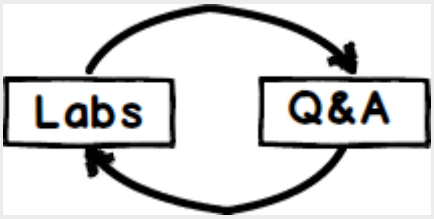
Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Learn MVC Project in 7 Days – Day 1

Introduction

As the title promises "Learn MVC Project in 7 days", so this article will have 7 articles i.e. 1 article for each day. So start reading this tutorial series with a nice Monday and become a MVC guy till the end of the week.

Day 1 is kind of a warm up. In this first day we will understand Why MVC over WebForms? , Issues with WebForms and we will do two Lab's one around controller and the around views.



After each one of these lab's we will run through a small Q and A session where we will discuss concepts of the lab. So the structure of this is article is Lab's and then Q and A.

In case for any Lab you have questions which are not answered in the Q and A please feel free to put the same in the comment box below. We will definitely answer them and if we find those question's recurring we will include the same in the article as well so that rest of the community benefit from the same.

So we just need your 7 day's and rest this article assures you become an ASP.NET MVC developer.

What do we need for doing MVC?

We just need Visual Studio and the good news is that visual studio is completely free. You can download

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Visual studio community edition from http://www.visualstudio.com/ , no expiry , no license issues and you do not need deep pockets.

ASP.NET vs MVC vs WebForms

"You are reading this article because you know ASP.NET and you want to upgrade yourself to MVC."

Sorry for the trouble, can you please read the above statement again and if you think its right then this section is a must read for you.

Lot of ASP.NET developers who start MVC for the first time think that MVC is different new, fresh from ASP.NET. But the truth is ASP.NET is a framework for creating web application while MVC is a great architecture to organize and arrange our code in a better way. So rather than MVC you can say ASP.NET MVC.

Ok so if the new thing is ASP.NET MVC what is the old thing called as, it's "ASP.NET WebForms".

Let me correct your vocabulary:-

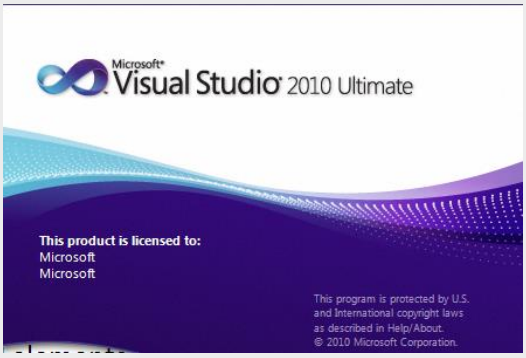
"You are reading this article because you know ASP.NET WebForms and you want to upgrade yourself to ASP.NET MVC."

So now that your vocabulary is corrected welcome to the world of ASP.NET MVC and let's start this tutorial.

Why ASP.NET Web Forms?

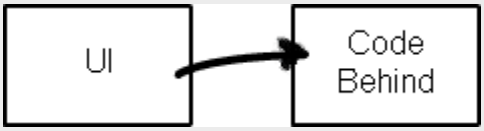
ASP.NET WebForms has served and successfully delivered web application for past 12 years. Let us try to understand the secret of what made WebForms so popular and successful.

If you see the success of Microsoft programming languages right from the days of VB (visual basic) it is due to RAD (Rapid application development) and visual programming approach. Visual programming was so much preached and successful in Microsoft that literally they named their IDE as "Visual studio".



Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

By using visual studio, developers where able to drag drop UI elements on a designer area and at the backend, visual studio generates C# or VB.NET code for those elements. These codes where termed as "Behind Code" or "Code Behind". In this code behind Developers can go and write logic to manipulate the UI elements.



So the visual RAD architecture of Microsoft has two things one is the UI and the other is the code behind. So for ASP.NET Web forms you have ASPX and ASPX.CS, for WPF you have XAML / XAML.CS and so on.

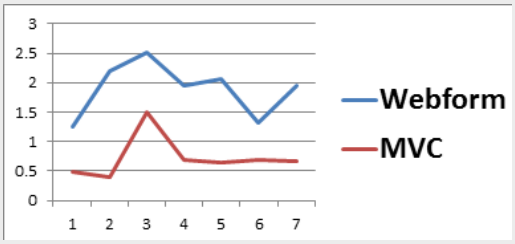
Problems with Asp.Net Web Forms

So when ASP.NET WebForms was so successful, why Microsoft thought of creating Asp.Net MVC. The main problem with ASP.NET WebForm is performance, performance and performance. In web application there are two aspects which define performance:-

1. Response time: - How fast the server responds to request?
2. Bandwidth consumption: - How much data is sent?

Response time issues

Let us try to understand why response time is slower when it comes to ASP.NET WebForms. We did a small load testing experiment of WebForm vs MVC and we found MVC to be twice faster.



Let us try to understand why ASP.NET MVC was better in performance in the above load test. Consider the below simple UI code and Code behind for that UI.

Assume the ASPX code has the below simple text box.

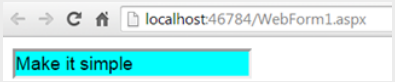
```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

In the code behind you have written some logic which manipulates the text box values and the back ground color.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
protected void Page_Load(object sender, EventArgs e)
{
        TextBox1.Text = "Make it simple";
        TextBox1.BackColor = Color.Aqua;
}
```

When you run the above program below is the HTML output.

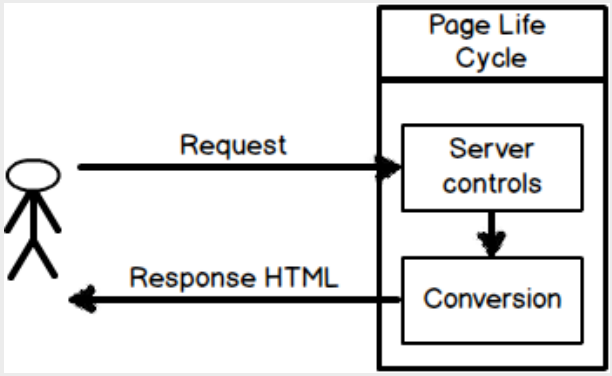


If you see the HTML output by doing view source it looks something as shown below.

```
<input name="TextBox1" type="text" value="Make it simple" id="TextBox1" style="background-color:Aqua;" />
```

Now stop reading for a moment, close your eyes and think. Try to get answers to the below questions:-

1. Is this an efficient way of generating HTML? Do we really need to make those long server trips to get those simple HTML on the browser?
2. Can't the developer write HTML straight forward, is it so tough?



If you see for every request there is a conversion logic which runs and converts the server controls to HTML output. This conversion gets worse and heavy when we have grids, tree view controls etc. where the HTML outputs are complicated HTML tables. Due to this unnecessary conversion the response time is less.

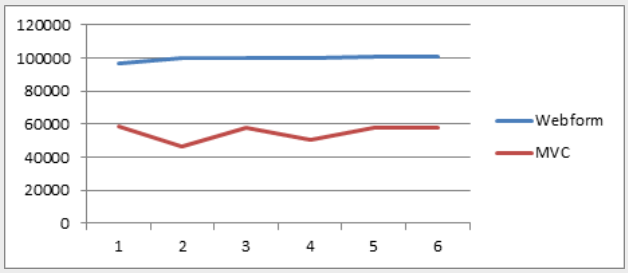
Solution for this problem: - "GET RID of CODE BEHIND", fold your sleeves and work with pure HTML.

Bandwidth consumption

ViewState has been a very dear and near friend of ASP.NET developers for past 10 years because it automatically saves states between post backs and reduces our development time. But this reduction in development time comes at a huge cost, ViewState increases the page size considerably. In this load test we found ViewState increases the page size twice as compared to simple ASP.NET WebForms.

Below is the plot of the content length emitted from WebForm and MVC.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



