MVC

Class notes

By

Mr . **Narasimha** sir

12/10/15

MVC

1) What is MVC?

* MVC stands for Model View Controller
* MVC is a design pattern used to develope the application

Note:
Microsoft was used MVC design pattern in ASP.NET MVC.

2) What is Design pattern?

* Design pattern is a solution for the problems occures in software development
* There are several design pattern that are available in software industry
* These design patterns are not specific to any company/technology.
* These can be used by any application to resolve the problems.

3) What are the Examples of Design pattern?

1. MVC — ASP.NET MVC
2. DAO — ADO.NET
3. Iterator — Foreach
4. proxy — WCF/Webservice.
5. Singleton — Wcf.

13/10/15

4) What is MVC design pattern? How to implement in ASP.NET MVC?

5) Explain the modules in MVC design pattern?

Ans(4&5):
* MVC is a design pattern that canbe used in application development.
* In MVC, application development is splited into three Modules:

1. Model
2. View
3. Control.

1. Model :-

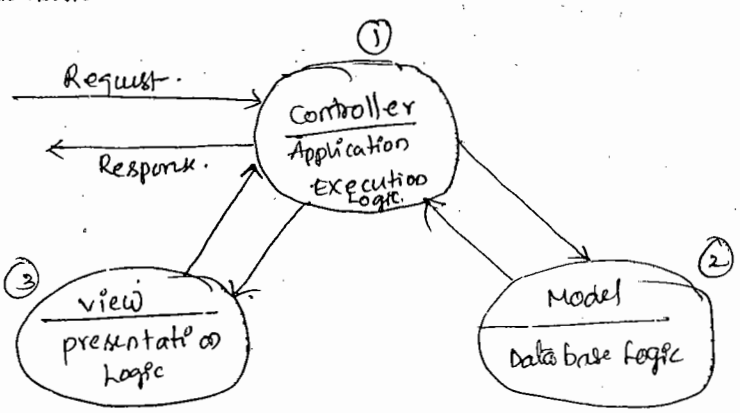
* Model is responsible for database related logic. Model will communicate with database to perform DB database operations.
* In ASP.NET MVC, Model is implemented by using ADO.NET (or) Entity Framework.
* File Extension : *.cs.

2. View :-

* view is responsible for presentation (UI) Logic.
* view will get the results from controller so that it will take care how to display to end user.
* In ASP.NET MVC, views are developed by using C# code & HTML
* File Extension : *.cshtml.

3. Controller :-

* Controller is responsible for application execution logic.
* Every request in MVC applications will come to Controller.
* Controller handles the request processing Logic.
* Controller will communicate with Model and view.
* In ASP.NET MVC, controller is implemented as class.
* File extension : *.cs.



6) What are different Technologies that using MVC design pattern?

1. Small Talk
2. Ruby on Rails
3. Struts
4. Spring
5. Python
6. ASP.NET MVC
7. Iphone OS
8. Angular JS.

14/10/15

ASP.NET MVC.

⇒ What is ASP.Net MVC?

* Asp.net is a framework used for developing web applications using MVC pattern.
* In Asp.Net MVC, project ~~no~~ development is divided into three modules

Model — Db logic
View — UI Logic
Controller — Execution logic.

* ASP.NET MVC was introduced by Microsoft in the year of 2009.
* After 2009, we have two applications to develop web applications in .Net:
  → ASP.NET
  → ASP.NET MVC.

Note: ASP.NET MVC is not replacement of ~~Asp~~.NET. It is an alternate to ASP.NET.

* ASP.NET MVC Framework was designed by Mr. Scott Guthriew in 2007.

Version history of ASP.NET MVC:-

- ASP.NET MVC — 2007 (Beta)
- ASP.NET MVC 1.0 — 2009
- ASP.NET MVC 2.0 — 2010.
- ASP.NET MVC 3.0 — 2011.
- ASP.NET MVC 4.0 — 2012
- ASP.NET MVC 5.0 — 2013.
- ASP.NET MVC 5.2 — 2014.

Note :-

VS 2010 SP1, VS 2010.

ASP.NET MVC 4.0

ASP.NET MVC Features :-

Q) :- What are advantages of ASP.NET MVC ? (or) What are important features of ASP.NET MVC ? (or) Why do we use ASP.NET MVC instead of ASP.NET ?

Ans: 1. Separation of Concern (or Code (SoC) :-

* SoC is a processing of dividing project development into separate Modules.
* In MVC, presentation logic (View) is separate from database Logic (Model)
* This is the important feature of MVC, remaining all features are depends on this
  Due to SoC, we can easily organize the development process of complex applications.
* It will be comfortable in development, testing and Maintanance phases

2. Loosely Coupled (Less dependency) :-

* Loosely coupled is talk about dependency of module in the project
* If an application is tightly coupled, such kind of applications are difficult to modify.
* MVC follows loosely coupling so that we can easily edit / extend at any point of time without modifying entire project.

Note:

project controlling Tools:

VSTS — visual studio | 1. check in - checked out process
VSTFS
VSS

3. parallel Development :-

* parallel Development is a procus developing the application by multiple programmers.
* In MVC, views and modules are developed parallel by more than one programmer
* If one programmer working on views, another programmer can works on models without depending / waiting for another.
* It will causes development procus will be more easy and faster.

15/10/15

4. Easy to perform unit Testing :-

* Unit Testing is a process of testing required portion of application which we modified / added.
* In Unit Testing, no need to test entire application.
* In MVC applications, due to separation modules we can easily implement unit Testing.
* we can test only required views / Model without testing entire application.
* MVC based applications are more comfortable for unit Testing.

5. TDD Support :-

* TDD stands for Test Driven Development.
* It is latest development procus.
* In TDD, testers will provide guidelines to developers.
* At the time of development, if any options provided testers, developers will be implemented in application.
* MVC applications are more comfortable to modify / adding new options without effecting entire project.

6. Clean URL's :-

* ASP. NET MVC applications uses different URL system which doesnot depends on the file system
* It is controller based URL's. In ASP. NET, we are using file system based URL's,
* MVC URL's are simple / short URL's hence it is called "clean URL's".

* These URL's are easy to perform SEO.

What is SEO?

* SEO stands for Search Engine Optimization
* It is a process of making our website URL is easily identified by search engine.

7. Improved performance :- (Faster in execution)

* In ASP.NET, execution of the page will take much time compare with ASP.NET MVC.
* Due to server controls, events, view state .. execution of the page will add extra burden on the ASP.NET Engine.
* In ASP.NET MVC, we are using only HTML controls (no server controls)
* Due to this page execution will be more faster.

* 8. More Controlling on HTML (Client Side Programming):-

- MVC applications are using completely HTML controls for Views Developement
- We can easily integrate any client side frameworks with Views

  e.g:- jQuery, AngularJS (Javascript Framework), Bootstrap. [CSS Framework]

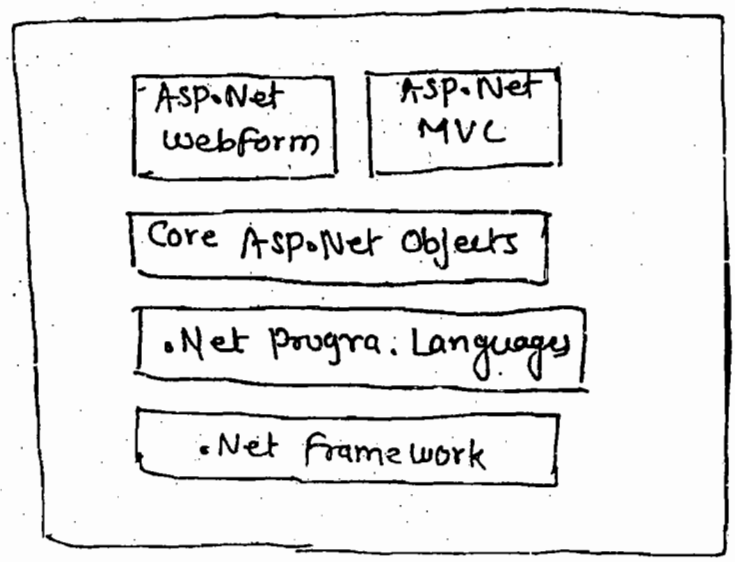
* 9. New Concepts Added :-

- Filters
- Routing, Attribute Routing
- Web API [service based]
- Integration of Entity Framework
- Scaffolding Template

- ASP.Net MVC New Concepts, in, order to make your application developement make more easy.

* 10. Easy to Learn & Implement :-

- Asp.Net MVC was developed based on Existing technologies of .Net
- So that any .Net student/programmer, can easily learn & implement ASP.Net MVC Applications.



⭐ Developing Asp.Net MVC Application :→

① Asp.Net framework provides multiple Templates includes to developing MVC applications :-

- Ⓐ Main Template
- Ⓑ Sub Template

② Asp.Net MVC 4 Web Applications :-

Sub Templates :-
- Empty
- Basic
- Internet Application
- Intranet Application
- Mobile Application
- Web API.

⭐ Note :- All the above templates will generate three primary as follows :-
1) [Controllers]
2) [Models]
3) [View]

(A) Template Information :-

(1) Empty :-

(a) This template is similar ~~to~~ the ASP.Net Empty Web Application.

(b) It generate Empty folders so that we can add request files.

(2) Basic :-

(a) Basic Template is similar to Empty Template

(b) It generates additional folders :-

[content] → *.css

[scripts] → *.js.

(3) Internet Applications :-

(a) It generates a default ready made project with default files.

(b) It is similar to ASP.Net Web Applications.

(c) It Uses "Forms Authentication".

(4) Intranet Applications :-

(a) This template is same Internet Applications.

(b) It uses "Windows Authentication".

(5) Mobile Applications :-

(a) Mobile Applications is also same as "Internet Application"

(b) But it targets Mobiles Device instead of Browser.

⑥ Web API :-

ⓐ It is new Service based Application

ⓑ It is developed based Asp.Net MVC framework.

ⓒ Web API Service, is called as "HTTP Service".

* Note :-

① Mobile Applications & Web API templates are introduced in Asp.Net MVC 4.0.

② Use "Basic" template to start your application Developement

File →> Project →> Asp.Net MVC 4 Web Appⁿ
→> Basic
→> OK.

Assignment

① Find open program.cs file. Find files

② Try to find means of <form> tag. ??

17/10/15

Folder structure of ASP.NET MVC Application :-

* ASP.NET MVC project templates generates following important folders:

1. App_Start :-

This folder contains .cs files to provide application level configration details.

* For example, routeconfig.cs file provides default controllers and action methods.

* RouteConfig.cs file is editable so that we can set default controller and action as per our requirement.

2. Content :-

* This folder contains styles related files .css

* we can also add user defined .css files in this folder.

3. Controller :-

* It contains all controller class files.

* Each controller is one class.

Eg: HomeController.cs
DemoController.cs.

4. Models :-

* This folder contains database programming related class files.

5. Scripts :-

* This folder contains client side script files .JS.

* user defined and pre-defined script files exists in this folder.

X - Every MVC project provides jquery library files, if required we can involve jquery in view development.

6. Views :-

* This folder contains the view pages required for action methods
* In ASP.NET MVC, Views are developed with C# and HTML.
* View pages are having extension .cshtml.

Eg: Index.cshtml
Login.cshtml.

Ex: Create a MVC Application to implement following webpages

1. Welcome page (Index page)
2. Courses page
3. About Us page
4. Contact Us page

Step①:- Create New Asp.NET MVC webapplication

→ open visual studio

→ Files → New → project → visual C# → web → Asp.NET MVC 4

web Application → Name: MVC Application1.

→ click "OK"

→ select "Basic" project template → click "OK"

→ This setup will create MVC project with required Folders.

Step②: Add Home Controller to project

→ Goto solution explorer.

→ Rt.click on "controllers" Folder. → Add → Controller

controller Name: Homecontroller.

→ click "Add" button

→ This ~~set~~ setup step will generate HomeController.cs file.

