”)
* <input id=”firstname” name=”firstname” type=”text” value=”Pranaya” />
* **@Html.TextBox(“firstname”, “Pranaya”, new { style = “background-color:Red; color:White; font-weight:bold”, title=”Please enter your first name” })**
* @Html.TextBox(“firstname”, “Pranaya”, new {@class= “redtextbox”,@readonly=”true” })
* Explain in project

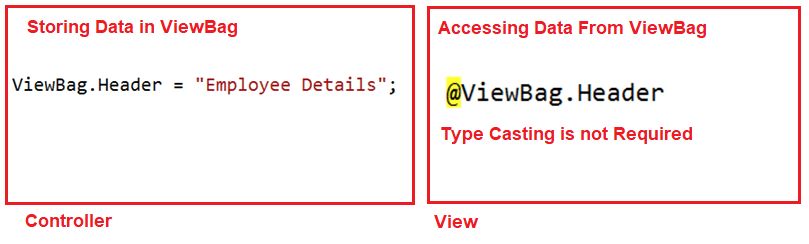
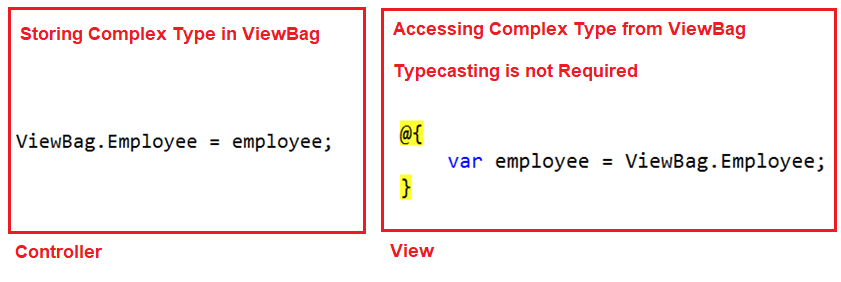
**What is View Data, View Bag and Temp Data**

* We can pass the model data from a controller to a view in many ways such as by using ViewBag, ViewData, TempData
* You can also use the Session and Application State variable as we use in our traditional Web Forms to manage the data during a user session or throughout the application
* Now the most important question that comes to your mind is when to use ViewData, ViewBag, TempData, Session, and Application as each one having its own advantages and disadvantages

**View Data**

* The ViewData in ASP.NET MVC is a mechanism to pass the data from a controller action method to a view.
* signature of the ViewData in MVC
* The return type of ViewData is ViewDataDictionary
* Therefore, we can say that the **ViewData in ASP.NET MVC** is a **weakly typed dictionary object**.
* As it is a dictionary object, so it is going to store the data in the form of **key-value pairs** where each **key must be a string** and the value that we are passing to the dictionary is going to be stored in the form of an **object type**.
* The most important point that you need to remember is, as it stores the data in the form of an object so while retrieving the data from ViewData **type casting is required**.
* If you are accessing string data from the ViewData dictionary, then it is not required to typecast the ViewData to string type.
* But it is mandatory to typecast explicitly to the actual type if you are accessing data other than the string type from the ViewData
* The ViewData in ASP.NET MVC can only transfer the data from a controller action method to a view. That means it is valid only during the **current request**.
* 
* 

**View Bag**

* The ViewBag in ASP.NET MVC is one of the mechanisms to pass the data from a controller to a view
* ViewBag in ASP.NET MVC Application
* ViewBag is a **dynamic property** (a new feature introduced in C# 4.0) of the Controller base class. The ViewBag is also like ViewData, which also transfers the data from a controller action method to a view
* ViewBag is operating on the new dynamic data type. The advantage is that we **do not require typecasting** while accessing the data from a ViewBag irrespective of the data that we are accessing
* 
* 

**Temp Data**

* The limitation of both ViewData and ViewBag is they are limited to one HTTP request only. Therefore, if redirection occurs then their values become null means they will lose the data they hold
* For example, we may need to pass the data from **one controller to another controller** or **one action method to another action method within the same controller**. Then in such situations like this, we need to use TempData.
* The TempData in ASP.NET MVC is one of the mechanisms to pass the small amount of temporary data from a controller to a view as well as from a controller action method to another action method either within the same controller or to a different controller.
* TempData in ASP.NET MVC
* We can say that the **TempData in ASP.NET MVC** is a**dictionary object**. As it is a dictionary object, so it is going to store the data in the form of **key-value pairs** where each **key must be a string** and the value that we are passing to the dictionary is going to be stored in the form of an **object type**.
* The most important point that you need to remember is, as it stores the data in the form of an object so while retrieving the data from **TempData type casting is required**
* If you are accessing string value from the TempData, then it is not required to typecast. However, it is **mandatory to typecast** explicitly to the actual type if you are accessing data other than the string type from the TempData.
* Syntax to store data in TempData
* **TempData[“YourData”] = “SomeData”;**
* Syntax to retrieve data from TempData
* **string strData = TempData[“YourData”].ToString();**
* **Keep Feature**