The size increase is because of extra bytes generated from ViewState, below is the snapshot of a ViewState. Lot of people can argue that ViewState can be disabled but then we all know how developers are, if there is option given they would definitely try that out.

```
<body>
    <form method="post" action="WebForm1.aspx" id="form1">
<div class="aspNetHidden">
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="WFRayhqG9cEK735FA/bNsy7ds1dONoQBBnjmU1l+1twCvwpME60QLVrlh/RxiOEopzEz6XyqChdzcjSBNF9LRFGzZd+SAH3XzF3vELxp9IHnDkRZATU8Na3y0WaXgPDuBvRLu6lxWkXkQ6qBi7f7
Gkx9L0DGOvzpU35clPDBlXgCUgOra7NbdWL3MT8QsLW3Zf1EehAXet0YUTAeBuRNRiKoactcnVdV8cC8UA0Yjjn0zo9oMwNu11xJ/G3E0Wan/BTXsM4jrBzLoQOMCv7yQQsshIYD1KGpG/Ql7LesvbNcPe2
cmQDg+nFXXCK7LLXSWzIzbx+HwUyekqHBGCzXOEUQ66XZRSSUiqnpOILA8BmCzE/66mLQiYPGUpQCfSNDCHV8WLM5C1YW7Su9ywlJ01HZJIQ5gE1tmxOO4CmoDlPvq+H4iEId2OTCfJxH84zR9aX1zqUnw4
7Kqz0pDiPHVB7GNH+p4kSGqCEWjDg8z1sOr3Q1d9qJUXwpUxGNaVcIeSut+P1OiGyqidGtHTMRnDrbd5bWmJeo8myTnDbHCO6zO8/3yJDkYW9zBEpVHB9ByZmOldMzw1YPwlWThZVrd+EArWfxymk3w1xN7
rSYMWnavPLD2nPiWy9bW63PjnHUVkN1JDdfHjSySRiAGDu4yf8HcaHWMcdtrSCJIKZGxFznfmoHPElf5z3bh0Rpd6yOGKJN+uQ0y3Y0trB4RRPYSash80xpgML4Axf7cKSymLNIzqnrsxD4vHQ2pZUT1WyF
QGuuM7ticxhd046E6fv3QLSaIKKcGF3HZiZNyWa5VKoDn+ssQ/p6igLyaeQfJ4WjrcvFNXP6y5XntucPBrRWEINbHgrhpLnRwGVMn7+/6jse8XW5oRRevHHGS6my2ojYNkXF/sg6vXthChcZ7QB7gvZl/0v
P2h3o09O6Ra3LURdrsEgk579WmCYkM0+svQ00EKA/EmWwTa+1lAc+MuwXwrty3NTJYzSQFbia8efqJ4K7JXz7D5z5XKSZCMudNKps5fvJ11CRTqG5ktXYCKHukMnHA7EpTXOpkWp4S2ADEoZwV73jDXQ+fu
hAOyCGoVzrMh90bDenBXgq9wN57T7JTg8Wng7QqPbh8TVAtwwHWHY/7aDKrvqGrKG9DgbPZvc4wIaaC75ZnXfL8w2Bq1eaZJCA3CIu+3xpfErs1zC6GRYlHbSsDfanXc0eoTClgl0fuAtazR5bGK79KhK8t
lKDMmYYOAIhua69G6fkO4VkMGV2tq/Umecm8pcDeeC4qMsS6t8nGG5UWPsHtRPNV1rtaMvc0Yc8Peln10ovi3gdI0OoPn/wb9t3OSFWif5lAXNg9ESWUa/IodAjF+/nD1ZguLVx46pUcR9d0Vh2GBXIbor
mwRKuKlWwri9ZaGWd8m3sojW+sEIcsjfucxQU88o6uY5hC48KeIdjUsdCnhS7CPc9tl+bO7UOecyBiFXIYfk1Gvyxxo9T35FjOfrjtWafzZxfGkZqsdT11QmbsnCqU+uYtDWGGZTP9Qcs/8s77XjoYjc0Bs
yHq8ArPWF9oGxXJD0uKUaKqW+8Ks9vHD0QazQMB0s28HWX+2VBNZZPjE5V95ChkQ0hmIZKKJ+YHnXPKFus29DSIRniLex8d5/B/zbfw82TwbAkRyRmu3uNV67n2NuhDf8/VLSbe+Z9oq4zb82iV3DjGiF4c
4g4Xa/OlVFDWxk1dBvcBIlpBkdxCxsHFjYX35Y0Lrfs0o2LAuyU0jy+B+kHvihwbs1gHXckXJAabBYTdYip+63A/9etzhxmEujR/dh5kUeg4ta7X+q6KDuQ8kSgpDWIW6Y9AZV25FP9/Z3QOd5gAsqLDSmj
36I1c4QPgTZlztya+QZmBc6NMDhnMEMrJ55z2mJsLLvDZOSrjJef/aAYCr6Gc5UMRf93zoara3PQ2OoC4NM/zfwCjTzO3NaD7Hr5vLCOnc4mlCnByanRJPg8Xtq+4NakX17VRYwl54EZW94ij7L37Al4OrCl
jKZnRLFldJQJd8DkAp68iqZt9IIasw20qcXdOe6XtFcTq5EFIJEHRUXVspljxkfoto7JD/KnjloiNUSQSZ0GZdTckKBePVe2MelgtR4cxk7AF1yyH2f5srFSU7lI0VgqOPEIdciw68cITQ1F2TK5Y94Dq0M
EXo8NVhGoZJDBtmgdUSREtHnmLvDV+YtbJZ7sBxryO0Q+A5JPb6/PMqXaunrveAIJxUPk1oX1iDql/+Iy512Blk+0uoj+X2sAHt0Y7xWNKsPD8wAmwSEHf1RzbmHFGJcatuJ14G0l8eOe+j/LJt9/zgXMif
tLx0+dtbu7lkj7OoTKF4Z9AX/GHdTK6XD9sXcTwuXxSroqD7PBTnIVBj69xydx+eHXLYuKTTy0Zq56V55OFBH23Q1qoHqTKLH6gRzj+anuJjsMgZ6CyhQ92i3LYB157XfO9lbM7TgdGr5wqePz9Nyq9CquU
I6WaN4ZoNUn54xun3FsNX/pj4ej6IpHIykOJPqTsltuloUwhs57/qMAUU88Xqu2Fzr+RLn59E2GP1DDc4dNLdhtsYc+q0koNP4Zbknz8x5Zo6zFcKRlMsRsk9OwGIDg0gMebrTtRaL126bu7jGnKFIg7qp8
CMHpigjs1xe8So/c2NWVcfc7X4RQ7PiPks0ty+Om/q4CbLHDgLXtlNkTISHCOThU4ovvWHLjmnsE3M/vGWRDBzPgNr6b2VIQ/iCzsp+BPOdg2S7jp7HR1H0ssscZRLOhElS7Psu9Y4eL9XrOIhmjhaoj2Po
KIS/MOCobyBf3dxsiH7xnqDO9v0jcS3EEO9NgWkUG8JlChIP7aOL7cb/sSn/7T+v2wahpdgaM1e0pp/bf4P/PF6S+mCZbdfCW6FvQEcWLFxTnRenOHzRQX51iINtkOGWf5TNt7eHGxdNT1t8rWk/bjMrsIL
VHNdIeyx1pt2opmefMrJSRltMKECgkI6bk0L3u7jQZqN0Qic4jEYZ6ml2DYyCPUcH1BaYaGqBHQ8c28G/R4EM+BcwcQmcSjB6dQzqnDSd0+t3kURncI5uOR2Vg7Uh1ES/WUPzhWFKOBuWV/+3gVCXbRPwz8
tuWYhTDoyyunvuaBIUrF8t/VzRw6MyqJBhAKS63BAeuQ3Au7hyydRrcUWMZg+cqPwwaGyLOArVuFZ/Qq7UzheoWlbjGomDswMUv4gNAousfFD4Xs2F4yoxtdFcBIVxcUUDGSqwnUjZzSK/zJoNQKCiE+ODq
vGBbsn8SHs1L0wJ8HyVHTYfy1skXJd3uRzISGQpKottVMdkV7ID6FQ7nbZnjaM7VpNwAdFPJq2PaQRm2VVPA7nNab9zkeNHRvR55/zMEIdeBmmZeRRoVfpGZt91bdlwzfWAFrXFpaxjypNuecsdqI5kdZaa
/nxtNF+74CNPhIbPxS+EiycMJXwLBurIUKxRPNcdi9gDnHwqdXexFQ9ZKz5DDPC7Rwii1huos7FDnodp46/Hd/nlmEF+WiPQLa7tlgYBjuSnkgfBCnW7YPQot7ztMJ0MsxExZ4HQ0l+w4NaHB5RYP+Cd4gd
xmZf6s0C4A0qj6dU6jZ5s0mP6wM1lKNRK0M2n9vj5yefURcpVUEGBrfvXc+qZvQdBht6oG+fzv5pYRykADW19kGe4xtLOELEUkgGBQMBUm92N7meP9TwpMcYxStdVpBv9C8gJTzci4PdaduOU8Eh/dfl9Ag
xpiFWANclIvgJia77YHBsnTXuJeZTh0b/Jh5JzJk3fd7Q6x9JT/mY38fuCmrKCXGuacTCkZHsuQ5axCkAj+RaaT8hkU495U9R6tnxpyW4jRxtgdvDmwNP8oTx+WRk9qxhVkLaWNnvnEzjnJHY9fRP09pK1z
X68N8pRe82LMofeTxCcec75pvvDpWdOKLL0NDHZ+cF+0RsSLYV1vYJl3TOfFrvBgGg5VvGcG3oEsFLs2X9tEIlR1w13agpLs2ErXivXT305Zcdo8vFAJKd+hfR0T+7anV9pNCEJHTKBfdWScJNf0dOF/nA8i
```

Solution for this problem: - "GET RID of SERVER CONTROLS".

Note: - The rest of the three points down below are brownie issues which have cropped up due to presence of code behind and server controls. But the main thing is always performance.

HTML customization

Now because we are salves of the code behind and ASP.NET web server controls, we have "NO IDEA" what kind of HTML can come out and how efficient they are. For example see the below ASPX code, can you guess what kind of HTML it will generate.

```
<asp:Label ID="Label1" runat="server" Text="I am label">
<asp:Literal ID="Literal1" runat="server" Text="I am a literal">
<asp:Panel ID="Panel1" runat="server">I am a panel</asp:Panel>
```

Will Label generate DIV tag or SPAN tag? If you run the above code below are the respective generated HTML. Label generates a SPAN, Literal generates simple text, and Panel generates DIV tag and so on.

```
<span id="Label1">I am label</span>
I am a literal
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
<div id="Panel1">I am a panel</div>
```

So rather than generating HTML using server controls how about writing HTML directly and taking complete control of HTML.

So the solution for this problem is "DO NOT USE SERVER CONTROLS" and work with direct HTML.

The other great benefit of working directly with HTML is that your web designers can work very closely with the developer team. They can take the HTML code put in their favorite designer tool like Dreamweaver, front page etc. and design independently. If we have server controls these designer tools do not identify them easily.

Reusability of code behind class

If you watch any professional ASP.NET WebForm project you will notice that code behind class is where you have huge amount of code and the code is really complicated. Now this code behind page class inherits from "System.Web.UI.Page" class. This class is not a normal class which can be reused and instantiated anywhere. In other words you can never do something as shown below for a WebForm class:-

```
WebForm1 obj = new WebForm1();
obj.Button1_Click();
```

Because the "WebForm" class cannot instantiate without "request" and "response" object. If you have ever seen the "Button Click" events of "WebForm" they are as shown in the code below. From the code you can know how difficult it is to instantiate the same.

```
protected void Button1_Click(object sender, EventArgs e)
{
// The logic which you want to reuse and invoke
}
```

Solution for this problem: - "GET RID of SERVER CONTROLS and CODE BEHIND".

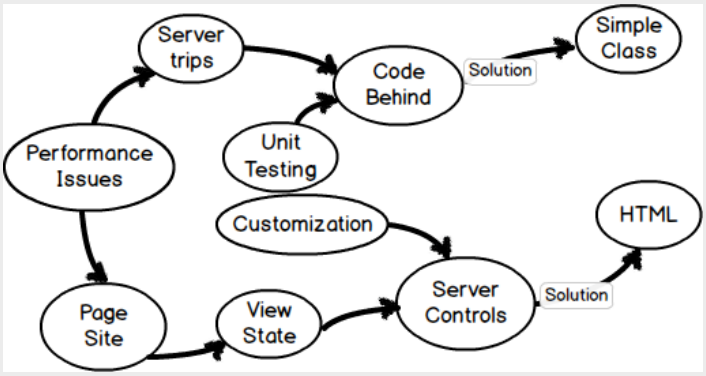
Unit Testing

As said in the previous section you cannot instantiate behind code straight forward it's very difficult to do unit testing or I will say automation testing on the code behind. Someone has to manually run the application and do the testing.

What's the solution?

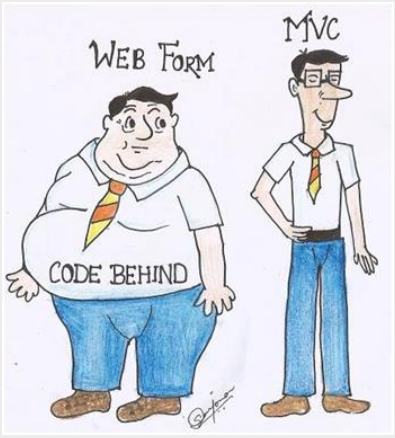
If we read the four issues mentioned in the previous section with ASP.NET WebForms the main culprit are two people "Code Behind" and "Server controls". Below is root cause diagram I have drawn. In this I started with problems, what is the cause for it and the solution for the same. The complete diagram zeroes on two things "Code Behind" and "Server controls".