Step③ :-

prepare required action method in ~~action~~ HomeController class as follows:

File Name : HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MVCApplication40.Controllers
{
 public class HomeController : Controller
  {
    public ActionResult ~~Courses~~ Index()
    {
      return View();
    }
    public ActionResult Courses()
    {
      return View();
    }
    public ActionResult Aboutus()
    {
      return View();
    }
    public ActionResult ContactUs()
    {
      return View();
    }
  }
}
```

Step ④ : Add view pages for above action methods of HomeController

→ Rt. click on ActionMethod
→ Select "Add view"
→ click "Add" button.

This step will generate corresponding view page with .cshtml file extension.

Step ⑤ : prepare the required content in view pages.

FileName : Index.cshtml

```
<h1> Satya Technologies </h1>
<hr/>
<a href ="/Home/ courses "> Courses </a>
<br/>
<a href ="/Home/ About Us "> About Us </a>
<br/>
<a href ="/Home/ Contact Us"> Contact us </a>
```

File Name : Courses.cshtml

```
<h1> courses page </h1>
<hr/>
<UL>
    <li> Asp </li>
    <li> Asp.Net </li>
    <Li> C# </LI>
    <Li> MVC </LI>
    <Li> Angular Js </Li>
</UL>
<hr/>
<a href ="/"> Back to Index </a>.
```

File Name: About Us.Cshtml

```
<h2> About Us page </h2>
<hr/>
<p> This page contains information about Me... </p>
<hr/>
<a href ="/"> Back to Index </a>
```

File Name: ContactUs.Cshtml

```
<h2> Contact Us page </h2>
<hr/>
<p> This page contains Contact No: 9966311434 </p>
<hr/>
<a href = "/"> Back to Index </a>
```

19/10/15

Controllers in ASP.NET MVC:-

* In MVC, C - stands for Controller.
* Controller is responsible for application execution logic.
* Controller is important Module in MVC which will take care of execution of every request.
* Execution process always starts from controller only.
* Controller is acts as a mediator between Model and view so that model communicate with view through controller.

Responsibilities of controller:

* In order to process the request controller will perform following activities
  1. Receive the request.
  2. process the request
     - → communicate with Model
     - → communicate with View.

3. Send the ~~request~~ Response.

Note :-

At the time of processing request it will execute corresponding method which is called "Action Method".

Developing Controllers :-

* In ASP.NET MVC Application development, every controller is created as a class
* A project can have more than one controller class which is depends on the requirements.
* Each controller class defines corresponding action methods to process the request.

Rules for creating Controller :-

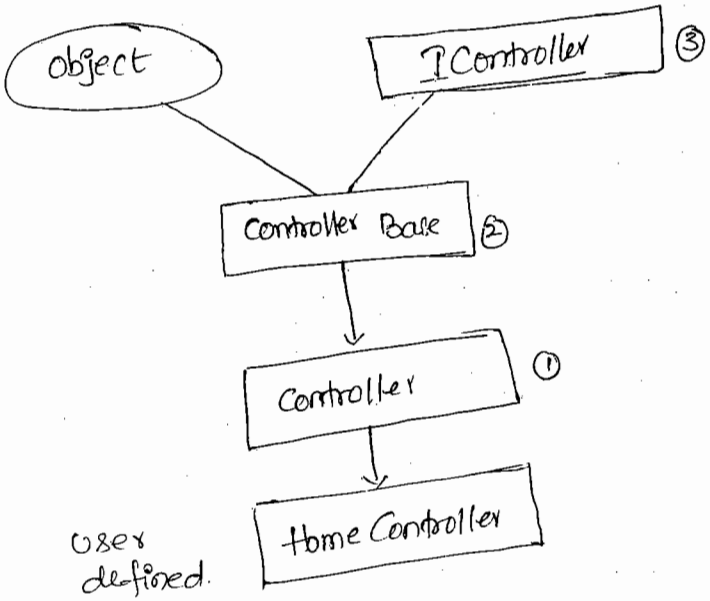
1. Controller class should be declare with "public" access specifier.
2. Controller class name should be end-with (postfix) "Controller" word.

   Eg :- Home Controller
   Student Controller
   EmpController ..etc.

3. All user defined controller classes should be inherit from "Controller" base class
   * A controller base class provides built-in objects, Methods to process request
   Eg : View(), Request, Response, Session, etc----
   * parent class "Controller" is predefined in System.web.mvc namespace.

20/10/15

Controller class hierarchy:-



Library Information:-

Library (or Assembly — System.web.Mvc.dll.

Name space — System.web.Mvc

Action Methods :—

* Every request that will come to MVC application will be mapping to controller and corresponding action methods.
* Action Methods contains the logic to process the request.
* ActionMethods are browsable from browser i,e user can make request to action Methods.
* At the time of request submission using "Submit" button, browser uses "Action" and "Methods" attributes of <form> tag.
* By default all action Methods in controller class are browsable.

Developing Action Methods :-

* By default every controller class comes with default action method called "Index()".

```
public ActionResult Index()
{
    return View();
}.
```

Details about Action Method :-

* Action Method return statement uses "View()" Method.
* View() is used to process the views
* View() Method is a pre-defined Method in controller class (parent class)
* At the time of execution, View() will execute and result will be return from action Method.
* View() Method return type is "View Result".
* In Action method definition return type is specified as "Action Result".
* Action Result is base class for "View Result".

⇒ Example of Result class in MVC

Base class : Action Result.

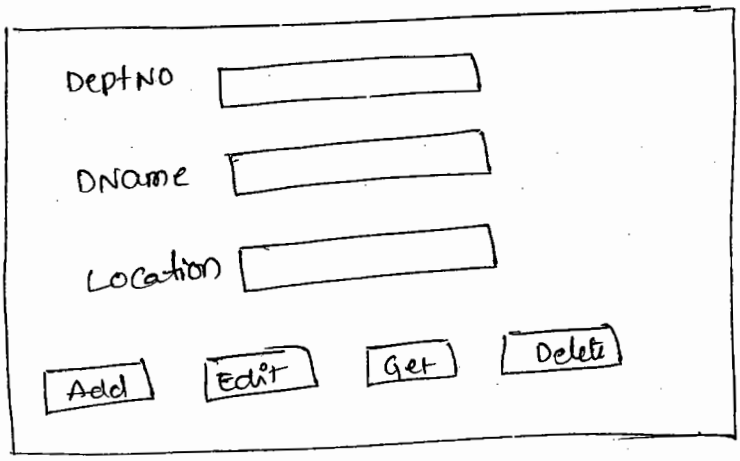
child class : ViewResult
ContentResult
RedirectToRouteResult
JsonResult
FileResult
etc...

Rules to Create Action Methods :-

CRUD : Create Read UPdate Delete

* Action Methods should be declare with "public" access specifier.
* Action Methods should be instance (non-static) Method.

→ Create a windows form perform CRUD operations on windows Depttable




```
Dept obj = new Dept();
obj.DepNO = int.parse(textBox1.Text);
obj.Dname = textBox2.Text;
obj.Loc = textBox3.Text;

Dboperations db = new Dboperations();
db.AddDept(obj);
MB.show("Dept details are added in db");
```

26/10/15

Types of Action Methods:-

* Methods that we prepare in Controller class are called "Action Methods".
* All Methods that we prepared in controller are divided three types:

1. Action Method with ViewResult.
2. Action Method with non-view result.
3. Non- Action Methods.

1. Action Method with viewResult:-

* Action Method that will return view page as output those methods comes under this type.
* Most of the action Method are return viewResult.
* These Methods return statement: return view();
* These Methods required view page with corresponding Method name (Eg: Index.cshtml).

2. Action Method with Non-view Result:-

* Action Methods that will returns data instead html formatted output.
* These methods will return single value/object/collection of objects.
* Especially in Ajax programming we are involving these Methods.
* These methods having return statement:

```
return Json();
return content();
- - -
```

Note: These methods does not required view pages.

3. Non - Action Methods :-

* By default all methods that are created in controller class are browsable.
* If you want to create methods that can access with in the controller, then we can create them as non-action methods.
* Non - Action Methods are not browsable. If any request will come to this method, it will throw 404 exception
* Non - Action Methods should be declared with an attribute called - "[NonAction]".

Eg:
```
[NonAction]
public int GetTotal()
{
  // code
}.
```

Example: About Action Methods.

1. Create New ASP.NET MVC application Add Home controller.
2. Define the following action methods in "Home Controller.cs" file:

```
Namespace MVCApplication47.controllers
{
public class HomeController : Controller
 {
  public ActionResult Index()
  {
    return View();
  }
  public ActionResult Details()
  {
    string Sname = "Scott";
   int m1 = 86, m2 = 96, m3 = 75;
   int total = GetTotal(m1, m2, m3);
   Response.Write("Student Name : " + Sname);
        Response.Write("<br>");
```

```
Response.Write(" Total Marks:" + total);
    return View();
}
[Non Action]
public int Gettotal(int m1, int m2, int m3)
{
    int n = m1 + m2 + m3;
    return n;
}
public ActionResult GetServerTime()
{
    string str = " Server Time: " + DateTime.Now.ToString("T");
        return Content(str);
    }
}
}
```

Step③: Add views for required Action Methods.

1. Index.cshtml:

```
<h1> Action Method Types Demo </h1>
<hr/>
<a href="/Home/Details"> Student Details </a>
<br/>
<a href="/Home/GetTotal"> GetTotal </a>
<br/>
<a href="/Home/GetServerTime"> GetServerTime </a>
```

2. Details.cshtml:

```
<hr/>
<h2> Student Details </h2>.
```

27/10/15

Views in ASP.NET MVC

* In MVC, V- stands for View.
* Views are responsible for presentation (UI) logic
* In MVC application execution, view pages are processed by a special engine called "view engine".
* views are developed by using following technologies:
  - CS (C# code)
  - Html (client side)
    - → HTML Tags
    - → CSS
    - → JS
    - → client side libraries (JQuery, Angular JS, etc).

View Engine :-

→ What is view engine?

* View Engine is a sub-system of MVC framework.
* view Engine can independently process the views.
* If any changes in view pages, no need to compile the entire project.

→ What are View Engines supported by ASP.NetMVC?

* MicroSoft developed two view engines that are integrated with asp.netmvc.
  1. WebForm (ASPX)
  2. Razor.

1. WebForm (ASPX) :

→ webform viewEngine is introduced in ASP.NetMVC 1.0.

→ It uses same style of asp.net webform (*.aspx).

* writting Server code in *.aspx file is complex.
* It is not much comfortable for programmers to perform:
  development, Modifications and testing
* we need to take care of separating server code and client-side items.

2. Razor Engine:-

* Razor Engine was introduced in ASP.NET MVC 3.0.
* Razor Engine is called as advanced view engine.
* Compare with ASPX view Engine, Razor view-Engine provides more advantages.

Advantages:

* Easy to develop views.
* Less no. of characters required.
* Easy to read and understand
* Easy to perform unit testing
* Automatic recognize html tags in side C# code.

Razor programming:-

* view pages that are targeting Razor engine having the extension *.cshtml.
* every view page contains html tags and C# code.
* By default content will be consider as html by viewengine.
* we need to differntiate with a special symbol "@".
* we can integrate server code (C# statements) in the following ways:
  1. Single-Line statement.
  2. In-Line statement.
  3. Multi line statement.

1. Single Line Statement :-

* It is used to declare variables/objects, assignment statement, etc--

Syntax :

```
@{ c# statement; }
```

Eg: `@{ int x=10; }`

`@{ Emp obj = New Emp(); }`

`@{ count = count+1; }` ...

2. In-Line statement :-

* This statement is used to involve server variables inside html tag.
* Differentiate server variables by attaching "@" symbol. otherwise it will
* considered as html and display the variable name to end-user.

Eg: `<h1> Welcome to @Oname </h1>`

`<span> @obj.Ename </span>.`

`<span> @(i+1) </span>.`

3. Multi Line Statement :-

* It is used to write more than one line of c# code.
* It also allows html tags
* Razor engine automatically detects c# statement and html tag.

Eg:

```
@{
    int x = 124;
    if(x%2==0)
    {
    <h1> x is even </h1>
    }
    else
    {
    <h1> x is odd </h1>
    } }
```

Eg: Create a view page to demonstrate usage of razor programming statements.

Step 1: Create Asp.Net MVC application and add HomeController.

Step 2: Add view for Index() action method.

→ Rt. click on Index()
→ Add view
→ click "Add" button.

Step 3: prepare the following content in Index.cshtml.

```
@* 1. Single Line statement *@ → comment style in razor.
@{ string uname = "Satya"; }

@* 2. multi In Line statement *@ — comments
<h1> Welcome to @uname </h1>
<hr/>
@* 3. Multiline statement *@ → comments.
@{
    int x = 124;
    if (x%2 ==0)
    {
    <h2> @x is even. </h2>
    }
    else
    {
    <h2> @x is odd. </h2>
    }
}
```

28/10/15

Examples :

1) Create a view page to display student information

code for Index.cshtml :

```
@{
  int sno = 1025;
  string sname = "scott";
  string cname = "ASP.NET MVC";
}

<h1> Student details </h1>
<hr />
<span style="color:Blue;">
  Student Id : @sno
  <br />
    Student Name : @sname
  <br />
    Student Course Name : @cname
</span>
```

o/p :

Student details

Student Id : 1025

Student Name : scott

Course Name : ASP.NET MVC

(2) Create the view page to display collection of course names by using Razor program

```
@{
  string[] ar = {"classic ASP", "ASP.net", "Asp.Net MVC", "JQuery",
                 "XML", "JavaScript", "CSS", "HTML"};
}

<h1> Course Names </h1>
<hr />
<ul>
  @{
```

```
<ul>
@{
   foreach (string Item in ar)
     {
      <li> Item </li>
     }
  }
</ul>
```

o/p

Course Names

- Classic Asp
- Asp.Net
- Asp.Net MVC
- JQuery

③ Create a view page to course name in table format with list class.