**Difference View Data, View Bag and Temp Data**

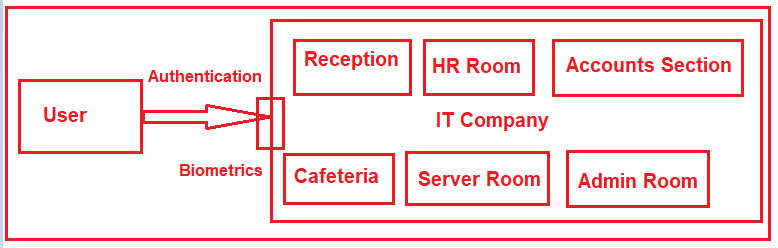
**Difference and Similarities between ViewData and ViewBag in ASP.NET MVC**

* In ASP.NET MVC, we can use both ViewData and ViewBag to pass the data from a Controller to a View.
* The ViewData is a dictionary object whereas the ViewBag is a dynamic property. Both ViewData and ViewBag are used to create loosely typed views in MVC.
* In ViewData, we use the string as the key to store and retrieve the data whereas in ViewBag we use the dynamic properties to store and retrieve data.
* The ViewData requires typecasting for complex data type and also checks for null values to avoid any exception whereas ViewBag does not require any typecasting for the complex data type.

**Authentication and Authorization in ASP.NET MVC**

* When you are developing any web application, then the most important thing that you need to take care of its security.
* That means we need to make sure that only authenticated and authorized user can access our webpage

**Authentication**

* Authentication is nothing but a process that ensures and confirms a user’s identity. In other words, we can say that it is a process to validate someone against some data source
* 
* The above image shows the different section of an IT Company like Reception, HR Section, Accounts Section, Server Room, etc. At the gate, we have biometrics to verify the employee.
* Suppose one user or one employee comes. This biometrics checks the employee credentials against some data source and if it found the employee is a valid employee then it only allows entering into the campus.
* This is nothing but Authentication.

**Authorization**

* Authorization is a security mechanism, which is used to determine whether the user has access to a particular resource, or not.
* The most point that you need to remember is, authentication happens first, then only authorization
* Once the user authenticated then he enters into the Campus. Then Authorization comes into the picture. Within the campus in which section he may allow entering is determined by the Authorization process.
* This is done by the Role of the user. If the user is having list privileges then he may not allow to each and every section.
* On the other hand, if the user is having the highest privileges then he may allow entering each and every section.

**Types of Authentication:**

* **Forms Authentication:** In this type of authentication, the user needs to provide his credentials through a form.
* **Windows Authentication:** Windows Authentication is used in conjunction with IIS authentication. The Authentication is performed by IIS in one of three ways such as basic, digest, or Integrated Windows Authentication. When IIS authentication is completed, then ASP.NET uses the authenticated identity to authorize access
* **Passport Authentication:** It is a centralized authentication service (paid service) provided by Microsoft which offers a single logon and core profile services for member sites.
* **None:** No Authentication provided. This is default Authentication mode

Explain with created projects

**Exception handling in ASP.NET MVC**

* HandlerError is used to handle exceptions while executing action methods.
* Explain with coding

**Validations in ASP.NET MVC**

<https://dotnettutorials.net/lesson/data-annotation-attributes-mvc/>

* Data Annotation Attributes in ASP.NET MVC Application, which is used for validating the data of a model.
* It is a challenging job for a web developer to validate the user input for any Web application. As a web developer, we not only validate the business logic at the client side that is in the browser, but also we need to validate the business logic running on the Server. That means as a developer we need to validate the business logic at both the client side as well as server side
* The client-side validation of the business logic gives the users immediate feedback on the information they entered into a web page and which is an expected feature in today’s web applications
* We can say that validations are nothing but some rules set by the developer on the input fields of a web page to satisfy the business rules for that particular input field in order have to maintain the proper data in a system
* There are two types of validations:
* **Server-side Validations** **Client Side Validations**
* While doing validations, as a developer we need to take care of not only the proper validation but also ensure that the validation meets the business rule as per the requirement
* When we talk about the validation in ASP.NET MVC framework, we primarily focus on validating the model value. That means has the user provided a required value. Is the value in the required range? Is the value in a proper format etc.?
* **We can do the following three types of validations:**
* HTML validation / JavaScript validation (i.e. Client-Side Validation)
* ASP.NET MVC Model validation (i.e. Server-side Validation)
* Database validation (i.e. Server-side Validation)
* The System.ComponentModel.DataAnnotations assembly has many built-in validation attributes, for example:
* **Required, Range, RegularExpression, Compare, StringLength, Data type**
* Along with the above build-in validation attributes, there are also many data types the user can select to validate the input. Using this data type attribute, the user can validate for the exact data type as in the following:
* **Credit Card number, Currency, Custom, Date, DateTime, Duration, Email Address, HTML, Image URL, Multiline text, Password, Phone number, Postal Code, Upload**
* **Explain with projects: ValidationExample**