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



The solution is we need to move the code behind to a separate simple class library and get rid of ASP.NET Server controls and write simple HTML.

In short the solution should look something as shown in the below image.

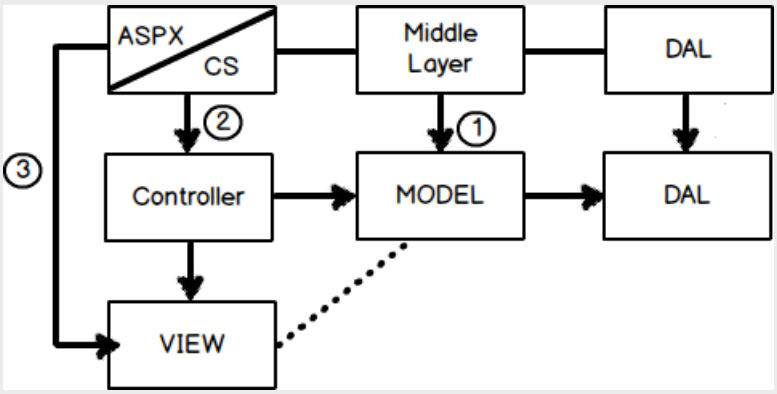


How Microsoft Asp.Net MVC tackles problems in Web Forms?

As said the code behind and server controls are the root cause problem. So if you look at the current WebForm architecture which developers are using its mostly 3 layer architecture? This three layer architecture comprises of UI which has ASPX and the CS code behind.

This UI talk's with .NET classes which you can term as middle layer, business logic etc. and the middle layer talks with data access layer.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



So MVC comprises of three sections Model, View and Controller. The code behind logic goes in to the controller. View is your ASPX i.e. pure HTML and your Model is your middle layer. You can see in the above diagram how those layers fit in.

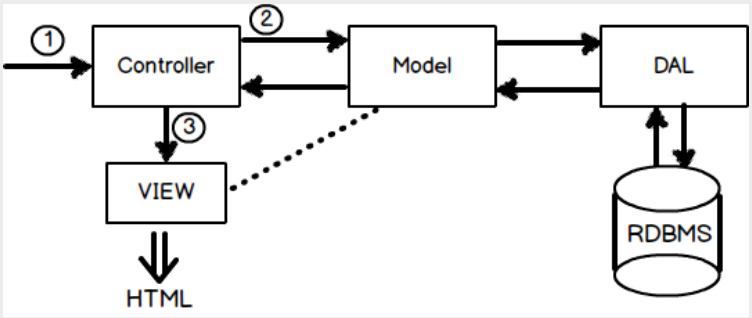
So if you see there are two major changes VIEW becoming simple HTML and code behind moving to simple .NET classes termed as controller.

ASP.NET MVC request flow in general moves as follows:-

Step 1:- The first hit comes to the controller.

Step 2:- Depending on the action controller creates the object of the model. Model in turn calls the data access layer which fetches data in the model.

Step 3:- This data filled model is then passed to the view for display purpose.



Now that we have understood the different components of MVC let's go in depth in to each one of these components, let us start doing some labs. Let us first start with controllers as they are the most important and central part of the MVC architecture.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

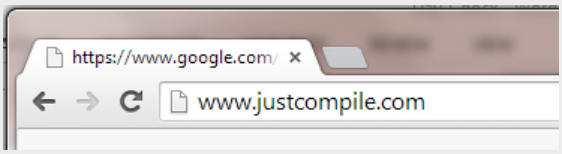
Understand Controller in Asp.Net MVC?

In order to understand Controller first we need to understand this term User interaction logic.

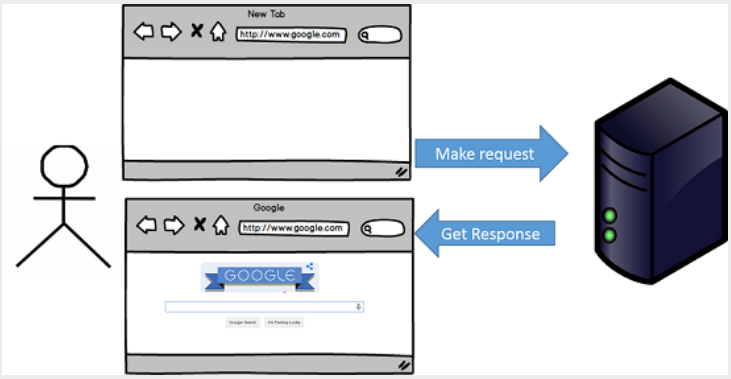
What is User Interaction logic??

Scenario 1

Did you ever gave a thought what happens, when end user hits a URL on a browser.



Browser sends request to server and server sends a response.



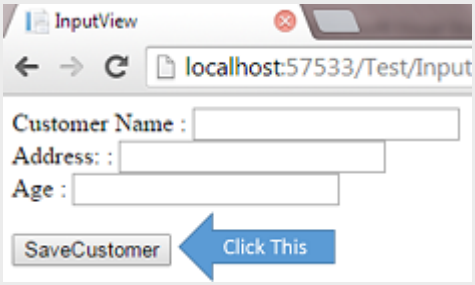
By means of such request, client is trying to interact with server. Server is able to respond back because some logic is written at server end to fulfill this request.

Some logic?? , So what exactly can be this logic?

Logic which will handle the user requests and user's interaction with server. In short User Interaction Logic

Scenario 2

It also possible that response sent by Server is an HTML response. HTML response which can consist of couple of input controls and a submit button.



Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

What will happen when "Save Customer" button is clicked?
If your answer is "Some event handler will handle the button click", then sorry ☹ .

"*In reality in web programming there is no concept of events. In case of Asp.Net Web Forms Microsoft wrote some code on behalf of us and brought us the feeling of event driven programming. It's just an abstraction or the right word would illusion.*"

When button is clicked a simple HTPP request is sent to the server. This time the difference is, values in the "Customer Name", "Address" and "Age" will be sent along with request. (In technical terms "values are posted to the server").
Ultimately, if it's a request then there must be a logic written in the server so that server can send back the response. In short there must be some user interaction logic written on the server.

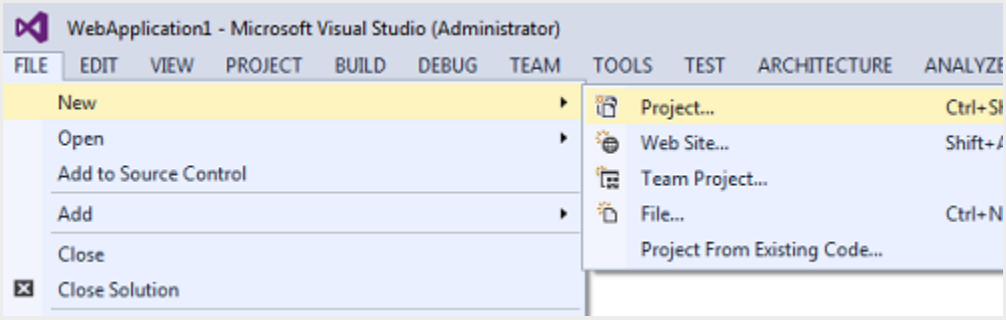
In Asp.Net MVC, the last letter C that is Controller is the one who will handle the user interaction Logic.

Understand Views in Asp.Net MVC

As we discussed earlier controller will handle the user's requests and send the response. Most commonly the response is HTML as browser understands that format much better. HTML with some images, texts, Input controls etc. Normally in technical world layer defining the user interface design is termed as UI layer and in MVC it is termed as View.

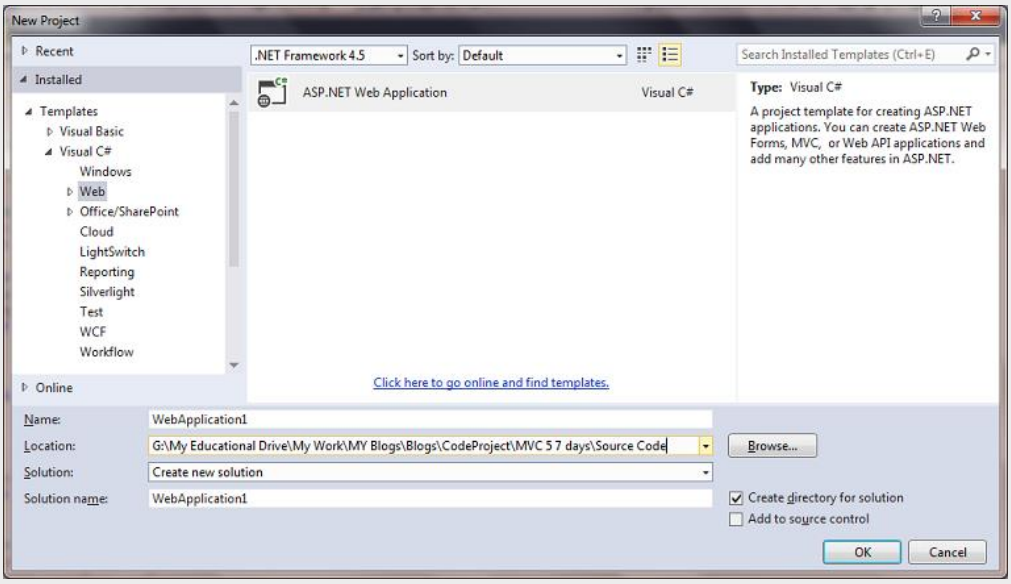
Lab 1 – Demonstrating Controller with a simple MVC hello world

Step 1 – Create MVC 5 Project
Step 1.1 Open Visual studio 2013(or higher). Click on File>>New>>Project.