```
@{
List <String> csList = new List <String>
{ "classic Asp", "ASP.NET", "C# ", "SQL", "AJAX" };
}
<h1> Course Names </h1>
<hr/>
<Table border = "2">
  <tr>
     <th> S.NO. </th>
     <th> Course Name </th>
  </tr>
@ {
    int n = 1;
   foreach (string item in csList)
   {
      <tr>
       <td> @n </td>
       <td> @item </td>
       </tr>
     n++ ;
   }
}
```

o/p :

Course Name.

S.NO	Course Name
1	classics
2	ASP.NET
3	C#
4	SQL
5	AJAX

<Form> tag

```
<Form action="URL" Method="post">
</Form>
```

Developing Data Entry forms in ASP.NET MVC :-

* End user will communicate with server by making requests from browser.
* These requested are divided into two types:
    1. Get Request.
    2. post Request.

1. Get Request :-

* we can make Get request either by entering url or click Hyperlink.
* The purpose of GET request is to get the page from server.

2. post Request :-

* we can make post request by click "Submit" button.
* Browser will submit the request to the URL that we specified in "action" attribute of <form/> tag.

Eg: <form action="/Home/Index" Method="post></Form>.

* Browser will submit input control values along with their names.

name-value pairs.

* In server code, we can read the value by using "Request" object.
* Request object will return the value in "String" format.

Eg: String s1 = Request["t1"];

→ In the above statement "t1" is the name of the textbox.

Identifying Request Type in Views :-

* If you want to prepare separate code for specific request, we need to identify the current request type.
* MVC view pages provides a property called "IsPost" to implement this
* IsPost is bool type

  Ispost – true (POST)

  IsPost – false (Get)

29/10/15

(4) Create a view page to display welcome message based on the username

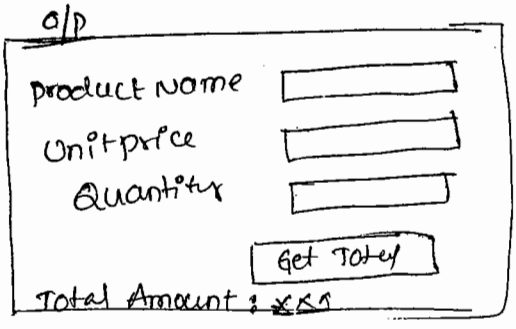
code for Index.cshtml :

```
<h1> Data Entry Form – Demo </h1>
<hr/>
<form action ="/Home/Index" Method ="post">
  Enter your Name:
<input type="text" id="t1" name ="t1" />
<br/> <br/>
<input type ="submit" value ="GetMessage" id ="sb1" name ="sb1" />
</form>
@{
  if (Ispost == true)
  {
   <span> This is post request </span>
   <br/>
   String s1 = Request ["t1"] ;
   <span> Welcome to @s1 </span>
  }
```

```
else
{
 <span> This is GET Request </span>
 }
}.
```

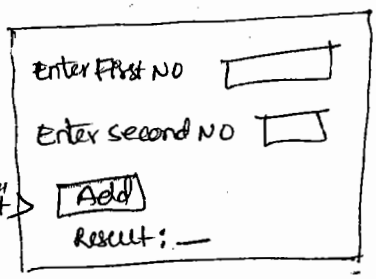
⑤ // Create a view page to get product price and total price.

```
<h1> product Details </h1>
<hr />
<Form action =" /Home/Index" method =" post">
  product Name :
<input type =" text" name =" pname" id =" pname" /> <br /> <br>
  Unit price :
<input type =" text" name =" uprice" id =" uprice" /> <br /> <br>
  Quantity :
<input type =" text" name =" Qty" id = " Qty" /> <br /> <br />
<input type =" submit" id = " sb1" name =" sb1" value =" Get Total" />
</Form>
<br />
@{
   if (IsPost == true)
   {
   int price = int.parse(Request [" uprice"]);
   int qty = int.Parse (Request [" Qty"]);
    int total = price * Qty ;
  <span> style =" Color : Red; "> Total Amount : <u> @total </u> </span>
  }
}.
```

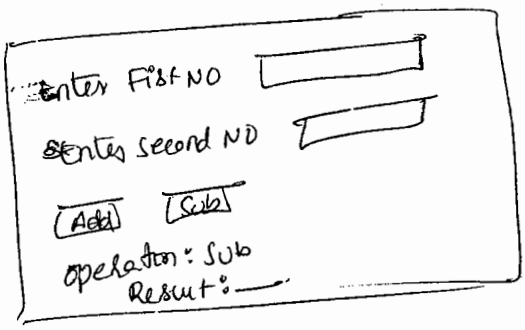


⑥ Create a view page to perform Sum operation.

```
<h1> Sum operation </h1>
<hr/>
<Form action = "/Home/Index" Method = "post">
   Enter First NO:
<input type = "text" name = "FirstNO" Id = "t1" /> <br/> <br/>
   Enter second NO:
<input type = "text" name = "SecondNO" Id = "t2" /> <br/> <br/>.
<br/> <br/>
<input type = "Submit" Id = "Sb1" name = "Sb1" value = "Add" />
</Form>
<br/>
@{ if (Ispost == = true)
   {
   int FNO = int.parse (Request ["FirstNO"]);
   int SNO = int. parse (Request ["second NO"]);
   int tot = FNO + SNO;
<span style = "color: Blue"> Result : @tot </span>
   }
}.
```



⑦ How to handle two Submit buttons in MVC



HTML Helpers in ~~View~~ MVC View Development:

* In MVC, views are developed by using HTML based content
* We need to prepare required design with basic html tags manually.
* Manual preparation of views will be time consuming.
* ASP.NET MVC framework introduced helper class that helps to generate html controls easily.
* These Helper methods returns string that contains required html tag.

Library Information:

NameSpace : System.Web.MVC

Class Name : Html Helper.

Object Name : Html.

Note: 'Html' is pre-defined object in view page, we can directly access "Html" object.

Helper Methods:-

1. @Html.Label()
2. @Html.TextBox()
3. @Html.Password()
4. @Html.TextArea()
5. @Html.CheckBox()
6. @Html.RadioButton()
7. @Html.DropDownList()
8. @Html.ListBox()
9. @Html.BeginForm()
10. @Html.Hidden.
11. @Html.ActionLink()

Note: HtmlHelper class doesnot provide Methods for buttons (Submit, Reset, Button).

Usage of BeginForm():

Syntax:

```
@using (Html.BeginForm("Action", "controller", FormMethod.post))
    {
     // required controls
    }
```

Eg:

```
@using (Html.BeginForm("Index", "Home", FormMethod.post))
   {
     // required controls
    }
```

output:

```
<Form action="/Home/Index" Method="post">
                ...
   </Form>.
```

Note: If you want to submit the values to same controller and same action, then no need to specify arguments in "BeginForm()".

(8) // Create Login page using Html Helpers.

~~code fo~~ → Add Login action Method to Home controller

Code for Login.cshtml.:

```
<h2> Login page </h2>
<hr/>
@using (Html.BeginForm())
{
  @Html.Label("User Name:")
  @Html.TextBox("Uid")
  <br/> <br/>.
```

```
@Html.Label("password:")
@Html.password("pwd")
<br/><br/>
<input type="submit" value="Login" id="sb1" name="sb1" />
}
@{
if (IsPost == true)
{
string s1 = Request["uid"];
string s2 = Request["pwd"];
if (s1 == "admin" && s2 == "admin")
{
<h2 style="color:Green;">Welcome to @s1</h2>
}
else
{
<h2 style="color:Red;">Invalid user id or password</h2>
}
}
}
```

30/10/15

Action Method parameters:

* In order to handle post request in action methods MVC Framework provides different types of action method parameters.
* We can organize the action methods for Get and Post request separately.
* We can create separate action methods for each request type.
* In order to differentiate these two methods, we can specify the corresponding details as follows:

1. Request Type attribute

```
[HttpGet]
[HttpPost]
```

2. Action method parameters.

2. Action Method parameters:-

* MVC Framework provides the following type of action method parameters (post Request):

1. Form Collection
2. Formal parameters
3. Model class object.

1. Form Collection:-

* Form collection is a pre-defined class in System.web.mvc namespace.
* Form collection works same as Request object.

* It stores the items in key-value pairs format.

Key – Name of the input field.

Value – Content in input field (String type).

Sample usage

```
[HttpPost]
public ActionResult Index(FormCollection frmobj)
{
  String str = frmobj["tr"];
  ------
  return View();
}.
```

2. Formal parameters :-

* In this option we can directly get the values in the form of pre-defined data type like : int, double, string, etc..

* If we this option, we need not to perform.

  - Reading data from collection
  - Type casting into required datatype.

* The above two activities will performed by controller in the background.

* Type casting is based on the data type of the parameter in action Method.

* Condition :- Name of parameter should be same as input field name

  Eg: TextBox Name.

Ex1: Create MVC Application to demonstrate usage of Action Methods parameters?

Code for
HomeController.cs :

```
namespace Mvcapplication
{
 public class HomeController : Controller
 {
   public ActionResult Index()
   {
     return View();
   }

[HttpGet]
 public ActionResult Login()
 {
   Response.Write ("this is GET Request");
  return View();
 }
[Http Post]
  public ActionResult Login(FormCollection frmobj)
  {
    Response.Write ("This is POST Request. <br>");

    String s1 = frmobj ["uid"];
    String s2 = frmobj ["pwd"];

 if (s1 == "admin" && s2 = "admin123")
 {
 Response.Write (" welcome to Admin");
 }
 else
 {
Response.Write (" Invalid userid or password");
 }
  return View();
}
```

```
[HttpGet]
  public ActionResult sum()
  {
   return view();
  }
 [Httppost]
   public ActionResult Sum(int x, int y)
   {
      int z = x+y; Response.write("Result:" +z);
     return view();
    }
}
```

code for Login.cshtml :

```
<h2> Login Page </h2>
 <hr/>
@ using (Html.BeginForm())
{
@Html.Label("User Name:")
@Html.TextBox("Uid")
<br/><br/>
@Html.Label("password :")
@Html.password("pwd")
<br/><br/>
<input type ="submit" value ="Login" Id ="sb1" name="sb1" />
}
```

Sum.cshtml :

```
<h2> Sum Operation </h2>
<hr/>
@using (Html.BeginForm())
 {
  @Html.Label("First- Number:")
   @Html.TextBox("x")
<br/><br/>
@Html.Label("Seeond Number:")
@Html.TextBox("y")
<br/> <br/>
<Input type="submit" value="Sum" id="Sbi" name="Sbi" />
 }
```

Index.cshtml :-

```
<h1> Action Method parameters Demo </h1>
<hr/>
<a href="/Home/Login"> Login </a>
<br/>
<a href="/Home/Sum"> Sum </a>
<br/>
```

31/10/15

3. Model class Object :-

* In database applications we need to transfer the values in the form of database related form object.
* In MVC, Model logic is prepared by using classes based on the database tables
* It required Model class object.
* TO handle these values action Method having a parameter called "Model class object".
* If you use this option, controller will perform all required steps:
    1. Create object
    2. Read the values from request
    3. Type casting
    4. Set the values to properties.

Condition: TextBox names in view should be same as property name of model class

Example: Create MVC application to procss Dept details by using Model class objects as Action method parameters.

1. /Models/Dept.cs :

→ Rt. click on "Models" folder and add class., Name is Dept.cs.

→ Click "Add" button.

```
Namespace MVCApplication50.Models
{
   public class Dept
   {
     public int Deptno { get; set; }
     public string Dname { get; set; }
     public string Loc { get; set; }
   }
}
```

Home Controller.cs

```
using MVCapplication50.Models;

namespace MVCapplication50.Controllers
{
  public class HomeController : Controller
  {
    public ActionResult Index()
    {
      return View();
    }
    [HttpPost]
    public ActionResult Index(Dept obj)
    {
      Response.Write("Deptno:" + obj.Deptno);
      Response.Write("<br> Dname:" + obj.Dname);
      Response.Write("<br> Loc :" + obj.Loc);
      return View();
    }
  }
}
```

Index.cshtml : (view)

```
<h2> Create Dept </h2>
<hr/>
@using (Html.BeginForm())
{
  @Html.Label("Deptno:")
  @Html.TextBox("Deptno")
  <br/> <br/>
```

```
@Html.Label("Dname:")
@Html.TextBox("Dname")
<br/> <br/>
@Html.Label("Location:")
@Html.TextBox("Loc")
<br/> <br/>
<input type="Submit" value="Create" id="sb1" name="sb1" />.
}.
```

GET Action Method Parameters :

* In some of the applications we need to send the values along with URLs to process the request.
* These values can be processed with the help of "GET" action method.
* ASP.NET MVC Framework provides the following ways to handle the Get action method parameters:
    1. Query String
    2. Route URL Parameter

1. Query String :-

* This is standard approach which is used all web Technologies.
* Attaching the name and values along with URL.
* It uses "?" to specify the values in URL.
* If use query string, you can use any name for parameter.

Eg: /Home/Details?dno=10
/Home/Edit?dno=10

Condition : Action method should be implemented with same name of the query string parameters name.

2. Route URL Parameter :-

* According to Route URL format we can attach additional value along with URL.
* This value is considered as "id". (Refer RouteConfig.cs)
* By default 'id' is optional. If you want you can specify.

Eg: /Home/Details/10

/Home/Edit/10.

Condition : Action method parameter name should be "id".

Example (2) // Create MVC Application to demonstrate of GET Action Method parameters

* HomeControllers.cs :

```
namespace MVCApplication52.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
return View();
}
// Get Action - Query string
public ActionResult Details1(int dno)
{
Response.Write("Requested Deptno :" + dno);
} return view();
```

```
// Get Action - Route Url parameter

public ActionResult Details2 (int id)
  {
  Response.Write ("Requested Deptno:" +id);
     return View();
   }
} }
```

Views

Index.cshtml:

```
<h1> GET Action Method parameters Demo </h1>
  <hr/>
  <h2> Using QueryString </h2>
<a href ="/Home/Details1? dno =10 "> Accounts </a>
<br/>
<a href =" /Home/Details1? dno = 20"> Sales </a>
<br/>
<a href ="/Home/Details1? dno = 30"> Marketing </a>.
<hr/>
  <h2> Using Route URL parameters </h2>
<a href ="/Home/Details2/10 "> Accounts </a>
<br/>
<a href = "/Home/ Details2 / 20"> Sales </a>
<br/>
<a href =" /Home/ Details2 / 30"> Marketing </a>
<br/><br/>
<hr/>.
```

→ Add views for Details1 and Details2 action Methods.

Details.cshtml : <h2> Details 1 </h2>

Details2.cshtml : <h2> Details 2 page </h2>.

02/11/15

Datasharing between controller to view :-

Q): How to share the data from Controller / Actionmethod to view? /

What are the datasharing techniques used to transfer the result from controller to view? /

What are the Asp.Net MVC specific techniques to Maintain state between controller to view?

Ans: Every request that will come to Asp.Net MVC received by controller and processed by corresponding action Method.

→ After procesing the request, results should be displayed in view,

→ ASP.NET MVC provides following techniques in order to transfer the results from controller's action Methods to view:

1. View Data
2. View Bag
3. Temp Data
4. Strongly Typed view.

1. View Data :-

* ViewData is collection object.
* It stores the values in the form Key-Value.

  Key - String

  Value - Object.

* We need to perform typecasting in order to read the values from view data because it stores all the values in "object" (root class) format.

Usage:

```
ViewData["Key"] = Value; // Set.