**Passing data from Controller to View**

* There are many ways to pass Model class data to the View, using a Controller class
* By using ViewBag, ViewData, TempData we can pass the data from Controller to View
* The other way of passing the data from Controller to View can be by passing an object of the model class to the View
* Explain project: **PassData**

**What is Razor Engine?**

* Razor view engine in asp.net mvc is syntax that allows you to write server side code on view.
* Razor is not a new programming language if you know C#, Vb.Net and bit HTML you can easily write Razor code. Razor supports C# and Visual Basic programming languages
* Razor allows you to write mix of HTML and server side code using C# or Visual Basic
* Generally in razor view engine code blocks are enclosed in "**@{ ... }"**. Here “@” is character that tells beginning of Razor syntax.
* @{

//Razor block

}

* In razor view engine we will **declare variable** like as shown below

@{

var weekDay = DateTime.Now.DayOfWeek;

}

* following represents syntax of **for loop** in asp.net mvc razor view engine

@for (int i = 0; i < 5; i++)

{

//Code Block

}

* Razor syntax are easy to learn and much clean than Web Form syntax. Razor uses @ symbol to make the code like as:

@Html.ActionLink("SignUp", "SignUp")

* <h2>@DateTime.Now.ToShortDateString()</h2>

**Some of Razor Syntax Rules for C# are given below**.

* It must be always enclosed in @{ ... }
* Semicolon “;” must be used to ending statements
* Files have .cshtml extension.
* Variables are declared with var keyword
* Inline expressions (variables and functions) start with @
* C# code is case sensitive

**What is Partial View?**

* When we need a common part of the user interface at multiple pages in a web application then we develop a partial view, hence partial view is a regular view which can be used multiple times in an application and has the file extension .cshtml.
* Sometimes we also use a partial view to divide a web page into small parts such as header, footer, and menu on Layout.
* Other examples are comments in blogging site, shipping and billing address in the invoice in e-commerce site etc
* If you are coming from asp.net web-forms background, then you can realize that partial views in MVC are similar to user controls in asp.net web forms
* Partial views help us to reduce code duplication. Hence partial views are reusable views like as Header and Footer views.
* We can call or display partial view within a view mainly in five ways. Those are

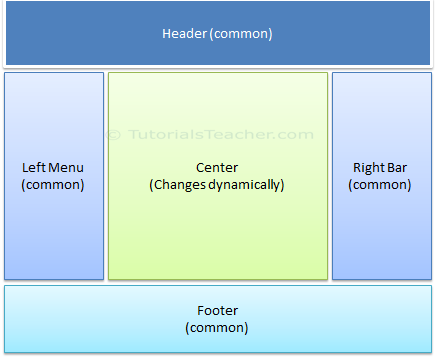
1. Html.RenderPartial
2. Html.Partial
3. Html.RenderAction
4. Html.Action
5. jQuery load function

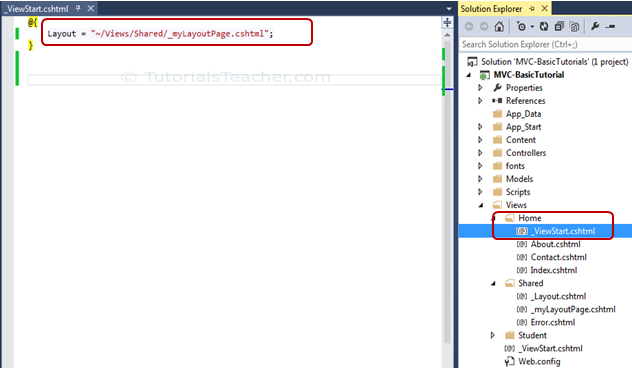
* Explain in project: **Partialview**

**Scaffolding in ASP.NET MVC**

**Layout and Viewstart page in in ASP.NET MVC**

* An application may contain common parts in the UI which remains the same throughout the application such as the logo, header, left navigation bar, right bar or footer section
* ASP.NET MVC introduced a Layout view, which contains these common UI parts, so that we do not have to write the same code in every page. The layout view is same as the master page of the ASP.NET web form application
* an application UI may contain Header, Left menu bar, right bar and footer section that remains same in every page and only the center section changes dynamically as shown below
* Layout view has same extension as other views, .cshtml or .vbhtml. Layout views are shared with multiple views, so it must be stored in the Shared folder.
* For example, when we created our [first MVC application](https://www.tutorialsteacher.com/mvc/create-first-asp.net-mvc-application) in the previous section, it also created \_Layout.cshtml in the Shared folder as shown below