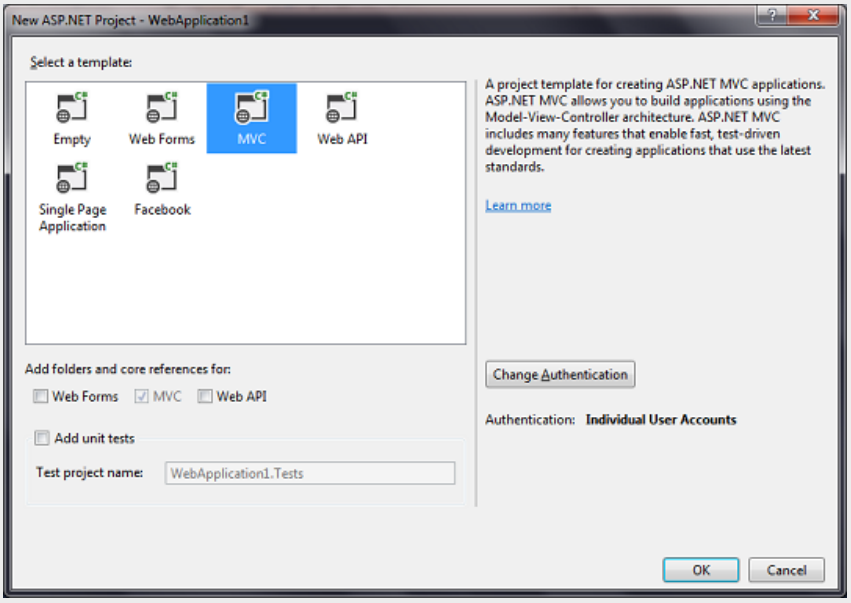


Step 1.2 Select Web Application. Put Name. Put Location and say ok.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

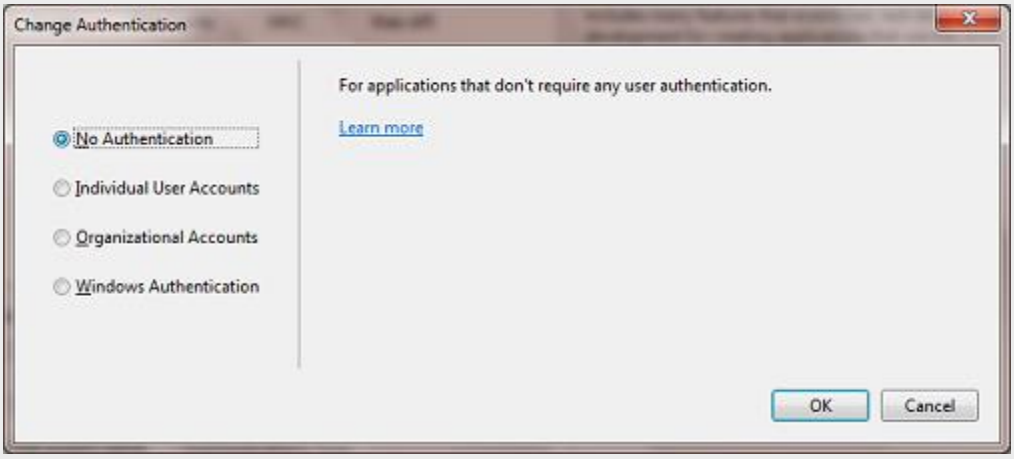


Step 1.3 Select MVC template



Step 1.4 Click on Change Authentication. Select "No Authentication" from "Change Authentication" dialog box and click ok.

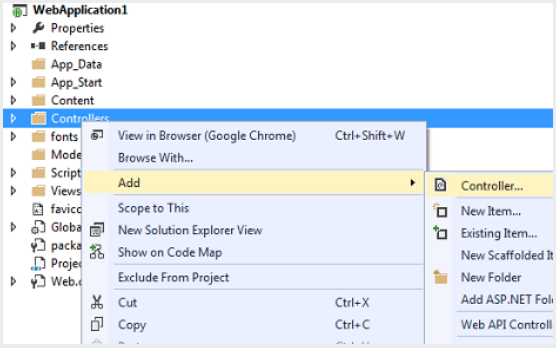
Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



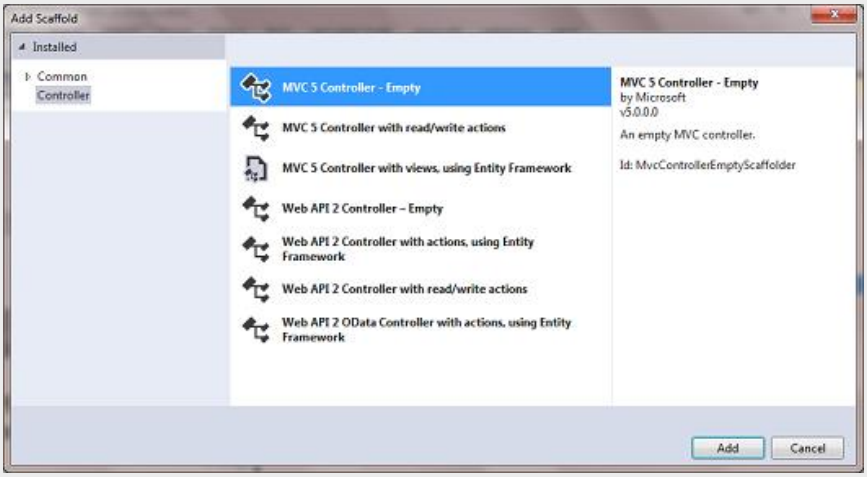
Step 1.5. Click ok.

Step 2 – Create Controller

Step 2.1. In the solution explorer, right click the controller folder and select Add>>Controller



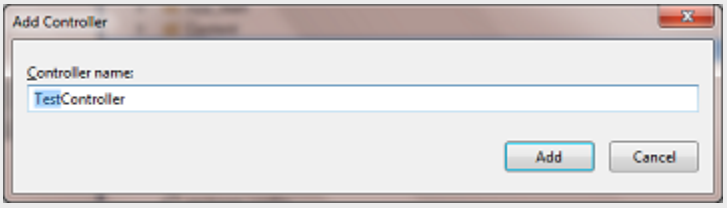
Step 2.2. Select "MVC 5 Controller – Empty" and click Add



Step 2.3.Put controller name as "TestController" and click Add.

Make sure not to delete the word controller. We will talk about it in detail soon.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



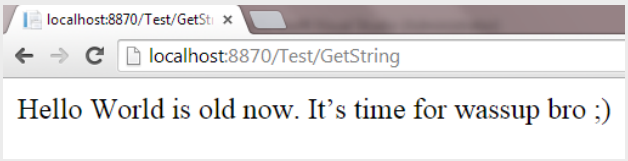
Step 3. Create Action Method

Open newly created TestController class. You will find a method inside it called "Index". Remove that method and add new public method called "GetString" as follows.

```
public class TestController : Controller
{
   public string GetString()
   {
      return "Hello World is old now. It's time for wassup bro ;)";
   }
}
```

Step 4. Execute and Test

Press F5. In the address bar put "ControllerName/ActionName" as follows. Please note do not type the word "Controller" just type "Test".



Talk on Lab 1

What is the relationship between TestController and Test?

TestController is the class name whereas Test is the controller name. When you type the controller name on the URL it should be without the word controller.

MVC follows Convention based approach.
It strictly look's into the conventions we used.
In Asp.Net MVC two things are very important.

1. How we name something?
2. Where we keep something?

What is Action Method?

Action method is simply a public method inside controller which accepts user's request and returns some response. In above example, action method "GetString" is returning a string response type.

Note: In Asp.Net Web Forms default return response is always HTML. In case we want to return something other than HTML (in Asp.Net Web Forms), we create HTTP handlers, override content type,

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

do response.end etc. It's not an easy task. In Asp.Net MVC it's very easy. If return type is 'string' you can just return string ☺ , you do not need to send complete HTML.

What will happen if we try to return an object from an action method?

Look at the following code block.

```
namespace WebApplication1.Controllers
{
   public class Customer
   {
      public string CustomerName { get; set; }
      public string Address { get; set; }
   }
   public class TestController : Controller
   {
      public Customer GetCustomer()
      {
         Customer c = new Customer();
         c.CustomerName = "Customer 1";
         c.Address = "Address1";
         return c;
      }
   }
}
```

Output of above action method will look as shown below.

When return type is some object like 'customer', it will return 'ToString()' implementation of that object.By default 'ToString()' method returns fully qualified name of the class which is "NameSpace.ClassName";

What if you want to get values of properties in above example?

Simply override "ToString" method of class as follows.

```
public override string ToString()
{
   return this.CustomerName+"|"+this.Address;
}
```

Press F5. Output will be as follows.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Is it must to decorate action methods with public access modifier?
Yes, every public method will become action methods automatically.

What about non-public methods?
They are simply methods of a class and not available publicly . In simple words these methods can not be invoked from the web.

What if we want a method to be public but not action method?
Simply decorate it with NonAction attribute as follows.

```
[NonAction]
public string SimpleMethod()
{
   return "Hi, I am not action method";
}
```

When we try to make request to above action method we will get following response.

Lab 2 – Demonstrating Views

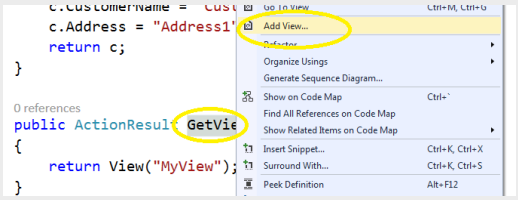
In the first lab we created a simple MVC application with just controller and simple string return type. Let us go add view part to the MVC application.

Step 1 – Create new action method
Add a new action method inside TestController as follows.

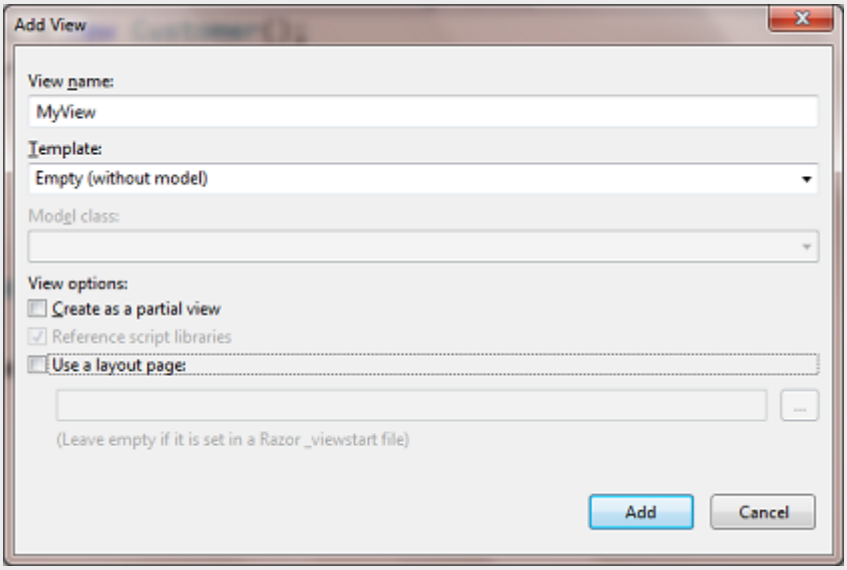
```
public ActionResult GetView()
{
   return View("MyView");
}
```

Step 2 – Create View
Step 2.1. Right click the above action method and select "Add View".