Variable = (type) ViewData["Key"]; // Get.
```

Note :-

* View Data is an object of "ViewDataDictionary" class.

2. ViewBag:

* ViewBag is also used for sending data from controller to view.
* viewBag is also same as viewData with few differences.
* viewBag is introduced in ASP.Net MVC 3.0.
* viewBag concept is developed on "dynamic typing" concept of C# 4.0.
* If you use viewBag, no need to perform typecasting.
* In ViewBag, Keys are used like a properties.

Usage:

```
ViewBag.Key = Value; // Set (or) ViewBag.variableName = Value;

variable = ViewBag.Key; // get. (or) ViewBag.variableName.
```

① Create MVC Application, process product data and store the result using ViewData and ViewBag.

HomeController.cs:

```
Namespace MVCApplication54.Controllers
{
 public class HomeController : Controller
  {
   public ActionResult Index()
   {
    return View()
    }
    [Http post]
    public ActionResult index (string Pname, int price, int qty)
  {
    int n = Uprice * qty;
    viewData["productName"] = Pname;
     viewBag.Total = n;
     return view();
  }
}
```

Index.cshtml:

```
<h2> product Details </h2>
<hr/>
@using(Html.BeginForm())
{
 @Html.Label("product Name:").
 @Html.TextBox("Pname")
 <br/> <br/>
 @Html.Label("unit price:")
 @Html.TextBox("Uprice")
  <br/> <br/>
 @Html.Label("Quantity:")
 @Html.TextBox("qty")
```

```
<br/> <br/>
<input type="Submit" id="Sb1" name="Sb1" value="Get Total" />
};
</hr> </hr>
@{
  if (IsPost ==true)
  {
  string str = (String) ViewData ["product Name"];
     int t = ViewBag.Total;
<span> product Name:@str </span>
  <br/>
  <span> Total Amount : @t </span>
  }
}
```

Note :

* ViewBag is also uses "ViewDataDictionary" class internally to organize the items.
* viewData and viewBag are collections based object so that we can store multiple items in them.

3. TempData :-

* TempData is also same as ViewData . stores the items in object format. But TempData is available in current action and redirected action.
* viewData/ viewBag scope is only one action and corresponding view.
* TempData scope is current action, redirected action and its view.

usage :

```
TempData ["Key"] = value; // set
   variable = (Type) TempData ["Key"]; // Get.
```

Note :

→ TempData is an object of "TempDataDictionary" class.

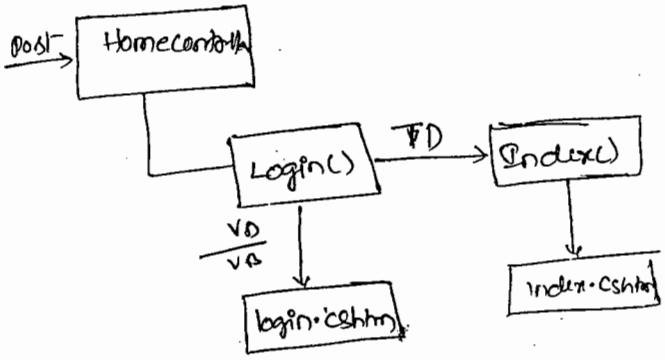
→ Tempdata will be cleared after processing redirected request.

(2) // create MVC application to share data between action methods using tempdata.

Code for HomeController.cs :

```
namespace MVCApplication55.Controllers
{
 public class HomeController : Controller
 {
  public ActionResult Login()
  {
   return View();
  }
 [Http post]
 public Action Result Login(string uid, string pwd)
  {
   if(uid == "Admin" && pwd == "admin123")
   {
 TempData ["Username"] = uid;
 return RedirectToAction("Index");
 }
 else
 {
  ViewBag.ErrorMessage = "Invalid user id or password";
     return View();
  }
}
```

```
public ActionResult Index()
{
  return View();
}
}
```



Code For Login.cshtml

```
<h2> Logic page </h2>
<hr/>
@Using(Html.BeginForm())
{
 @Html.Label("Username");
 @Html.TextBox("Uid")
<br/> <br/>
@Html.Label("password :");
@Html.TextBox password("pwd")
<br/> <br/>
<Input type = "submit" value = "Login" id = "sp1" name = "sb1" />
}
<br/>
@{
 if(IsPost == true)
 {
  string str = viewBag.Errormessage;
  <span style = "color:Red;"> @str </span>
} }
```

(update Route config.cs)

Code Index.cshtml

```
<h2>Index Page</h2> <br/>
@{
  string str = (string) Tempdata ["UserName"];
  <h2> Welcome to @str </h2>
}
```

3/11/15

* List <int> obj = new list <int> ();

4 Strongly Typed Views :-

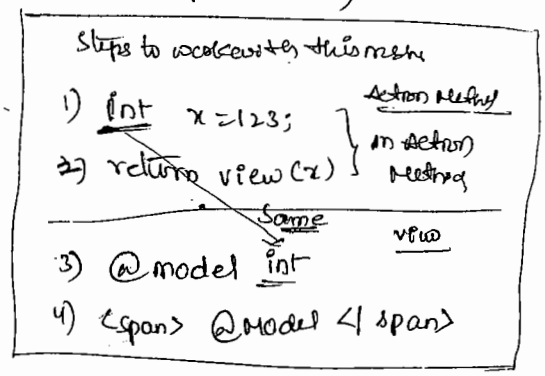
Q) What is a Strongly-typed view?

Ans: If the view is designed by targeting specific Model class object/ objects, then that view is called "Strongly Typed view".

* In Strongly Typed view, view is bind with corresponding Model class object/ objects.

* Most of the database operations related views are developed by using strongly typed view.

* Scaffolding Templates (view templates for DB operations) works based on strongly typed views only.



Benefits :

→ It supports intellisense
→ Design time type verification.
→ No need to perform type casting.
→ More comfortable for database related operations.

Ex : 1. Dept obj = new Dept(); obj.Deptno = 10; obj.Dname = "Account";
2. return view(obj);
3. ~~@using~~ @using MVCApplication.Models. → this is for user defined, For predefined need not required (like int, string etc.
@Model Dept.
4. @model.Deptno
@ model.Dname
(Property)

Steps to Work with - Strongly Typed View :-

1. prepare data to send view.
2. Send data using View() Method.

} Action Method

3. Specify data type using @model directive.
4. Access data using Model property

} View section.

Note : If we are using user defined data-types in strongly views then you should import corresponding namespace in view.

Ex: @using MVCApplication1.Models ✓
3. @Model Dept. ✓

Example : Create MVC Application to display department details by using strongly Typed view.

1. Models/Dept.cs :

```
namespace MVCApplication56.Models
{
 public class Dept
 {
  public int Deptno {get; set;}
  public string Dname {get; set;}
  public string Loc {get; set;}
} }
```

2. HomeController.cs

```
using MVApplication56.Models
Namespace MVCApplication56.Controllers
{ public class HomeController : Controller
{
```

```
public ActionResult Index()
{
  //Step1
   Dept obj = new Dept();
     obj.Deptno=10";
     obj.Dname="Accounts";
      obj.Loc="Hyd";
   //Step2
    return View(obj);
  }
public ActionResult Index2()
{
 List<Dept> deptList = New List<Dept>();

Dept d1 = New Dept() {Deptno = 10", Dname="Accounts"; Locaton ="Hyd"};
Dept d2 = New Dept() {Deptno =20"; Dname="Marketing"; Location ="Chennai"};
Dept d3 = New Dept() {Deptno = 30"; Dname ="Sales"; Locaton ="Mumbai"};

    return (deptList)
     deptList.Add(d1);
     deptlist.Add(d2);
      deptlist.Add(d3);
  return View(deptList);
  }
}
}
```

Index.cshtml: @* slip 3 *@

```
@ Using MvcApplication56.Models
@Model.Dept
  <h1> Department Details </h1>
 <hr/>
 @* slip 4 *@
 <span>
    Dept No: @model.Deptno <br/>
    Dname : @Model.Dname <br/>
    Location : @Model.Loc. <br/>
    </span>
```

Index2.cshtml:

```
@using MVCApplication56.Models
@model List<Dept>
 <h1> All Department Details</h1>
 <hr/>
<table boader ="2">
  <tr>
    <th> Deptno </th>
    <th> Dname</th>
    <th> Loc </th>
  </tr>
@{
  foreach (Dept item in Model)
  {  <tr>
       <td> @item.Deptno </td>
       <td>@item.Dname </td>
    </tr> <td> @item.Loc </td>
```

}}

</table>

4/11/15

Note : we can replace @model List <Dept> statement with following statement in Index2.cshtml

@model IEnumerable <Dept>

→ List class implemented IEnumerable interface.

Models in ASP.NET MVC :-

→ In MVC, M- stands for Model

→ Model in MVC contains database related logic.

→ According to MVC, "Model" represents classes that used to perform database operations.

→ It helps us to perform all database operations in entire MVC applications.

→ Model will communicate with database.

→ Models in MVC is implemented by using following Techniques:

1. ADO.NET
2. Entity Framework.

classes in Model :

→ the classes that are preparing in model are categorized into two types

1. Entity Classes.
2. DataContext classes.

1. Entity classes:-

→ Entity classes are used to storing and transfer the data within your application.

→ Entity class represents database tables.

→ class name is similar to database table columns names.

2. DataContext classes:-

→ Data Context classes contains the logic to perform database operations.

→ It will define all the required methods to communicate with databases.

→ It will receive entity class objects and send those details to database.

Example: Create ASP.NET MVC Application to perform CRUD operations on dept table using ADO.NET.

Index.cshtml:

```
@ using MVCApplication57.Models
@ model IEnumerable < Dept>
 <h1> All Dept Details </h1>
 <br/>
<a href =" /Home/ Create" > Create New </a>
 <br/><br/>
<table border ="2">
    <tr>
      <th> Deptno </th>
      <th> Dname </th>
       <th> Loc </th>
      <th> Operations </th>
  </tr>
@{
```

```
Foreach (Dept item in model)
{
  <tr>
    <td> @item.Deptno </td>
    <td> @item.Dname </td>
    <td> @item.Loc </td>
<a href="/Home/Edit/@item.Deptno"> Edit </a> |
<a href="/Home/Delete/@item.Deptno"> Delete </a> |
<a href="/Home/Details/@item.Deptno"> Details </a>
    </td>
  </tr>
}
</table>
```

Create.cshtml :

```
<h2> Create New Dept </h2>
<hr>
@using (Html.BeginForm())
{
@Html.Label("Deptno :")
@Html.Textbox("Deptno")
<br /> <br />
@Html.Label("Dname :")
@Html.Textbox("Dname")
<br /> <br />
```

```
@Html.Label("Location:")
@Html.TextBox("loc")
<br/> <br/>
<input type="Submit" id="sb1" name="sb1" value="Create" />
}
<br/>
<a href="/">Back to Index</a>
```

Details.cshtml :

```
@using mvcApplication57.models
@model Dept
<h1> Depastment Details </h1>
<hr/>
<span>
Deptno : @Model.Deptno <br/>
Dname : @model.Dname <br/>
Location : @model.Loc <br/>
</span>
<br/>
<a href="/"> Back to Index </a>.
```

HomeController.cs :-

```
using mvcApplication57.models;
namespace MVCApplication57.Controllers
{
public class HomeController : Controller
{
DataContext db = New DataContext();
```

```
List<Dept> deptlist = db.GetDepts();
    return view(deptlist);
}

public ActionResult Create()
{
    return view();
}

[HttpPost]
public RedirectToAction("Index");

public ActionResult Create(Dept obj)
{
    db.AddDept(obj);
    return RedirectToAction("Index");
}

public ActionResult Details(int id)
{
    Dept obj = db.GetDept(id);
    return view(obj);
}

public ActionResult Edit(int id)
{
    Dept obj = db.GetDept(id);
    return view(obj);
}

[HttpPost]
public ActionResult Edit(Dept obj)
{
    db.EditDept(obj);
```

```
    return RedirectToAction("Index");
  }

  public ActionResult Delete(int id)
  {
     Dept obj = db.GetDept(id);
     return View(obj);
  }
  [HttpPost]
  public ActionResult Delete(String id)
  {
    int n = int.Parse(id);
     db.DeleteDept(n);
    return RedirectToAction("Index");
  }
}
```

Edit.cshtml:

```
@using MVCApplication57.Models
@Model Dept
<h2> Edit Dept </h2>
<hr/>
@using(Html.BeginForm())
{
  @Html.Label("Deptno:")
  @Html.Textbox("Deptno", Model.Deptno, new {@readonly=true})
    <br/><br/>
@Html.Label("Dname:")
@Html.TextBox("Dname", model.Dname)
<br/> <br/>
```

```
@Html.Label("Location:")
@Html.TextBox("Loc", Model.LOC)
<br/><br/>
<input type="Submit" id="Sb1" name="Sb1" value="Edit" />
}
<br/>
<a href="/">Back to Index</a>
```

Delete.CSHtml:

```
@ using MVCApplication57.Models
@model Dept
<h1> Delete Department </h1>
<hr>
<span>
Deptno: @model.Deptno <br/>
Dname: @model.Dname <br/>
Location: @model.Loc <br/>
</span>
<br/>
@using (Html.BeginForm())
{
<span> Do you want to delete? </span>
<input type="submit" id="Sb1" name="Sb1" value="Delete" />
}
<br/>
<a href="/"> Back to Index </a>.
```

Write Entity classes & DataContext class.

ASP.NET MVC 5.2

Model classes to perform CRUD Operations on Dept table using ADO.NET

By Mr.Narasimha

File1: /Models/Dept.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MvcApplication1.Models
{
    public class Dept
    {
        public int Deptno { get; set; }
        public string Dname { get; set; }
        public string Loc { get; set; }
    }
}
```

FileName: /Models/DataContext.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

using System.Data;
using System.Data.SqlClient;
```

```
namespace MvcApplication1.Models
{
    public class DataContext
    {

        public List<Dept> GetDepts()
        {
            List<Dept> deptList = new List<Dept>();
           string connStr = "Server=narasimha-pc;
       Database=TestDb; Integrated Security=true;";
            string cmdText = "SELECT * FROM DEPT";

       SqlDataAdapter da = new SqlDataAdapter(cmdText,
    connStr);
            DataTable dt = new DataTable();
            da.Fill(dt);

            foreach (DataRow item in dt.Rows)
            {
                Dept obj = new Dept();
                obj.Deptno = (int)item["Deptno"];
                obj.Dname = (string)item["Dname"];
                obj.Loc = (string)item["Loc"];

                deptList.Add(obj);
            }
            return deptList;
        }
```

```
        public Dept GetDept(int n)
        {
            Dept obj = new Dept();

            string cmdText = "SELECT * FROM DEPT WHERE
DEPTNO=" + n;
            string connStr = "Server=narasimha-pc;
        Database=TestDb; Integrated Security=true;";

      SqlConnection conn = new SqlConnection(connStr);
      SqlCommand cmd = new SqlCommand(cmdText, conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();

            if (dr.HasRows == true)
            {
                dr.Read();
                obj.Deptno = (int)dr["DEPTNO"];
                obj.Dname = (string)dr["DNAME"];
                obj.Loc = (string)dr["LOC"];
            }

            dr.Close();
            conn.Close();
            return obj;
        }
```

```
        public void AddDept(Dept obj)
        {
            string cmdText = string.Format("INSERT INTO
DEPT VALUES({0}, '{1}', '{2}')", obj.Deptno, obj.Dname,
obj.Loc);
           string connStr = "Server=narasimha-pc;
        Database=TestDb; Integrated Security=true;";

  SqlConnection conn = new SqlConnection(connStr);
  SqlCommand cmd = new SqlCommand(cmdText, conn);
            conn.Open();
            cmd.ExecuteNonQuery();
            conn.Close();
        }

        public void EditDept(Dept obj)
        {
            string cmdText = string.Format(" UPDATE DEPT
SET  DNAME='{0}' , LOC='{1}'  WHERE DEPTNO={2}",
obj.Dname, obj.Loc, obj.Deptno);

           string connStr = "Server=narasimha-pc;
        Database=TestDb; Integrated Security=true;";

SqlConnection conn = new SqlConnection(connStr);
SqlCommand cmd = new SqlCommand(cmdText, conn);
            conn.Open();
            cmd.ExecuteNonQuery();
            conn.Close();
        }
```

```
        public void DeleteDept(int n)
        {
            string cmdText = string.Format(" DELETE FROM
DEPT WHERE DEPTNO={0}", n);
            string connStr = "Server=narasimha-pc;
        Database=TestDb; Integrated Security=true;";

SqlConnection conn = new SqlConnection(connStr);
SqlCommand cmd = new SqlCommand(cmdText, conn);
            conn.Open();
            cmd.ExecuteNonQuery();
            conn.Close();
        }
    }
}
```

➔ Prepare similar type of classes for Emp and Student Tables.

➔ Prepare the Emp and Student tables in Database

- **Emp** : Empno, Ename, Job, Sal, Deptno
- **Student** : StudentId, Sname, Course, Email, ContactNo

➔ **Develop MVC Applications to implement the CURD operations on these tables.**

06/11/15

Entity Framework 6.0

* Entity Framework is an ORM Tool developed by Microsoft
* EF is used to perform all database operations easily.
* EF will generate required classes to perform CRUD operations.
* EF was introduced in the year 2009.

What is ORM?

→ ORM stands for Object Relational Mapping.
→ ORM concept will mapping relational tables with programming class objects
→ There are several ORM tools are developed based on ORM concept.

What are the examples of ORM tools?

1. Entity Framework (.NET)
2. Hibernate (Java)

What are the advantages of ORM tools? (or) what are the advantages of EF?

→ EF generates required code to perform database operations automatically.
→ Implementation of database operations will be more easy.
→ Reduce the coding burden on the programmer.
→ It makes application development will be more faster.

Note :-
→ All ORM tools generates two types of classes:
1. Entity classes.
2. DataContext Class.

Entity Framework Methods:

operation	VS 2010	VS 2012 / VS 2013
Create	→ AddObject()	→ Add()
Read All	→ ToList()	→ ToList()
Read Single	→ Single or Default()	→ Find()
Update	→ —	→ Entry()
Delete	→ DeleteObject()	→ Remove()
Update the changes in database	→ SaveChanges()	→ —

Example: Create MVC Application to Read & Create operation on Dept table using Entity Framework?

1. Create ASP.NET MVC Application
2. Add "Dept" table using Entity framework in models folder.
   - → Goto solution Explorer.
   - → Rt. click on [Models] Folder.
   - → Add → New Item.
   - → Visual C#
     - ↳ Data-
   - → select "ADO.NET Entity Data Model" file
     - Click "Add" button.
3. provide the required details in "Entity Data Model" wizard.
   - → generate from Database
   - → provide Connection details.
   - → select required tables.
   - → click 'finish' button.

Note :- This step will generate required classes to perform db operations. Build the project to update in your application.

4. Add Home Controller and define required action methods:

HomeController.cs:

```
using MVCApplication58.Models;
namespace MVCApplication58.Controllers
{
public class HomeController: Controller
{
SafeydbEntities db = new SafeydbEntities();
public ActionResult Index()
{
List<Dept> deptlist = db.Depts.ToList();
return View(deptlist);
}
public ActionResult Create();
{
return view();
}
[HttpPost]
public ActionResult Create(Dept obj)
{
db.Depts.Add Objects (obj);
db.SaveChanges();
return RedirectToAction("Index");
}
}
}
```

5. Add Views for action methods

→ Index.cshtml

→ Create.cshtml

(Refer prev example views to prepare the views ADO.NET Example)

01/11/17

Scaffolding Templates in MVC :-

* Scaffolding templates are used to generate view pages automatically to perform database operations.

* It generates the required content for creating views for CRUD operations.

* MVC Framework provides following scaffolding templates to create views for database operations (CRUD operations)

1. List (Display all items)
2. Create (Insert)
3. Edit (Update)
4. Delete (Delete)
5. Details (Display single item).

Note :

* It will generate only required content, later we can customize (modify) as per our requirements.

* Scaffolding templates works based on the model class (entity class) as strongly-typed view.

What are the advantages of scaffolding templates?

1. Makes views preparation is easy for database operations.
2. It will reduce the coding burden on the programmer.
3. Makes application development more faster.

Steps to generate views with scaffolding templates:-

→ Rt. click on Action method.
→ Add view
→ provide the required details:
  ↳ Check "Create a strongly-typed view" checkbox
  ↳ Model class : Emp.
  ↳ scaffolding Template : List.
→ click 'Add' Button

Note:

In lastest versions of visual studio, we need specify Data context class also

Data Context class : SatyaDbEntities.

Example 1:

To create Asp.net MVC Application to perform CRUD operations on Emp table using scaffolding templates based on entity framework

Step 1 : Create new MVC Application and add Emp table to models by using EF. and Build the project.

Step 2 : Add HomeController and prepare the required Action methods.

- File : HomeController.cs

```
using MVC-Application 59. Models;

name space MVC-Application59. Controllers
{
public class HomeController : Controller
{.
 SatyaDbEntities db = New SatyaDbEntities();
 public ActionResult Index()
{
List <Emp> emplist- = db. Emps . ToList();
 return view (emplist);
 }
 public ActionResult Creat()
 {
  return view();
 }
 [Http Post]
 public ActionResult Creat (Emp obj)
 {
    db. Emps . Add object (obj);
    db . Savechanges();
 return RedirectToAction ("Index");
  }
 public ActionResult Details (int id)
 {
  Emp obj = db. Emps. SingleOrDefault (x => x . Empno == id );
    return view(obj);
  }
```

```
public ActionResult Delete(int id)
{
  Emp obj = db.Emps.SingleOrDefault(x => x.Empno == id);
    return View(obj);
}
[HttpPost]
public ActionResult Delete(String id)
{
  int n = int.Parse(id);
  Emp obj = db.Emps.SingleOrDefault(x => x.Empno == n);
    db.Emps.DeleteObject(obj);
     db.SaveChanges();
  return RedirectToAction("Index");
}
public ActionResult Edit(int id)
{
  Emp obj = db.Emps.SingleOrDefault(x => x.Empno == id);
    return View(obj);
}
[HttpPost]
public ActionResult Edit(Emp obj1)   (carries updated values)
{
Emp obj2 = db.Emps.SingleOrDefault(x => x.Empno == obj1.Empno);
     obj2.Ename = obj1.Ename;
     obj2.Job   = obj1.Job;
     obj2.Sal   = obj1.Sal;
     obj2.Deptno = obj1.Deptno;
```

```
        db.SaveChanges();
        return RedirectToAction("Index");
    }
}
}
```

4. Add view pages for action methods by using scaffold templates.

Note: If hyperlinks lines not prepared properly in Index.cshtml (list template) update with the following statement:

Delete:

```
@Html.ActionLink("Edit", "Edit", new { /* id=item.EmpNo */ })
```

Update:

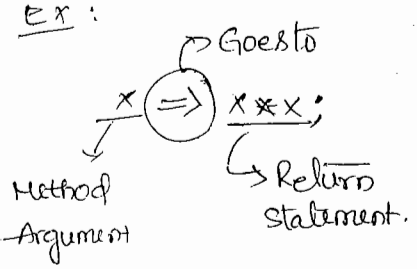
```
@Html.ActionLink("Edit", "Edit", new { id=item.EmpNo })
```

09/11/15

Lambda Expressions :-

* Lambda expressions are used to simplify the usage of delegates in .net applications.
* If we use lambda expressions, no need to prepare:
  - Separate class
  - Separate Method.
  - Object for the class.
* Directly we can provide the required details with simple expressions.
* This concept introduced in C# 3.0.
* In C# 2.0, anonymous methods are introduced. But lambda expressions are more simplified that "Anonymous Methods".

Ex:



Example 1:

Write a C# program to implement delegate usage with regular approach, anonymous methods and lambda expressions.

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication35
{
// Delegate declaration
delegate int MathDelegate(int x);
class Test()
{
```

```
public int Square (int x)
  {
    int z = x * x;
     return z;
  }
}

public class program
{
   static void Main()
{
  Test obj = New Test();
   int result = 0;
// Regular usage C#1.0.
MathDelegate d1 = New MathDelegate (obj.Square);
  result = d1(5);
C.WL (" Regular usage of Delegate");
 C.WL(" Result:" +result);
  C.WL();
// Using Anonymous Methods - C# 2.0
MathDelegate d2 = delegate (int x)
                  {
                    int z = x * x;
                    return z;
                  };
   result = d2(5);
  C.WL (" Using Anonymous methods");
   C.WL (" Result:" + Result);
  C.WL();
```

```
// Lambda Expressions - C# 3.0
MathDelegate d3 = x => x * x;
  result = d3(5);
c.WL("Using Lambda Expressions");
  c.WL("Result:" + Result);
  c.WL();
  c.RL();
 }
}
```

→ Applications of lambda expression:

1. Filtering data from collections
2. Entity Framework.
3. Scaffolding Templates.

Example-2 : // write a program to process integer array by using lambda expressions in orders to findout the counts based specified condition.

```
using System;
U.S. collections.Generic;
U.S. Linq;
U.S. Text;
namespace consoleApplicaton3J
{
 public class program
 {
  static void Main()
 {
 int[] ar = {31,7,10,28,9,5,15,35,4,2,13,20,69,24};
    int n=0;
  n = ar.Count();
   c.WL("All Items:" +n);
```

```
n = ar.Count(x => x%2==0);
c.WL("Event numbers:" +n);

n = ar.Count (x => x%2==1);
c.WL("odd numbers:" +n);

n = ar.Count (x => x%5==0);
c.WL("Five Multiples:" +n);

n = ar.Count (x => x%2==0 && x>=20);
c.WL("Event numbers greater than 20 are:" +n);
Console.ReadLine();
}
}
```

10/11/15

LINQ

* LINQ :- Language Integrated Queries.
* LINQ concept integrated SQL style of querying techniques in .NET programming languages (C# / VB.NET)
* It is used to communicate with different data sources (collections).
* Entity Framework will returns the database values in collections format.
* If you want to filter / sort these values, you can use LINQ programming.
* NO need to execute any commands across the database.
* Use LINQ Queries to extract the data from the EF collections in .NET Applications.