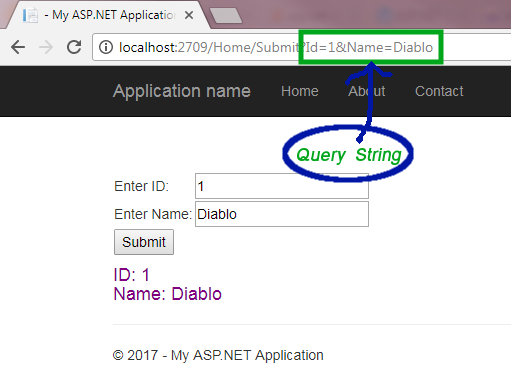
* You can set the layout view in multiple ways, by using \_ViewStart.cshtml or setting up path of the layout page using Layout property in the individual view or specifying layout view name in the action method
* For example, the following \_ViewStart.cshtml in the Views folder sets the Layout property to "~/Views/Shared/\_Layout.cshtml". So now, \_layout.cshtml would be layout view of all the views included in Views and its subfolders.
* So by default, all the views derived default layout page from \_ViewStart.cshtml of Views folder
* \_ViewStart.cshtml can also be included in sub folder of View folder to set the default layout page for all the views included in that particular subfolder only.
* For example, the following \_ViewStart.cshtml in Home folder sets Layout property to \_myLayoutPage.cshtml. Therefore, this \_ViewStart.cshtml will influence all the views included in the Home folder only. So now, Index, About and Contact views will be rendered in \_myLayoutPage.cshtml instead of default \_Layout.cshml
* 
* Explain in project: **partialview**

**Http Postings – GET, POST, PUT, DELETE**

* HTTP is a HyperText Transfer Protocol that is designed to send and receive data between client and server using web pages. HTTP has two methods that are used for posting data from web pages to the server. These two methods are HttpGet and HttpPost.
* Here, a web browser is the client and an application on a computer that hosts a web site is the server
* We use GET method to return items
* We use POST method to create a new item.
* We use PUT method to update an item.
* We use the DELETE method to remove an item

**GET**

* HttpGet method sends data using a query string.
* The data is attached to URL and it is visible to all the users.
* However, it is not secure but it is fast and quick.
* It is mostly used when you are not posting any sensitive data to the server like username, password, credit card info etc.
* Data is limited to max length of query string.
* It is very useful when data is not sensitive
* It can carry only text data.
* When you will run the application and pass value using HttpGet method, you will see that data is attached to URL using Query String.
* http://localhost:2709/Home/Submit?Id=1&Name=Diablo
* In this previous link, the data is passed using query string because submit button used HttpGet Method to post data. You can see that Id=1&Name=Diablo in URL.



**POST**

* **HTTPPost** method hides information from URL and does not bind data to URL. It is more secure than HttpGet method but it is slower than HttpGet.
* It is only useful when you are passing sensitive information to the server
* Data is sent via HttpPost method, is not visible to user.
* It is more secured but slower than HttpGet.
* It is used when sending critical data.
* It can carry both text and binary data
* Explain in Project

**Difference between HttpGet and HttpPost Method**

* HTTPGet method is default whereas you need to specify HTTPPost attribute if you are posting data using HTTPPost method.
* HTTPGet method creates a query string of the name-value pair Whereas HTTPPost method passes the name and value pairs in the body of the HTTP request
* HTTPGet request has limited length and mostly it is limited to 255 characters long whereas HTTPPost request has no maximum limit.
* HTTPGet is comparatively faster than HTTPPost. HTTPPost takes extra time in encapsulating the data.  
  HTTPGet can carry only string data whereas HTTPPost can carry both string and binary data.
* HTTPGet method creates readable url so it can be cached and bookmarked whereas such facility is not available in HTTPPost method

**AJAX calling in ASP.NET MVC, JQuery**

Explain in MVCUsingAJAX

**CRUD operations in ASP.NET MVC**

**CRUD operations in ASP.NET MVC using Entity Framework**