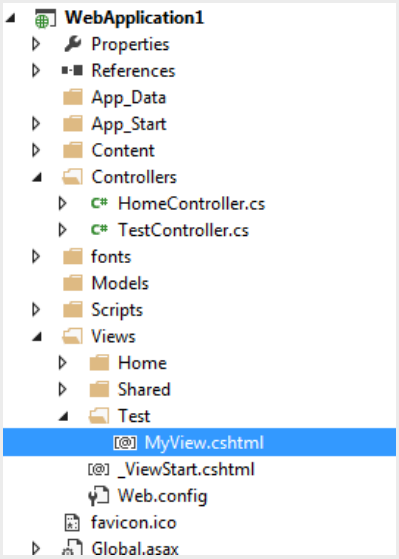
Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Step 2.2. In the "Add View" dialog box put view name as "MyView", uncheck "use a layout" checkbox and click "Add".



It will add a new view inside "Views/Test" folder in solution explored



Step 3 – Add contents to View

Open MyView.cshtml file and add contents as follows.

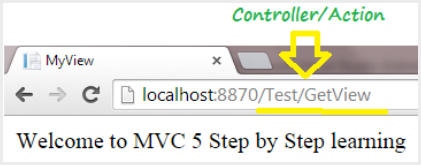
```
@{
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>MyView</title>
</head>
<body>
    <div>
        Welcome to MVC 5 Step by Step learning
    </div>
</body>
</html>
```

Step 3. Test and Execute

Press F5 and execute the application.



Talk on Lab 2

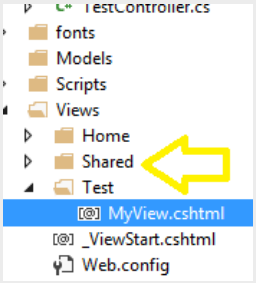
Why View is placed inside Test Folder?

In Asp.Net MVC, Views associated with the particular controller is placed inside a special folder. This special folder will be named as "ControllerName" and placed inside Views folder (located in the root folder). For every controller only those views will be available which are located inside its own folder.

For example: All the views related to Test controller will be placed inside "~/Views/Test" and Test controller can access only those views which are inside Test folder.

Can't we reuse some of the views across multiple controllers?

Yes, we can. For that we will keep those files inside a special folder called "Shared".



Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Views located inside this Shared folder will be available to all the controllers.

Is it possible that one action method is referencing more than one views?

Yes. Look at the following code.

```
public ActionResult GetView()
{
   if(Some_Condition_Is_Matching)
   {
      return View("MyView");
   }
   else
   {
      return View("YourView");
   }
}
```

Note: In MVC views and controllers are not tightly coupled. One action method can refer more than one view and one view can be referred by more than one action method (by keeping them in Shared folder). It provides better reusability

What is the purpose of View function?

View function is defined inside controller base class.

1. It will find the view inside "View/Controller" or "Views/Shared/" folder with the name Argument.SomeExtension.
    - Argument means argument passed to the View function.
    - SomeExtension means any valid extension
2. Based on the extension of the view, pass it to the corresponding View Engine.
   Example: If extension is ".aspx" then "Aspx View Engine" and if it ".cshtml" then "Razor View engine".

   Note: In MVC 5 support for aspx view engine is removed. That's why in "Add View" dialog box there is no option to select the ViewEngine.

3. View Engine parses the server syntaxes in the view and creates in pure HTML response. Example when it's razor view engine, it understand razor syntaxes written in the view and parse it to pure HTML string.
4. This HTML string will be returned to the end user.

Note: In the above example we have not used any razor syntaxes hence in this case ViewEngine just take the View and return the same as it is.

What is ActionResult?

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

ActionResult encapsulates response of a user request. In Asp.Net MVC when end user makes a request to an action method, action method may return ActionResult.

What is ViewResult?

HTML output retrieved from View function is ViewResult. In simple words it's a result of view.

What is the relation between ActionResult and ViewResult?

ActionResult is the abstract class whereas ViewResult is the multi-level child of ActionResult. Multilevel because, ViewResult is the child of ViewResultBase and ViewResultBase is the child of ActionResult.

If we want to return ViewResult why ActionResult is the ViewResult?

To achieve polymorphism. Look at the following example.

```
public ActionResult GetView()
{
   if(Some_Condition_Is_Matching)
   {
      return View("MyView");
   }
   else
   {
      return Content("Hi Welcome");
   }
}
```

In the above example, when some condition is matching we are returning we are invoking "View" function which will return ViewResult whereas in some other condition we are invoking "Content" function which is returning Content Result.

What is ContentResult?

ViewResult represents a complete HTML response whereas ContentResult represents a scalar text response. It's just like returning pure string. Difference is ContentResult is an ActionResult wrapper around string result. ContentResult is also the child of ActionResult.

Is it possible to invoke View function without Parameter?

Yes, then it will find the view with name "CurrentActionName".

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Learn MVC Project in 7 Days – Day 2

Passing Data from Controller to View

View created in the Lab 2 is very much static. In real life scenario it will display some dynamic data.
In the next lab we will display some dynamic data in the view.
View will get data from the controller in the form of Model.

Model

In Asp.Net MVC model represent the business data.

Lab 3 – Using ViewData

ViewData is a dictionary, which will contains data to be passed between controller and views.
Controller will add items to this dictionary and view reads from it. Let's do a demo.

Step 1 - Create Model class

Create a new class Called Employee inside Model folder as follows.

```
public class Employee
{
   public string FirstName { get; set; }
   public string LastName { get; set; }
   public int Salary { get; set; }
}
```

Step 2 - Get Model in controller

Create Employee object inside GetView method as follows

```
Employee emp = new Employee();
emp.FirstName = "Sukesh";
emp.LastName="Marla";
emp.Salary = 20000;
```

Note: Make sure to put using statement in the top or else we have to put fully qualified name of employee.

Step 3 – Create ViewData and return the View

Store Employee object in ViewData as follows.

```
ViewData["Employee"] = emp;
Return View("MyView")
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 4 - Display Employee Data in View

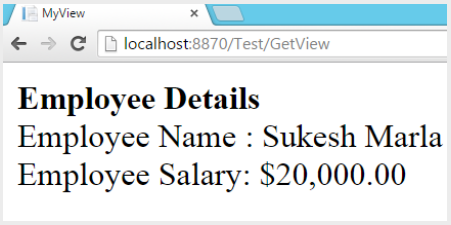
Open MyView.cshtml.

Retrieve the Employee data from the ViewData and display it as follows.

```
<div>
  @{
    WebApplication1.Models.Employee emp=(WebApplication1.Models.Employee)
      ViewData["Employee"];
  }

  <b>Employee Details </b><br />
  Employee Name : @emp.FirstName @emp.LastName <br />
  Employee Salary: @emp.Salary.ToString("C")
</div>
```

Step 5 - Test the output

Press F5 and test the application.



Talk on Lab 3

What is the difference between writing Razor code with brace brackets (that is "{" and "}") and without brace brackets?

In the last lab @emp.FirstName can be replaced with following code snippet.

```
@{
        Response.Write(emp.FirstName);
}
```

@ Without brace brackets simply display the value of variable or expression.

Why casting is required?

ViewData holds objects internally. Every time a new value is added into it, it get boxed to object type. So unboxing is required every time we try to extract value out of it.

What is the meaning of "@emp.FirstName @emp.LastName"?

It means Display First Name followed by a space and then last name.

Can we write same thing with single @ keyword?

Yes, then syntax for this will be @(emp.FirstName+" "+emp.LastName)

Why hardcoded Employee class is created in Controller?

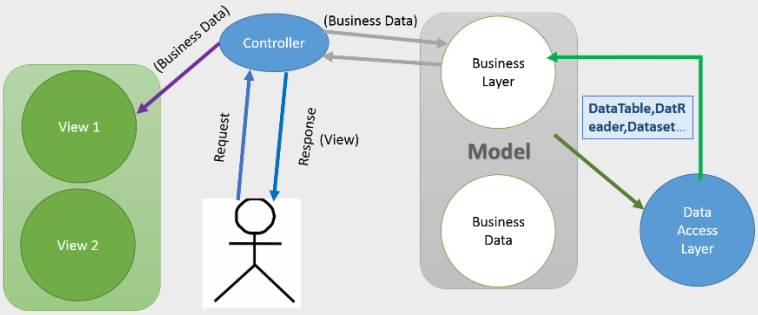
Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Just for demo purpose. In real time we will get it from may be database or wcf or web service or may be from somewhere else.

What about the Database Logic/ Data Access Layer and Business Layer?

- Data Access Layer is one of the unspoken layer in Asp.Net MVC. It's always there but never included in MVC definition.
- Business layer as explained prior, it's a part of Model.

Complete MVC structure



Lab 4 – Using ViewBag

ViewBag is just a syntactic sugar for ViewData. ViewBag uses the dynamic feature of C# 4.0 and makes ViewData dynamic.

ViewBag internally uses ViewData.

Step 1 – Create View Bag

Continue with the same Lab 3 and replace Step 3 with following code snippet.

```
ViewBag.Employee = emp;
```

Step 2 - Display Employee Data in View

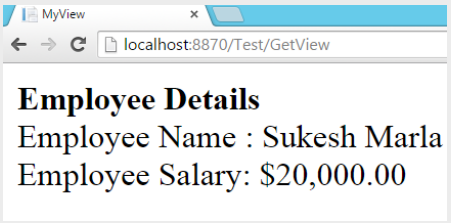
Change Step 4 with following code snippet.

```
@{
   WebApplication1.Models.Employee emp = (WebApplication1.Models.Employee)
      ViewBag.Employee;
}
<b>Employee Details</b><br />
Employee Name: @emp.FirstName @emp.LastName <br />
Employee Salary: @emp.Salary.ToString("C")
```

Step 3 - Test the output

Press F5 and test the application

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Talk on Lab 4

Can we pass ViewData and get it as ViewBag?

Yes, We can. Vice versa is also possible. As I said before, ViewBag is just a syntactic sugar for ViewData,

Problems with ViewData and ViewBag

ViewData and ViewBag is a good option for passing values between Controller and View. But in real time projects it's not a good practice to use any of them. Let's discuss couple of disadvantages of using ViewData and ViewBag.

Performance issues

Values inside the ViewData are of type Object. We have to cast the value to correct type before using it. It adds additional overhead on performance.

No Type safety and no compile time errors

If we try to cast values to wrong type or if we use wrong keys while retrieving the values, we will get runtime error. As a good programming practice, error should be tackled in compiled time.

No Proper connection between Data sent and Data Received

As a developer I personally found this as a major issue.

In MVC, controller and View are loosely connected to each other. Controller is completely unaware about what's happening in View and View is unaware about what's happening in Controller.

From Controller we can pass one or more ViewData/ViewBag values. Now when Developer writes a View, he/she have to remember what is coming from the controller. If Controller developer is different from View developer then it becomes even more difficult. Complete unawareness. It leads to many run time issues and inefficiency in development.

Lab 5 - Understand strongly typed Views

Reason for all three problems of ViewData and ViewBag is the data type. Data type of values inside ViewData, which is "Object".

Somehow if we were able to set the type of data which need to be passed between Controller and View problem will get solved and that's where strongly typed Views comes to picture.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Let's do a demo. This time we will take our View requirement to next level. If salary is greater than 15000 then it will be displayed in yellow colour or else green colour.

Step 1 – Make View a strongly typed view

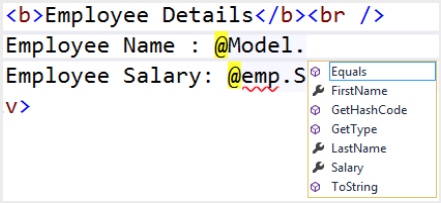
Add following statement in the top of the View

```
@model WebApplication1.Models.Employee
```

Above statement make our View a strongly typed view of type Employee.

Step 2 – Display Data

Now inside View simply type @Model and Dot (.) and in intellisense you will get all the properties of Model (employee) class.



Write down following code to display the data

```
<b>Employee Details</b><br />
Employee Name : @Model.FirstName @Model.LastName <br />
@if(Model.Salary>15000)
{
   <span style="background-color:yellow">
      Employee Salary: @Model.Salary.ToString("C")
   </span>
}
else
{
   <span style="background-color:green">
      Employee Salary: @Model.Salary.ToString("C")
   </span>
}
```

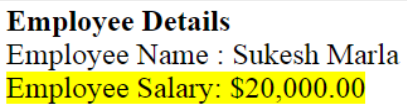
Step 3 – Pass Model data from Controller Action method

Change the code in the action method to following.