LINQ programming Implementation:

→ In order to implement LINQ, we have to use following steps:

1. prepare data source.
2. Create LINQ Query
3. Execute LINQ Query.

Note:

1. Linq queries always returns the results as "IEnumerable<T>".
   - In this 'T' refers datatype which is depends on the collection type.
2. Library Information:

   Library : System.Core.dll

   Namespace : System.Linq.

Example1:- write a c# program to extract the data from array by using LINQ.

```
using System;
using System.Linq;

class program
{
  static void Main()
  {
  //step1

int[] ar = {10, 27, 33, 93, 25, 40, 38, 94, 91, 9, 4, 13};

// step 2

var q1 = From x in ar
         where x%2==0
         orderby x descending
         select x;

  //Step3
  foreach(int item in q1)
  {
    C.W(" " +item);
  }
   C.WL();
  }
  C.RL();
}
```

Sample Queries:

1) var q1 = from n in ar
where n%2==0
Select n;

2) var q2 = From n in ar
Select n;

3) var q3 = from n in ar
where n>=50
Select n;

4) var q4 = from n in ar
where (n%2==0 && n>=50)
selectn;

5) var q5 = From n in ar
Order-by n
Select n;

6) var q6 = From n in ar
Orderby n descending
Selectn;

7) var q7 = From n in ar
where n%2==0
Orderby n
Select n;

Note:

Query variable provides a method called "count" to know no. of items in the result.

Usage: int n = q1.Count();

Example2 : create MVC Application to filter employee data based on department number by using LINQ Query; display the result descending order of their salaries

Step 1 : Create New MVC Application and add 'Emp' table to models using EF. and Build the project.

2. Add Home controller and prepare action methods :

```
        using mvcApplication60.Models;

namespace MvcApplication60.Controllers
{
  public class HomeController : Controller
  {
    public ActionResult Index()
    {
      return View();
    }

SathyaDbEntities db = New SathyaDbEntities();

public ActionResult EmpDetails(int id)
{

 var q1 = From e1 in db.Emps.ToList()
          where e1.Deptno == id
          orderby e1.Sal descending
          select e1;

      return View(q1);

  }
}
}
```

Index.cshtml :

```
<h2> Filter data using LINQ </h2>
<hr/>
<a href="/Home/Empdetails/10" >Accounts </a> <hr/>
<a href="/Home/Empdetails/20"> Sales </a> <hr/>
<a href="/Home/Empdetails/30"> Marketing </a> <hr/>
<a href="/Home/Empdetails/40"> operations </a> <hr/>
```

Empdetails.cshtml :

→ Add view by using scaffold template – List template

→ Add the following line to display employees counts

```
<h3> No. of Employees : @Model.Count() </h3>.
```

12/11/15

Relationship in Entity Framework :-

* Entity Frameworks supports all types relationships between the entities.
* These relationships are prepared according to database tables.
* Based on the database table relations, EF will generate relation ships in application.
* EF generates special property in entity classes called "Navigation properties".
* By using these navigation properties", we can access the data from other table without writing any joins.

Eg: Suppose If "Emp" table having foreign key relation with "Dept" table.

1. A department can have multiple employees

→ EF generates a navigation property called "Emps".

2. An Employee belongs to only one department:

→ EF generates a navigation property "Dept".

Example: Create MVC Application to display data from emp and Dept tables by using EF Relations.

Hint: Create Dept and Emp table with foreign key relations in your data base.

1. Create MVC Application and Add Emp, Dept tables to models by using EF.

2. You can find relations b/w entities with navigation properties in Model1.edmx file.

→ Build the project.

3. Add the HomeController and prepare the methods.

HomeController.cs:

```
using MVCApplication61.Model1;
namespace MVCApplication61.Controllers
{
  public class HomeController : Controller
  {
  SatyaDbEntities db = new SatyaDbEntities();
  public ActionResult Index()
  {
  List<emp> emplist = db.Emps.ToList();
    return View(emplist);
  }
  public ActionResult Index2()
  {
  List<Dept> Deptlist = db.Depts.ToList();
  return View(Deptlist);
  }
}
```

Index.cshtml

```
@ using MVCApplication64.Models
@ model List<Emp>
<h1> Employee Details </h1>
<hr/>
<Table border="2">
 <tr>
 <th> Ename </th>
 <th> Job </th>
 <th> Dname </th>
 <th> Location </th>
</tr>
@{
foreach (Emp item in Model)
 {
 <tr>
    <td> @item.Ename </td>
    <td> @item.Job </td>
    <td> @item.Dept.Dname </td> <td> @item.Dept.Loc </td>
 </tr>
 }
</table>
```

Navigation property which contains Depart object which depart ment employees

Index2.cshtml

```
@using MvcApplication64.Models
@model IEnumerable<Dept>

<h1> Department Information </h1>
<hr>
<table border="2" width="400px">
  <tr>
      <th> Deptno </th>
      <th> Dname </th>
      <th> EmpNames </th>
  </tr>
@{
foreach (Dept item in model)
{
  <tr>
      <td> @item.Deptno </td>
      <td> @item.Dname </td>
      <td>
        <ul>
          @{
            foreach (Emp item1 in item.Emps)
            {
              <li> @item1.Ename </li>
            }
          }
        </ul>
      </td>
  </tr>
}
</table>
```

navigation property

13/11/15

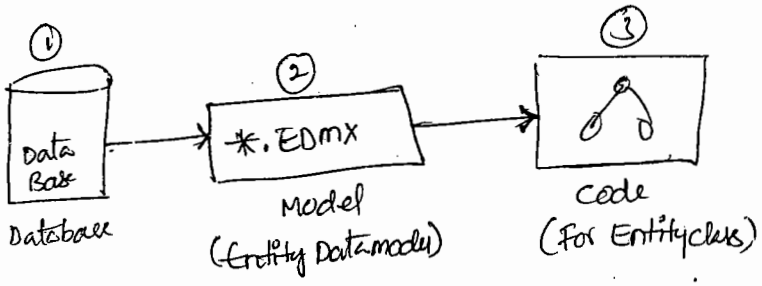
Entity Framework Implementation Approaches:

* Entity Framework provides different approaches in order to implement database operations.

* All these operations will do same tasks, but the way of implementation will be different.

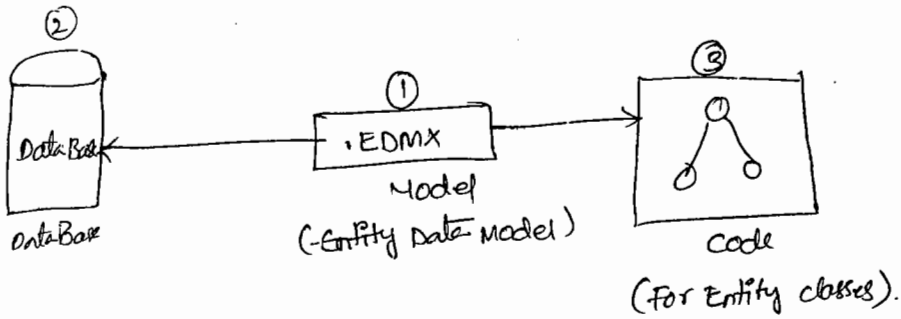
1. Database first approach
2. Model First approach
3. Code First approach.

1. Database First approach :-



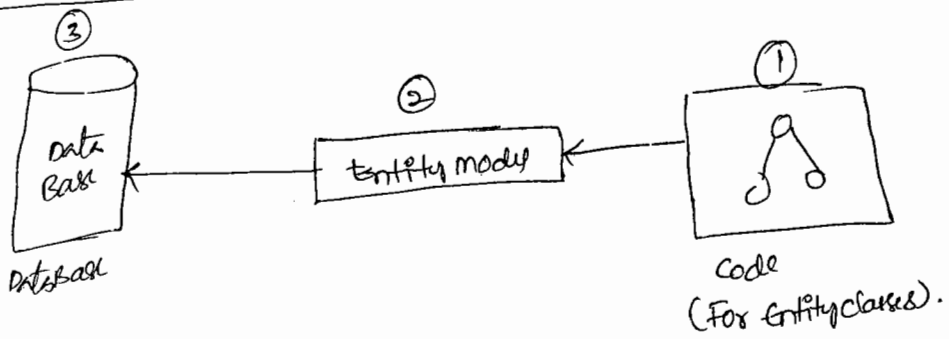
1. In this approach, first we have to create data base and corresponding tables
2. Based on the data base tables, we need to create Entity Data Model File(.Edmx) file and code for Entity class.
3. It is the first approach introduced in Entity Framework.
4. In this approach, Entity classes and Entity data model will autogenerate by Entity Framework.

2. Model First Approach :-



* In this approach, we can directly create Entities diagrams on the designer of *.Edmx file (empty model)
* In this approach, entity model preparation is manual implementation.
* Based on the preparation of entities in *.edmx file, we can generate
  - ↳ DataBase Tables
  - ↳ Code for entity classes, DataContext class.

3. Code First Approach :-



* Code First approach was introduced in Entity Framework 4.1 version.
* In code-first approach:
  - ↳ First we have to create code for entity classes based on our requirement.
  - ↳ Next we will generate required entity model details and database.
* Entity Framework will create the data base at the time of execution based on your entity classes and configuration.

Note :-
→ In this approach no need to add "*.edmx" file.

* Code-First approach is also called "Domain Driven Design".

Example: Create MVC application to perform database operations on student data - By using code first approach of Entity framework"

Step 1: /Models/Student.cs.

Student.cs:

```
namespace MVCapplication62.Models
{
  public class Student
  {
    public int Id{get;set;}
    public string Sname{get;set;}
    public string Course{get;set;}
    public int Duration{get;set;}
  }
}
```

Steps-2
/Models/TestDbEntities.cs

TestDbEntities.cs:

```
using System.Data.Entity;
namespace MVCApplication62.Models
{
  public class TestDbEntities:DbContext
  {
    public DbSet<Student> Students{get;set;}
  }
}
```

Step-3 : web.Config, Add connectionstring

```
<connectionstrings>
<add name="TestDbEntities"
     connectionstring="Server=Narasimha; Database=TestDb; Integrated
     Security=true;" providerName="System.Data.SqlClient" />
</connectionstrings>
```

Step 4 : HomeController.cs :

```
using MVCApplication62.Models;
namespace MVCApplication62.Controllers
{
public class HomeController : Controller
{
  TestDbEntities db = new TestDbEntities();
  public ActionResult Index()
  {
   List<Student> stlist = db.Students.ToList();
   return View(stlist);
  }
public ActionResult Create()
  {
   return View();
  }
[HttpPost]
  public ActionResult Creati(Student obj)
  {
      db.Students.Add(obj);
      db.SaveChanges();
}}} return RedirectToAction("Index");
```

Step-5: Add views by using scaffolding templates.

Step-6: Execute the application.

Note:

→ If database and corresponding tables are not exists, Entity framework will create required database and tables.

→ If exists, It reuses existing database and tables.

16/11/15

DataAnnotations and Validations

1) What is dataannotations?

DataAnnotations are a set of attribute classes.

→ DataAnnotation attributes are used to perform validations on Model class (Entity class) in mvc applications

→ In ASP.NET MVC, validations are applied on model class properties.

→ validations are implemented using validations attribute classes.

→ These classes are belongs to "DataAnnotations" namespace.

→ DataAnnotations also used to provide customization on model class properties.

Library Info:

Library: System.ComponentModel.DataAnnotations.dll.

Namespace: System.ComponentModel.DataAnnotations

Validation Attributes:

1. Required
2. StringLength
3. RegularExpression
4. Range
5. compare.

Additional Attributes:

1. Key
2. Display
3. ScaffoldColumn
4. DataType.

Example: create MVC Application to implement validation on customer data using DataAnnotations.

1. /Models/customer.cs :

```
using System.ComponentModel.DataAnnotations;
using System.Web.MVC; // only VS 2010.

namespace MVCApplication63.Models
{
 public class customer
 {
  [Required]
  [Display(Name="customer First Name")]
   public String FirstName{get; set;}.

  [Required(ErrorMessage="Last Name should not be empty")]
  [StringLength(10, minimumLength=3)]
   public string LastName{get; set;}

 [Required]
 [Range(20, 40)]
  public int Age{get; set;}

 [RegularExpression("\\d{6}")]
 public string pincode{get; set;}

 [DataType(DataType.EmailAddress)]
  public String Email{get; set;}

 [compare("Email")]
  public String ConfirmEmail{get; set;}
 }
}
```

2. HomeController.cs

```
using MVCApplication68.Models;

namespace MVCApplication68.Controllers
{
 public class HomeController : Controller
 {
  public ActionResult Create()
  {
   return View();
  }

[HttpPost]
  public ActionResult Create (Customer obj)
  {
   if(ModelState.IsValid =true)
    {


   Response.Write("Customer Details are registered"); // save to db logic
    }
   else
   {
   Response.Write("Customer data contains some invalid data. please provide
                   valid data");

   return View();

 }
}
```

3. Add view page for create action by using scaffolding template.

4. Update default action as "Create" in RouteConfig.cs,

8/11/15

Layout Views in MVC

① What is Layout view?