```
Employee emp = new Employee();
emp.FirstName = "Sukesh";
emp.LastName="Marla";
emp.Salary = 20000;
return View("MyView",emp);
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 4 – Test the output



Talk on Lab 5

Is it required to type fully qualified class Name (Namespace.ClassName) in View every time?

No, we can put a using statement.

```
@using WebApplication1.Models
@model Employee
```

Is it must to make View a strongly typed view always or can we go with ViewData or ViewBag sometimes?

As a best practice always make the view a strongly typed view.

Can we make our View a strongly typed view of more than one model?

No, we can't. In real time project we often end up at a point where we want to display multiple models in same view. Solution for this requirement will be discussed in next lab.

Understand View Model in Asp.Net MVC

In Lab 5 we have violated MVC principle. According to MVC, V that is View should be pure UI. It should not contain any kind of logic. We have done following three things which purely violates MVC architecture rules.

- Append First name and Last Name and Display it as Full Name - **Logic**
- Display Salary with Currency – **Logic**
- Display Salary in different color based on value. In simple words Change appearance of HTML element based on some value. – **Logic**

Other than these three issues, there is one more point worth discussion.
Let say we have situation where we want to display more than one kind of data in the View.
Example – Show Current logged in User's Name along with Employee data

We can implement this in one of the following ways.

1. Add UserName property to Employee class – Every time we want to display new data in the view, adding new property to employee class seems illogical. This new property may or may not be related to Employee.
2. Use ViewBag or ViewData – We already discussed problems of using this approach.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

ViewModel a solution

ViewModel is one of the unspoken layer in the Asp.Net MVC application. It fits between Model and View and act as data container for View

Difference between Model and ViewModel?

Model is Business specific data. It will be created based on Business and Database structure.
ViewModel is View specific data. It will be created based on the View.

How it works exactly?

It's simple.

- Controller handle the user interaction logic or in simple words, handles the user's requests.
- Controller get one or more model data.
- Controller will decide which View suits best as response for the correct request.
- Controller will create and initialises ViewModel object from Model data retrieved based on View Requirement
- Controller will pass ViewModel data to View by means of ViewData/ViewBag/Stongly typed View.
- Controller will return the view.

How View and ViewModel will be connected here?

View is going to be a strongly typed view of type ViewModel.

How Model and ViewModel will be connected?

Model and ViewModel should be independent of each other. Controller will create and initialises ViewModel object based on one or more Model object.

Let's do a small lab to understand it better.

Lab 6 – Implementing View Model

Step 1 – Create Folder

Create a new folder called ViewModels in the project

Step 2 – Create EmployeeViewModel

In order to do that, let's list all the requirement on the view

1. First Name and Last Name should be appended before displaying
2. Amount should be displayed with currency
3. Salary should be displayed in different colour (based on value)
4. Current User Name should also be displayed in the view as well

Create a new class called EmployeeViewModel inside ViewModels folder will looks like below.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
public class EmployeeViewModel
{
   public string EmployeeName { get; set; }
   public string Salary { get; set; }
   public string SalaryColor { get; set; }
   public string UserName{get;set;}
}
```

Please note, in View Model class, FirstName and LastName properties are replaced with one single property called EmployeeName, Data type of Salary property is string and two new properties are added called SalaryColor and UserName.

Step 3 – Use View Model in View

In Lab 5 we had made our View a strongly type view of type Employee. Change it to EmployeeViewModel

```
@using WebApplication1.ViewModels
@model EmployeeViewModel
```

Step 4 – Display Data in the View

Replace the contents in View section with following snippet.

```
Hello @Model.UserName
<hr />
<div>
   <b>Employee Details</b><br />
   Employee Name : @Model.EmployeeName <br />
   <span style="background-color:@Model.SalaryColor">
      Employee Salary: @Model.Salary
   </span>
</div>
```

Step 5 – Create and Pass ViewModel

In GetView action method, get the model data and convert it to ViewModel object as follows.

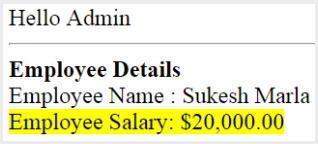
```
public ActionResult GetView()
{
   Employee emp = new Employee();
   emp.FirstName = "Sukesh";
   emp.LastName="Marla";
   emp.Salary = 20000;

   EmployeeViewModel vmEmp = new EmployeeViewModel();
   vmEmp.EmployeeName = emp.FirstName + " " + emp.LastName;
   vmEmp.Salary = emp.Salary.ToString("C");
   if(emp.Salary>15000)
   {
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
        vmEmp.SalaryColor="yellow";
    }
    else
    {
        vmEmp.SalaryColor = "green";
    }
    vmEmp.UserName = "Admin"
    return View("MyView", vmEmp);
}
```

Step 5 – Test the output

Press F5 and Test the output



Same output as Lab 5 but this time View won't contain any logic.

Talk on Lab 6

Does it means, every model will have one View Model?

No, Every View will have its corresponding ViewModel.

Is it a good practice to have some relationship between Model and ViewModel?

No, as a best practice Model and ViewModel should be independent to each other.

Should we always create ViewModel? What if View won't contain any presentation logic and it want to display Model data as it is?

We should always create ViewModel. Every view should always have its own ViewModel even if ViewModel is going to contain same properties as model.

Let's say we have a situation where View won't contain presentation logic and it want to display Model data as it is. Let's assume we won't create a ViewModel in this situation.

Problem will be, if in future requirement, if we have been asked to show some new data in our UI or if we asked to put some presentation logic we may end with complete new UI creation from the scratch. So better if we keep a provision from the beginning and Create ViewModel. In this case, in the initial stage ViewModel will be almost same as Model.

Lab 7– View With collection

In this lab we will display list of Employees in the View.

Step 1 – Change EmployeeViewModel class

Remove UserName property from EmployeeViewModel.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
public class EmployeeViewModel
{
   public string EmployeeName { get; set; }
   public string Salary { get; set; }
   public string SalaryColor { get; set; }
}
```

Step 2 – Create Collection View Model

Create a class called EmployeeListViewModel inside ViewModel folder as follows.

```
public class EmployeeListViewModel
{
   public List<EmployeeViewModel> Employees { get; set; }
   public string UserName { get; set; }
}
```

Step 3 – Change type of strongly typed view

Make MyView.cshtml a strongly typed view of type EmployeeListViewModel.

```
@using WebApplication1.ViewModels
@model EmployeeListViewModel
```

Step 4– Display all employees in the view

```
<body>
   Hello @Model.UserName
   <hr />
   <div>
     <table>
        <tr>
           <th>Employee Name</th>
           <th>Salary</th>
        </tr>
        @foreach (EmployeeViewModel item in Model.Employees)
        {
           <tr>
              <td>@item.EmployeeName</td>
              <td style="background-color:@item.SalaryColor">@item.Salary</td>
           </tr>
        }
     </table>
   </div>
</body>
```

Step 5 – Create Business Layer for Employee

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

In this lab, we will take our project to next level. We will add Business Layer to our project.

Create class called EmployeeBusinessLayer with a method called GetEmployees.

```
public class EmployeeBusinessLayer
{
   public List<Employee> GetEmployees()
   {
      List<Employee> employees = new List<Employee>();
      Employee emp = new Employee();
      emp.FirstName = "johnson";
      emp.LastName = " fernandes";
      emp.Salary = 14000;
      employees.Add(emp);

      emp = new Employee();
      emp.FirstName = "michael";
      emp.LastName = "jackson";
      emp.Salary = 16000;
      employees.Add(emp);

      emp = new Employee();
      emp.FirstName = "robert";
      emp.LastName = " pattinson";
      emp.Salary = 20000;
      employees.Add(emp);

      return employees;
   }
}
```

Step 6 – Pass data from Controller

```
public ActionResult GetView()
{
   EmployeeListViewModel employeeListViewModel = new EmployeeListViewModel();

   EmployeeBusinessLayer empBal = new EmployeeBusinessLayer();
   List<Employee> employees = empBal.GetEmployees();

   List<EmployeeViewModel> empViewModels = new List<EmployeeViewModel>();

   foreach (Employee emp in employees)
   {
      EmployeeViewModel empViewModel = new EmployeeViewModel();
      empViewModel.EmployeeName = emp.FirstName + " " + emp.LastName;
      empViewModel.Salary = emp.Salary.ToString("C");
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
        if (emp.Salary > 15000)
        {
            empViewModel.SalaryColor = "yellow";
        }
        else
        {
            empViewModel.SalaryColor = "green";
        }
        empViewModels.Add(empViewModel);
    }
    employeeListViewModel.Employees = empViewModels;
    employeeListViewModel.UserName = "Admin";
    return View("MyView", employeeListViewModel);
}
```

Step 7 – Execute and Test the Output

Press F5 and execute the application.

Hello Admin

Employee Name	Salary
johnson fernandes	$14,000.00
michael jackson	$16,000.00
robert pattinson	$20,000.00

Talk on Lab 7

Can we make View a strongly typed view of List<EmployeeViewModel>?

Yes, we can.

Why we create a separate class called EmployeeListViewModel, why didn't we made View a strongly typed view of type List<EmployeeViewModel>?

If we use List<EmployeeViewModel> directly instead of EmployeeListViewModel then there will be two problems.

1. Managing future presentation logic.
2. Secondly UserName property. UserName is not associated with individual employees. It is associated with complete View.

Why we removed UserName property from EmployeeViewModel and made it part of EmployeeListViewModel?

UserName is going to be same for all the employees. Keeping UserName property inside EmployeeViewModel just increase the redundant code and also increases the overall memory requirement for data.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Learn MVC Project in 7 Days – Day 3

Data Access Layer

A real time project is incomplete without Database. In our project we didn't spoke database layer yet. First Lab of Day 3 will be all about database and database layer.

Here we will use SQL Server and Entity Framework for creating Database and Database Access layer respectively.

What is Entity Framework in simple words?

It's an ORM tool. ORM stands for Object Relational Mapping.

In RDBMS world, we speak in terms of Tables and Columns whereas in .net world (which is an object oriented world), we speak in terms of Classes, objects and properties.

When we think about any data driven application we end up with following two things.

- Write code for communicating with database (called Data Access Layer or Database logic)
- Write code for mapping Database data to object oriented data or vice versa.

ORM is a tool which will automate these two things. Entity framework is Microsoft ORM tool.

What is Code First Approach?

In Entity framework we can follow one of these three approaches

1. Database First approach – Create database with tables, columns, relations etc. and Entity framework will generates corresponding Model classes (Business entities) and Data Access Layer code.
2. Model First approach – In this approach Model classes and relationship between them will be defined manually using Model designer in Visual studio and Entity Framework will generate Data Access Layer and Database with tables, columns, relations automatically.
3. Code First approach – In this approach manually POCO classes will be created. Relationship between those classes will be defined by means of code. When application executes for the first time Entity framework will generate Data Access Layer and Database with tables, column and relations automatically in the database server.

What is mean by POCO classes?

POCO stands for "Plain Old CLR objects". POCO classes means simple .Net classes we create.

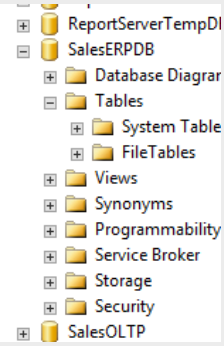
In our previous example Employee class was simply a POCO class.

Lab 8 – Add Data Access layer to the project

Step 1– Create Database

Connect to the SQL Server and create new Database called "SalesERPDB".

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



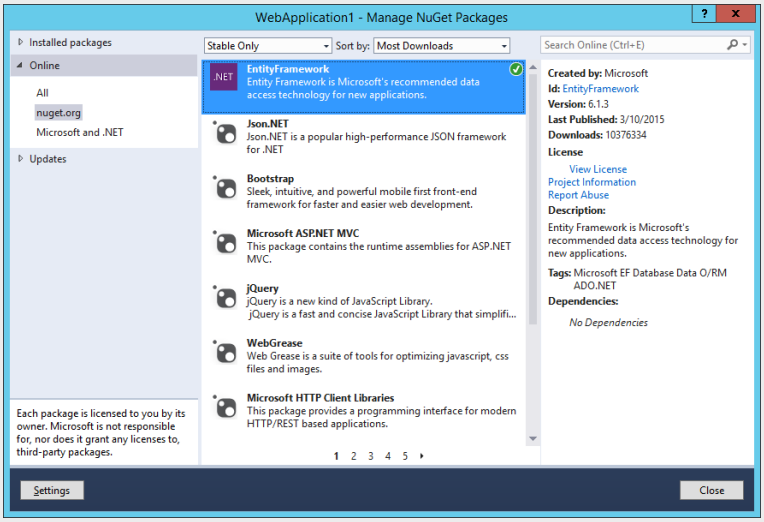
Step 2 – Create ConnectionString

Open Web.config file and inside Configuration section add following section

```
<connectionStrings>
  <add connectionString="Data Source=(local);Initial Catalog=SalesERPDB;Integrated Security=True"
      name="SalesERPDAL"
      providerName="System.Data.SQLClient"/>
</connectionStrings>
```

Step 3 – Add Entity Framework reference

Right click the project >> Manage Nuget packages. Search for Entity Framework and click install.



Step 4 – Create Data Access layer.

- Create a new folder called "DataAccessLayer" in the root folder and inside it create a new class called "SalesERPDAL"
- Put using statement at the top as follows.

```
using System.Data.Entity;
```

- Derive "SalesERPDAL" class from DbContext class

```
public class SalesERPDAL: DbContext
{
}
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 5 – Create primary key field for employee class

Open Employee class and put using statement at the top as follows.

```
using System.ComponentModel.DataAnnotations;
```

Add EmployeeId property in Employee class and mark it as Key attribute.

```
public class Employee
{
   [Key]
   public int EmployeeId  { get; set; }
   public string FirstName { get; set; }
   public string LastName { get; set; }
   public int Salary { get; set; }
}
```

Step 6 – Define mapping

Put following using statement in the top for "SalesERPDAL" class

```
using WebApplication1.Models;
```

Override OnModelCreating method in SalesERPDAL class as follows.

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
   modelBuilder.Entity<Employee>().ToTable("TblEmployee");
   base.OnModelCreating(modelBuilder);
}
```

Note: In above code snippet "TblEmployee" represents the table name. It automatically get created in runtime.

Step 7 – Create property to hold Employees in Database

Create a new property called Employee in "SalesERPDAL" class as follows

```
public DbSet<Employee> Employees
```

DbSet will represent all the employees that can be queried from the database.

Step 8 – Change Business Layer Code and get data from Database

Open EmployeeBusinessLayer class. Put using statement in the top.

```
using WebApplication1.DataAccessLayer;
```

Now change GetEmployees method class as follows.

```
public List<Employee> GetEmployees()
{
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

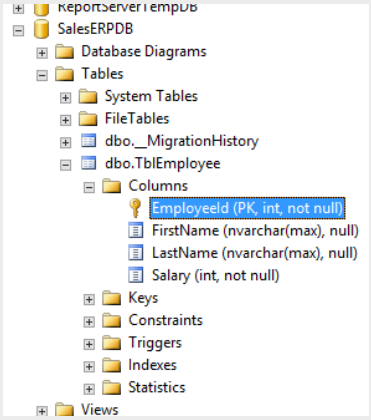
```
    SalesERPDAL salesDal = new SalesERPDAL();
    return salesDal.Employees.ToList();
}
```

Step 9 – Execute and Test

Press F5 and execute the application.

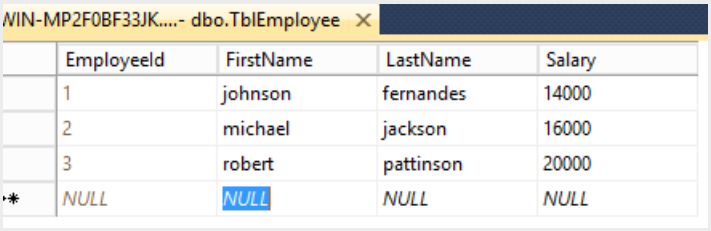
Right now we don't have any employees in the database so we will see a blank grid.

Check the database. Now we have a table called TblEmployee with all the columns.



Step 9 – Insert Test Data

Add some dummy data to TblEmployee table.

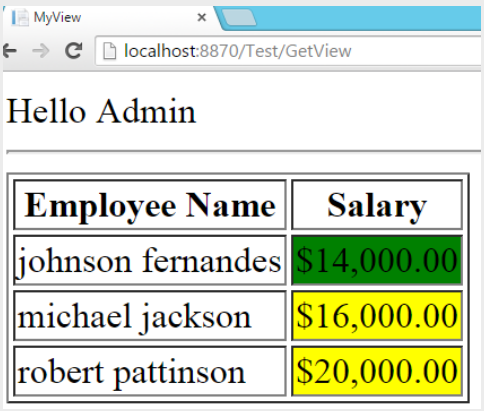


EmployeeId	FirstName	LastName	Salary
1	johnson	fernandes	14000
2	michael	jackson	16000
3	robert	pattinson	20000
NULL	NULL	NULL	NULL

Step 10 – Execute and test the application

Press F5 and run the application again.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Here we go ☺

Talk on Lab 8

What is DbSet?

DbSet simply represent the collection of all the entities that can be queried from the database. When we write a Linq query again DbSet object it internally converted to query and fired against database. In our case "Employees" is a DbSet which will hold all the "Employee" entities which can be queried from database. Every time we try to access "Employees" it gets all records in the "TblEmployee" table and convert it to "Employee" object and return the collection.

How connection string and data access layer is connected?

Mapping will be done based on name. In our example ConnectionString Name and Data Access Layer class name is same that is "SalesERPDAL", hence automatically mapped.

Can we change the ConnectionString name?

Yes, in that case we have to define a constructor in Data Access Layer class as follows.

```
public SalesERPDAL():base("NewName")
{
}
```

Organize everything

Just to make everything organized and meaningful let's do couple of changes.

Step 1 - Rename

- "TestController" to "EmployeeController"
- GetView action method to Index
- Test folder (inside Views folder) to Employee
- and "MyView" view to "Index"

Step 2 – Remove UserName property from EmployeeListViewModel

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 3 – Remove UserName from View

Open Views/Employee.Index.cshtml View and remove username from it.

In simple words, remove following block of code.

```
Hello @Model.UserName
  <hr />
```

Step 2 – Change Index Action Method in EmployeeController

Accordingly Change the code in Index action in EmployeeController as follows.

```
public ActionResult Index()
{
   ......
   ......
   ......
   employeeListViewModel.Employees = empViewModels;
   //employeeListViewModel.UserName = "Admin"; -->Remove this line -->Change1
   return View("Index", employeeListViewModel); //-->Change View Name -->Change 2
}
```

Now at the time of execution URL will **"..../Employee/Index"**

Lab 9 – Create Data Entry Screen

Step 1 – Create action method

Create an action method called "AddNew" in EmployeeController as follows

```
public ActionResult AddNew()
{
   return View("CreateEmployee");
}
```

Step 2 – Create View

Create a view called "CreateEmployee" inside View/Employee folder as follows.

```
@{
   Layout = null;
}
<!DOCTYPE html>
<html>
<head>
   <meta name="viewport" content="width=device-width" />
   <title>CreateEmployee</title>
</head>
<body>
   <div>
      <form action="/Employee/SaveEmployee" method="post">
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
        First Name: <input type="text" id="TxtFName" name="FirstName" value="" /><br />
        Last Name: <input type="text" id="TxtLName" name="LastName" value="" /><br />
        Salary: <input type="text" id="TxtSalary" name="Salary" value="" /><br />
        <input type="submit" name="BtnSave" value="Save Employee" />
        <input type="button" name="BtnReset" value="Reset" />
    </form>
  </div>
</body>
</html>
```

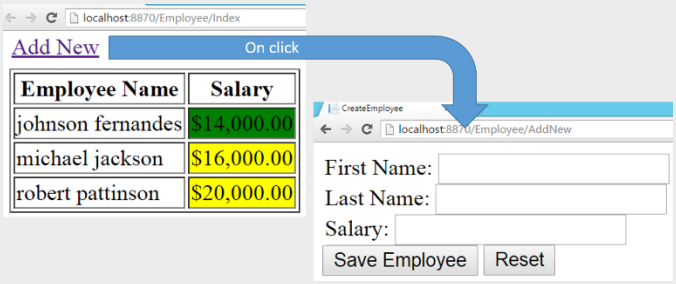
Step 3 – Create a link in Index View

Open Index.cshtml and add a hyperlink pointing to AddNew Action URL.