→ Layout view is a special type of view in ASP.NET MVC.

→ Layout views are similar to Master pages in ASP.NET WebApplications.

→ Layout views are used to maintain a common layout (structure) across all of the views in your application.

→ It act as a view template, so that we can create rest of the views with the same structure.

Advantages:

→ we can avoid duplication of code in view pages.

→ Application development will be faster and easier.

→ Modifications will be easier.

→ Reduce the time and efforts of programmer.

Files that are involved in views processing:

1. ~/views/_ViewStart.cshtml.
2. ~/views/Home/Index.cshtml
3. ~/views/shared/_Layout.cshtml.

Note: In the above files, second file will changes based on required action.

② What is the purpose of _ViewStart.cshtml?

→ _ViewStart.cshtml file will execute before processing of every view.

→ generally this file contains setting i.e layout name that is used to process the views.

```
@{
Layout = "~/views/shared/_Layout.cshtml";
}
```

Note: If you want to use different layout, you should update the above statement.

Important statements in Layout view:

1. @ viewBag.Title

   → It is in head section.

<title> @viewBag.Title </title>

   → the page-title will be inserted here. the title which you set in content views (Eg: Index.cshtml).

2. @RenderBody()

   → It is in body section

   → <body>

   @RenderBody()

   ...

   </body>

   → At the time of execution, razor engine will execute the corresponding view (Index.cshtml) and result will be placed at "@RenderBody()" location.

Note:

→ A project can have more than one layout file

→ we can't use "@RenderBody()" method morethan one time in the same layout.

Steps for creating Layouts views:

1. Add Layout view to "shared" folder.

   Name: _Layout2

   → Rt. click on "shared folder".

   → Add view

   → view Name: _Layout2.

   → check the "use a layout page" checkBox.

   → click "Add" button.

2. Add the following items in your layout file (_Layout2.cshtml)

1. @ViewBag.Title — Head
2. @RenderBody — Body.

3. Refer _Layout2.cshtml at the time of adding your regular views.

→ Rt. click on Action method.
→ Add View
→ Check "Use Layout page"
→ click browse button(---)
↳ select "_Layout2.cshtml".
↳ click "OK".
→ click "Add" button.

Note:

→ If you want to apply layout view for all views, you need update in _ViewStart.cshtml.

18/11/15

Filters in ASP.NET MVC

* ASP.NET mvc introduced an advanced concept called "filters".
* Filters are used to inject (adding) additional logic that will execute either before/after action method Execution.
* In MVC, filters are developed as attribute classes so that we can easily apply to action methods.
* It provides pre-action (before) and post-action (after) behaviour.
* Examples of predefined filters:
    1. [Authorize]
    2. [OutputCache]

What is custom filters?

→ A filter which is developed by programmer as per the requirement is called as "customfilter".

→ Every filter is an attribute class that will be processed by mvc engine.

→ We should follow the corresponding rules at the time of developing filters attribute classes.

Types of Filters in MVC:

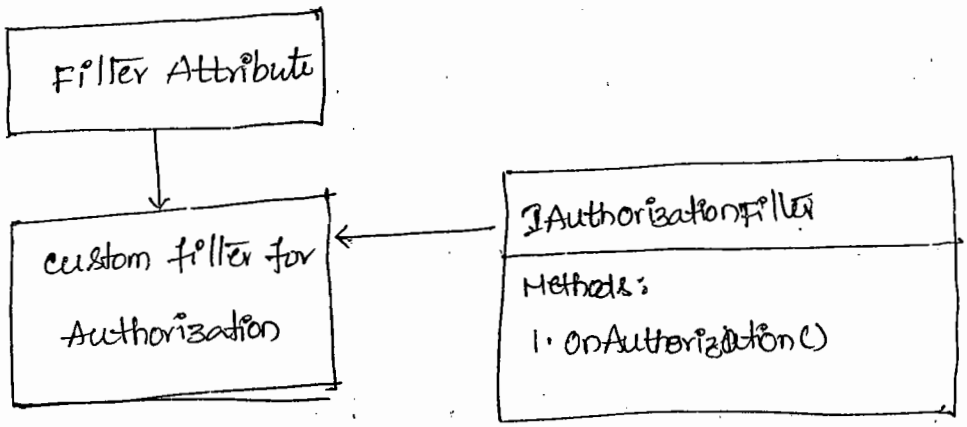
1. Authorization Filter
2. Action Filter
3. Result Filter
4. Exception Filter.

How to develop custom Filters?

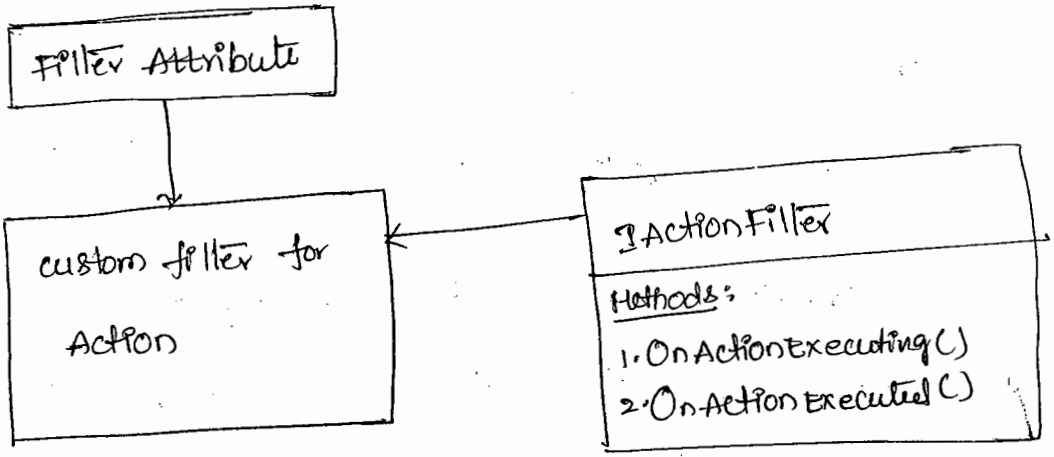
→ If you want to develop custom filters, you need to follow these rules:

1. It should be inherit from "FilterAttribute" class.
2. It should be implement corresponding interface based on the type of filter.

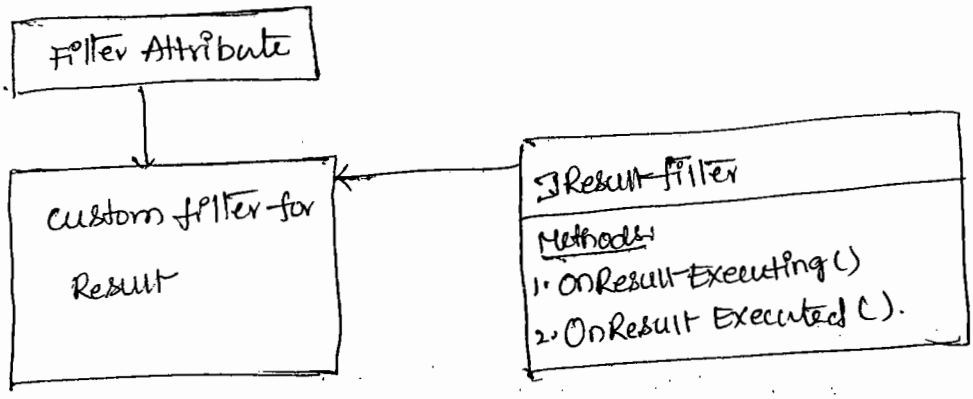
1. Authorization Filter :-



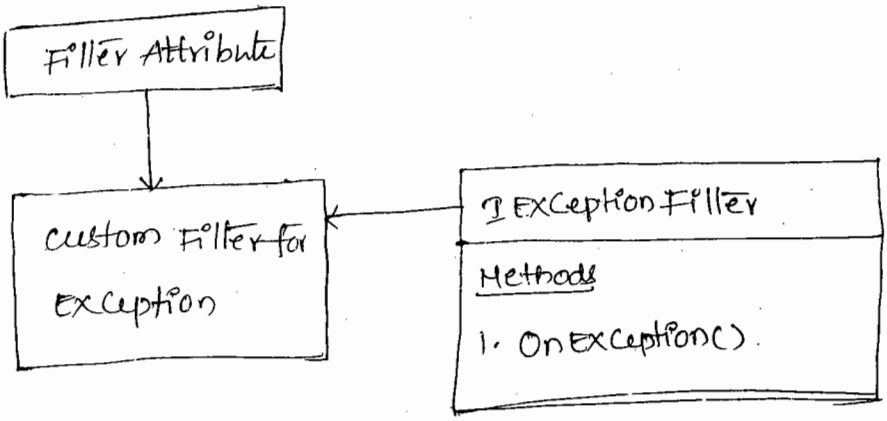
2. Action Filter :



3. Result Filter :



4. Exception Filter:



Method Arguments:

methods	method Arguments (these all are classes)
1. OnAuthorization	AuthorizationContext
2. OnActionExecuting	ActionExecutingContext
3. OnActionExecuted	ActionExecutedContext
4. OnResultExecuting	ResultExecutingContext
5. OnResultExecuted	ResultExecutedContext
6. OnException	ExceptionContext.

Example 1: Create MVC Application to implement custom filter for record log details of action method Execution.

steps: Create MVC Application and Add "Filters" Folder in solution Explorer.

2. Add class ~~to filters folder~~ file to "Filters" folder.

   Filename: MyCustomFilters.cs.

Define the required methods in the MyCustomFilters class as follows:

MyCustomFilter.cs:

```
using System.web.mvc;
using System.IO;
namespace MVCApplication 66.Filters
{
```

```
public class MyCustomFilter : FilterAttribute, IActionFilter.
{
  public void onActionExecuting(ActionExecutingContext obj)
  {
  StreamWriter sw = New StreamWriter ("D:\\CustomLogs.Text", true).
     sw.WriteLine("Actionmethod execution starts at :" + DateTime.Now.ToString());
        sw.close();
   }
public void onActionExecuted (ActionExecutedContext obj)
  {
 StreamWriter sw = New Streamwriter ("D\\ CustomLogs.txt", true);
 sw.writeLine("Action method execution completed at:" + DateTime.Now.ToString());
     sw.WriteLine ("_ _ _ _ _ _ _");
        sw.close();
     }
  }
}
```

HomeController.cs :-

```
using MVCApplication66.Fillers;
Namespace MVCApplication66.Controllers
{
  public class HomeController : Controller
  {
   [MyCustomFilter]
    public ActionResult Index()
   {
     return View();
    }
  }
}
```

4. Add view page for Index() action.

5. Execute the application.

Note :-

→ At the time of execution try to refresh the page and verify the CustomLogs.text file to know the status.

19/11/15

Security in ASP.NET MVC Application :-

→ ASP.NET MVC applications also provides different security options like regular ASP.NET webapplications

→ similar to ASP.NET, we need to configure required settings in web.config file

1. <Authentication>
2. <Authorization>

→ these settings should be provided in "web.config" which is in root folder.

→ 1. <authentication> tag is used to configure which type of security that you are using in your Application.

2. <authorization> tag is used to allow or deny the users.

Note: Based on these two tags, ASP.net mvc Framework will perform security.

Steps to implement security:

1. prepare only <authentication/> tag in web.config and provide required details
2. Create Account controllers and ~~action method for your application~~ with Login Action Method and corresponding views.
3. Create required controllers and action methods for your application.
4. Apply Authorization for verification:

   → in MVC security is applicable at three levels.

   1. Application level:

   ```
   <authorization>
     <deny users="?" />
   </authorization>
   ```

   2. Controller Level:

      Use [Authorize] before controller class name.

   3. Action Method Level:

      use [Authorize] before action method.

Note:

→ [Authorize] attribute is a pre-defined filter, It is comes under authorization filters type.

Example:

* Create ASP.NET MVC Application to implement security.

1. Create ASP.NET MVC Application and provide the authentication tag in web.config.

Note: open web.config that is in the root folder.

```
<authentication Mode="Forms">
    <forms Name="ASPX_AUTH"
           loginUrl="~/Account/Login"
           defaultUrl="/"
           timeout="30" />
</authentication>
```

2. AccountController.cs:

```
using System.Web.Security;
namespace MVCApplication67.Controllers
{
  public class AccountController : Controller
  {
    public ActionResult Login()
    {
      return View();
    }
  [HttpPost]
  public ActionResult Login (string uid, string pwd)
  {
    if(uid=="admin" && pwd=="admin")
    {
FormsAuthentication.RedirectFromLoginPage(uid, false);
```

```
    else
    {
    ViewBag.ErrorMessage = "Invalid user id or password";
    }
    return View();
    }
}
```

3. Login.cshtml :

```
<h2> Login page </h2>
<br/>
@using (Begin Html.BeginForm())
{
    @Html.Label("UserName")
    @Html.TextBox("uid")
<br/> <br/>
    @Html.Label("password:")
    @Html.password("pwd")
<br/> <br/>
<input type="submit" value="Login" id="sb1" name="sb1" />
}
<br/> <br/>
@{
    if(ispost == true)
    {
    string str = viewBag.Errormessage;
    <span style="color:Red;">@str </span>
    }
}
```

4. Add required controllers and Action methods to verify seeing

Eg: HomeController
- Index()
- products()
- categories()

TestController
- Index()

→ prepare required views for the above action methods.

5. Apply the authorization option at project level, controller level and action method level.

20/11/15

AJAX programming in ASP.NET MVC using jQuery

* What is AJAX?

* AJAX stands for Asynchronous JavaScript And xml.

* AJAX is a client side programming technique to communicate with server asynchronously (implicitly).

* By using AJAX, we can implement partial page updates.

Advantages:

1. partial page updates (updates required portion of the page).
2. Reduces network traffic.
3. Reduces burden on the server.
4. Quick response from server.

How to implement AJAX programming?

→ we can implement Ajax by using javascript.

→ If you use only javascript code, it will complex and time consuming

→ so that we can use any javascript libraries like JQuery, AngularJS, etc--

→ These libraries makes your client side programming will be more easy.

What is JQuery?

→ JQuery is a javascript library.

→ JQuery provides collection of pre-defined methods to implement any kind of javascript related activities (client side) easily.

Eg: ajax()
val() ← value from the
text() → get values from other than text box

Note: Access the content of html elements in JQuery:

→ $("Id")
→ $("#t1").val()
→ $("sp1").html()
→ $("sp1").text()

21/11/15

~~Routing in ASP.~~

Example: create ~~ASP.~~NET MVC Application to get the Employee count based on the deptno ~~from~~ server with AJAX communication using JQuery

Hint: By default all mvc applications contains jquery file in script folder. It is imported in _Layout file.

1. Create new ~~ASP.~~NET MVC Application.
2. Add "Emp" table to Models folder by using EF. Build the project
3. Home Controller.cs

```
using MVCApplication70.Models;
namespace MVCApplication70.Controllers
{
public class HomeController : Controller
{
 public ActionResult Index()
{
return View();
}
SathyaDbEntities db = new SathyaDbEntities();
public ActionResult GetEmpCount(int Id)
{
 var q1 = from e1 in db.Emps.ToList()
          where e1.Deptno == id
          select e1;
string str = "Employee Count:" + q1.Count();
 return Json(str, JsonRequestBehaviour.AllowGet);
}
}
```

4. Index.cshtml :-

```
<h2> Ajax programming using jQuery </h2>
<hr/>
Enter deptno : @Html.TextBox("t1")
<br/> <br/>
<button id="b1" onclick="f1()"> Get Emp count </button>
<br/> <br/>
<span id="sp1"> </span>
```

5. prepare required javascript functions in myscripts.js.

Steps to Add javascript file:

→ Goto solution explorer
→ Rt. click on "scripts" folder
→ Add → New Item
→ select "Java script file"
→ name : Myscripts.js
→ click "Add" button.

Myscripts.js :

```
function f1()
{
  var dno = $("#t1").val();
  $.ajax(
  {
    url : "/Home/GetEmpCount/" + dno,
    type : "GET",
    data : "",
    contentType : "application/JSON;",
    success : function(response)
  }
```

```
    {
        $("#Sp1").text(response);
    }
});
}
```

6, → Import my Scripts.JS file in _layout.cshtml file.

→ Drag "MyScripts.JS" file from Scripts folder and drop it in layout file before closing </body> tag.

Routing in ASP.NET MVC.

What is Routing?

Routing is a concept of mapping incoming request with server resources that should process the request.

→ In ASP.NET MVC, routing will mapping each request with specific controller and action methods.

→ Routing plays an important role in an Asp.net MVC application Execution flow

How routing works in ASP.NET MVC?

→ MVC framework uses a "RouteHandler" to perform routing.

→ RouteHandler uses "RouteTable" to identify the routes.

→ Default route and other route details are available in "RouteConfig.cs" file.

Note:

→ Route registration required statements will be executed in Global.asax file (Application_Start).

What are the advantage of routing?

→ we can achieve clear/ clean URLs feature of mvc

→ Search Engine Optimization (SEO) will be easy.

→ User friendly URLs.

What is customizing Routes? (CR)

How to add extra routes?

→ providing user friendly route URLs is called "customizing Routes".

→ customizing Routes can be implemented in two ways:

1. Using RouteConfig.cs
2. Attribute Routing.

1. By using RouteConfig.cs :

```
routes.MapRoute(
        name: "Route1",
        url: "Admin",
        defaults: New { controller = "StoreManager" action = "Index" / Login" } );
```

Note: All custom routes should be placed before default route.

2. Attribute Routing :-

What is attribute routing?

→ Attribute Routing is a concept providing route information by using [Route] attribute.

→ It is introduced in ASP.NET MVC 5 version.

→ Attribute Routing is configured in controller class file itself.

→ Regular routing is configured in "RouteConfig.cs" file.

Implementation :

Step 1 : Enable Attribute routing in RouteConfig.cs file

```
routes.MapMvcApplicationAttributeRoutes();
```

Step 2 :

```
public class StoremanagerController : Controller
{
 [Route ("Admin")]
 public ActionResult Login()
 {
   return View();
 }
}
```

23/11/15

Web API

What is WebAPI?

* WebAPI is new frame work to develop services based on application.
* webAPI services are called "HTTP" services.
* WebAPI works as device independent i.e it can access from all devices like desktop, laptop, tablets, mobile phones.ect
* webAPI uses only HTTP protocol based methods for communication

Why microsoft introduced webAPI even-though it is having WCF/ web services?

* WCF/ web services requires additional protocol called "SOAP".
* SOAP stands simple Object Access protocol.
* SOAP is a XML based protocol. Due to this additional process required serialisation and de-serialisation.
* WCF introduced another services called "RESTful" services that uses only HTTP.
* but it is complex to configure WCF services.
* In order to address these problems Microsoft introduced ASP.NET WebAPI in 2012.

How WebAPI is related to ASP.NET MVC?

* WebAPI concept is developed based on ASP.NET MVC framework.
* webAPI uses Models, Controllers, Action Methods, Attributes, DataAnnotations, Filters, etc.....

Advantages of web API:

1. It does not required any special protocols like SOAP, TCP, etc,
2. It does not required any configuration details, we can simply acces by using URL.
3. Multiple Http clients can communicate with webAPI.

   Eg: Desktop, Mobile devices, Tablets, Browsers, etc.

WebAPI Methods for CRUD Operations:-

CRUD operation	Request Type	URL	Action Method Attribute
ReadSingle	GET	/api/EmpApi/1025	[Http Get]
ReadAll	GET	/api/EmpApi	[Http Get]
Create	POST	/api/EmpApi	[Http Post]
Update	PUT	/api/EmpApi	[Http Put]
Delete	DELETE	/api/EmpApi/1025	[Http Delete]

Developing WebAPI Service:

→ Developing of web API is similar to regular MVC controller.

→ Every WebAPI controller class should be inherit from "ApiController" class.

→ "ApiController" belongs to system.web.Http

Note: In web API request processing is based on "Request Type" instead of the action method name.

Developing client Application :-

* client application will communicate with webApp service by using "HttpClient" class.
* Client Application is contains only URL of service, rest of the details will be executed on Service.

Library Information :-

Library Information : System.Net.Http.dll

Namespace : System.Net.Http.

class : HttpClient

Http Client class Methods :-

1. GetAsync ("URL")
2. PostAsJsonAsync <T> ("URL", obj)
3. PutAsJsonAsync <T> ("URL", obj)
4. DeleteAsync ("URL").

Example1 :

Create web API service to provide Employee details by using ASP.NET MVC web API concept.

1. Create New ASP.NET web Application. ↳ Name: WebAPI ~~project~~ Service Application.
   - ↳ select "Empty" template
   - ↳ check 'mvc' and 'Web'.
   - ↳ click 'Add' button.
2. Add Emptable to Models by using Entity Framework. build the project.
3. Add ~~API~~ webApi controller to your project
   - ↳ Rt. click on controller folder.
   - ↳ Add ↳ controller
   - ↳ select "WebAPI2 - Empty controller".
   - ↳ click "Add" button.
   - ↳ Name: EmpApiController.
4. Define the required methods in EmpApiControllers.cs.

```
using webApiServiceApplication.Models;
Namespace WebApiServiceApplication.Controllers
{
 public class EmpApiController : ApiController
 {
 SathyaDbEntities db = New SathyaDbEntities();
 [HttpGet]
 public Emp GetEmp(int id)
 {
 Emp obj = db.Emps.Find(id);
 return obj;
 }
```

```
[HttpGet]
public List<Emp> GetEmps()
{
List<Emp> emplist = db.Emps.ToList();
return emplist;
}

[HttpPost]
public String AddEmp(Emp obj)
{
    db.Emps.Add(obj);
    db.SaveChanges();
    return "Emp Details are added to db";
}
}
```

5. Add Home Controller to your project

- ↳ Add → controller
- ↳ Select "MVC 5 controller Empty"
- ↳ Click 'Add' button
- ↳ Name: Home Controller.

6. Add view for Index and provide the content as follows:

Index.cshtml:

```
<h2> Web API Service is Running </h2>
```

7. Execute the project so that client project will use service.

———x———

Create another MVC-Application as client to consume the above.

webApiService.

1. Create New ASP.NET MVC project (same as above project)

Name: WebApiClientApplication.

2. Add Empclass to Models folder

Emp.CS

```
namespace webApiclient-Application.Models
{
 public class Emp
 {
  public int empno {get; set;}
  public string Ename {get; set;}
  public string Job {get; set;}
  public int Sal {get; set;}
  public int Deptno {get; set;}
 }
}
```

→ Build the project.

3. Add HomeController and define the Action methods as follows:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Net.Http;
using System.Net.Http.Formatting;
using WebApiClientApplication.Models;

namespace WebApiClientApplication.Controllers
{
  public class HomeController : Controller
  {
    HttpClient client = new HttpClient();

    public ActionResult Index()
    {
      string url = "http://localhost:1577/api/EmpApi";

       HttpResponseMessage resObj = client.GetAsync(url).Result;

      List<Emp> empList = resObj.Content.ReadAsAsync<List<Emp>>().Result;

      return View(empList);
    }

    public ActionResult Details(int id)
    {
      string url = "http://localhost:1577/api/EmpApi/" + id;
      HttpResponseMessage resObj = client.GetAsync(url).Result;
```

```
            Emp obj  = resObj.Content.ReadAsAsync<Emp>().Result;
            return View(obj);
        }

        public ActionResult Create()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Create(Emp  obj)
        {
            string url = "http://localhost:1577/api/EmpApi";
            HttpResponseMessage resObj = client.PostAsJsonAsync<Emp>(url, obj).Result;
            string  str  = resObj.Content.ReadAsAsync<string>().Result;
            return RedirectToAction("Index");
        }
    }
}
```

4. Add views for above action methods by using scaffolding templates.

5. Execute and verify.

24/11/17

Bootstrap in MVC 5.2

What is Bootstrap?

→ Bootstrap is client side framework for web applications.

→ Bootstrap will provide all pre-defined css classes that we can apply for html elements.

→ In MVC 5.2, bootstrap files will comes along mvc project template.

→ By default it is included in layout file

```
<link href="~/content/bootstrap.min.css" rel="stylesheet" />
```

How to apply bootstrap css classes to HTML tags?

→ we can add bootstrap styles to html tags by providing corresponding css class name.

→ Bellow are the examples of bootstrap css classes that we can apply buttons:

1. btn btn-default
2. btn btn-primary
3. btn btn-success
4. btn btn-info
5. btn btn-danger
6. btn btn-warning.

Usage:

```
<input type="button" class="btn btn-primary" value="Login" />
```

http://getbootstrap.com/ — Download Bootstrap file for VS2012

(1) What is minification?

→ Minification is a concept of reduce the file size by removing unnecessary new lines, whith space and comments.

→ It also shortening variablenames.

→ minification performances code optimizations to *.js, & *.css.

Eg: bootstrap.min.css
jquery-1.11.3.min.js.

Advantages: Due to this browser can easily download the files from server and faster in execution.

(2) what is Bundling?

→ It makes easy to combine or bundle multiple files into a single bundle.

→ you can apply this concept on css, javascript files.

→ It will improve the performance of your webpage.

→ In single communication, browser will get multiple files.

(3) What is viewmodel? (VM)

```
public class StudentFacultyVM
{
    public List<Student> StList { get; set; }
    public List<Faculty> FcList { get; set; }
}
```

View

@model IStudentFacultyVM

model.stlist — use foreach and generate separate table

model.flist — use foreach and generate separate

---

(3) What is viewmodel? Why do we use this concept?

(or)

How to transfer more than one model class object/objects from controller to view?

→ ViewModel(VM) is a concept of creating separate class in order to sending multiple model class objects to view

→ It is a special class designed to transfer multiple model class objects from controller to view.

→ This class will be prepared in "Models" Folder.

~~public class IStudentFaculty~~

(4) How to use outputcache attribute in ASP.NET MVC?

```
class HomeController : Controller
{
[OutputCache(Duration = 20, VaryByParam = "none")]
public ActionResult Index()
{
    ViewBag.Message = DateTime.Now.ToString();
    return View();
}
}
```

(5) What is partial view in MVC?

→ partial view is used to prepare small portion of view content which can be reusable in other views.

→ It is like a web user control in Asp.net.

How to Create partial View?

1. Rt. click on "shared" folder
   ↳ select "Add view".

2. provide view Name
   Eg: _Header

3. Check "Create as partial view" Checkbox

4. Click "Add" button.

Note: use the bellow statement to accessing partial view from other views.

```
@{ Html.RenderPartial("_Header"); }
```