```
<a href="/Employee/AddNew">Add New</a>
```

Step 4 – Execute and Test the application

Press F5 and execute the application



Talk on Lab 9

What is the purpose of form tag?

In day 1 of the series we have understood that "Web world won't follow Event driven programming model. It follows request-response model. End user make the request and server sends response." Form tag is one of the way to make request in HTML. As soon as the submit button inside form tag gets clicked, a request will be sent to the URL specified in action attribute.

What is method attribute in Form tag?

It decides the type of request. Request may be one of the following four types - get, post, put and delete.

As per the web standards we should use–

- Get - > When we want to get something
- Post -> When we want to create something
- Put -> When we want to update something
- Delete -> when we want to delete something.

How making request using Form tag is different from making request via browser address bar or hyperlink?

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

When request is made with the help of Form tag, values of all the input controls are sent with the request for processing.

What about checkbox, radio buttons and Dropdowns? Will values of this controls also sent?
Yes, All input controls (input type=text, type=radio, type=checkbox) and also dropdowns (which represented as "Select" element).

How values will be sent to server?
When request is of type Get, Put or Delete, values will be sent as Query string parameters.
When it's a post request values will be sent as posted data.

What is the purpose of name attribute in input controls?
As discussed before values of all input controls will be sent along with request when submit button is clicked. It makes server receive more than one value at a time. To distinguish each value separately while sending every value is attached with one key and that key will be simply "name" attribute.

Does name and id attribute serves same purpose?
No, as per last question "name" attribute will be used internally by HTML when the request is being sent whereas "id" attribute will be used by developers inside JavaScript for some dynamic stuffs.

What is the difference between "input type=submit" and "input type=button"?
Submit button will be specially used when we want to make request to the server whereas simple button will be used to perform some custom client side actions. Simple button won't do anything by its own.

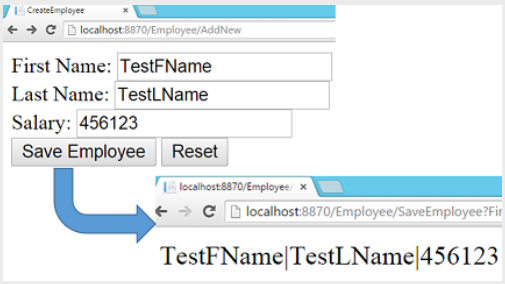
Lab 10 – Get posted data in Server side/Controllers

Step 1 – Create SaveEmployee Action method
Inside Employee Controller create an action method called SaveEmployee as follows.

```
public string SaveEmployee(Employee e)
{
        return e.FirstName + "|"+ e.LastName+"|"+e.Salary;
}
```

Step 2 – Execute and Test
Press F5 and execute the application.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Talk on Lab 10

How Textbox values are updated in Employee object inside action method?

In Asp.Net MVC there is a concept called as Model Binder.

- Model Binder will executes automatically whenever a request is made to an action method (action method which will contain parameter).
- Model binder will iterate though all primitive parameters of a method and then it will compare name of the parameter with each key in the incoming data (Incoming data means either posted data or query string).When match is found, corresponding incoming data will be assigned to the parameter.
- After that Model binder will iterate through each and every property of each and every class parameter and compare each property name with each key in incoming data. When match is found, corresponding incoming value will be assigned to the parameter.

What will happen when two parameters are specified, one as "Employee e" and second as "string FirstName"?

FirstName will be updated in both primitive FirstName variable and e.FirstName property.

Will Model Binder work with composition relationship?

Yes it will, but in that case name of the control should be given accordingly.

Example

Let say we have Customer class and Address class as follows

```
public class Customer
{
        public string FName{get;set;}
        public Address address{get;set;}
}
public class Address
{
        public string CityName{get;set;}
        public string StateName{get;set;}
}
```

In this case Html should look like this

```
...
...
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
...
<input type="text" name="FName">
<input type="text" name="address.CityName">
<input type="text" name="address.StateName">
....
...
```

Lab 11 – Reset and Cancel button

Step 1 – Add Reset and Cancel button

Add a Reset and Cancel button as follows

```
...
...
...
<input type="submit" name="BtnSubmit" value="Save Employee" />
<input type="button" name="BtnReset" value="Reset" onclick="ResetForm();" />
<input type="submit" name="BtnSubmit" value="Cancel" />
```

Note: Save button and Cancel button have same "Name" attribute value that is "BtnSubmit".

Step 2 – define ResetForm function

In Head section of Html add a script tag and inside that create a JavaScript function called ResetForm as follows.

```
<script>
  function ResetForm() {
    document.getElementById('TxtFName').value = "";
    document.getElementById('TxtLName').value = "";
    document.getElementById('TxtSalary').value = "";
  }
</script>
```

Step 3 – Implement cancel click in EmplyeeController's SaveEmployee action method.

Change SaveEmployee action method as following

```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
  switch (BtnSubmit)
  {
    case "Save Employee":
      return Content(e.FirstName + "|" + e.LastName + "|" + e.Salary);
    case "Cancel":
      return RedirectToAction("Index");
  }
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

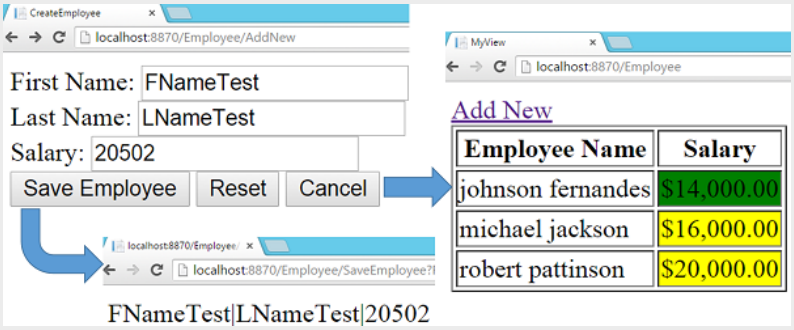
```
    return new EmptyResult();
}
```

Step 4 – Execute the application.

Press F5 and execute the application. Navigate to the AddNew screen by clicking "Add New" link.

Step 5 – Test Reset functionality

Step 6 – Test Save and Cancel functionality



Talk on Lab 11

Why same name is given to both Save and Cancel button?

We know that, as soon as submit button is clicked, a request is sent to the server. Along with the request values of all the input controls will be sent.

Submit button is also an input button. Hence value of the submit button (which is responsible for the request) will be sent too.

When Save button will be clicked, value of Save button that is "Save Employee" will be sent with request and when Cancel button is clicked, value of Cancel button that is "Cancel" will sent with request.

In Action method, Model Binder will do remaining work. It will update the parameter values with values in input data (coming with request)

What are the other ways to implement multiple submit buttons?

There are many ways. I would like to discuss three of them.

1. Hidden Form element

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 1 – Create a hidden form element in View as follows.

```
<form action="/Employee/CancelSave" id="CancelForm" method="get" style="display:none">

</form>
```

Step 2 – Change Submit button to normal button and post above form with the help of JavaScript.

```
<input type="button" name="BtnSubmit" value="Cancel"
onclick="document.getElementById('CancelForm').submit()" />
```

2. Change action URL dynamically using JavaScript

```
<form action="" method="post" id="EmployeeForm" >
...
...
        <input type="submit" name="BtnSubmit" value="Save Employee"
onclick="document.getElementById('EmployeeForm').action = '/Employee/SaveEmployee'" />
...
        <input type="submit" name="BtnSubmit" value="Cancel"
onclick="document.getElementById('EmployeeForm').action = '/Employee/CancelSave'" />
    </form>
```

3. Ajax
   Instead of submit button use simple input button and onclick of it make pure Ajax request using jQuery or any other library.

Why we have not used input type=reset for implementing Reset functionality?

Input type=reset control won't clear the values, it just set the value to default value of a control.

Example:

```
<input type="text" name="FName" value="Sukesh">
```

In above example default value of control is "Sukesh".

If we use input type=reset for implementing Reset functionality then by default "Sukesh" will be set in the textbox every time "reset" button is clicked.

What if names are not matching with property names of the classes?

This is a very common question during interviews.

Let say we have HTML as follows

```
First Name: <input type="text" id="TxtFName" name="FName" value="" /><br />
Last Name: <input type="text" id="TxtLName" name="LName" value="" /><br />
Salary: <input type="text" id="TxtSalary" name="Salary" value="" /><br />
```

Now our Model class contain property names as FirstName, LastName and Salary. Hence default model binder won't work here.

In this situation we have following three solutions

- Inside action method, retrieve posted values using Request.Form syntax and manually construct the Model object as follows.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
public ActionResult SaveEmployee()
{
      Employee e = new Employee();
      e.FirstName = Request.Form["FName"];
      e.LastName = Request.Form["LName"];
      e.Salary = int.Parse(Request.Form["Salary"])
....
....
}
```

- Use parameter names and Creates Model object manually as follows.

```
public ActionResult SaveEmployee(string FName, string LName, int Salary)
{
   Employee e = new Employee();
   e.FirstName = FName;
   e.LastName = LName;
   e.Salary = Salary;
....
....
}
```

- Create Custom Model Binder and replace default model binder as follows.

Step 1 – Create Custom Model Binder

```
public class MyEmployeeModelBinder : DefaultModelBinder
{
   protected override object CreateModel(ControllerContext controllerContext,
ModelBindingContext bindingContext, Type modelType)
   {
      Employee e = new Employee();
      e.FirstName = controllerContext.RequestContext.HttpContext.Request.Form["FName"];
      e.LastName = controllerContext.RequestContext.HttpContext.Request.Form["LName"];
      e.Salary =
int.Parse(controllerContext.RequestContext.HttpContext.Request.Form["Salary"]);
      return e;
   }
}
```

Step 2- Replace default model binder with this new model binder

```
public ActionResult SaveEmployee([ModelBinder(typeof(MyEmployeeModelBinder))]Employee
e, string BtnSubmit)
{
          ......
```

What does RedirecttToFunction do?

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

It generates RedirectToRouteResult Just like ViewResult and ContentResult (discussed in Day 1), RedirectToRouteResult is a child of ActionResult. It represents the redirect response. When browser receives RedirectToRouteResult, it makes new request to new action method.
Note: Here browser is responsible for new request hence URL will get updated to new URL.

What is EmptyResult?
One more child of ActionResult. When browser receives EmptyResult as a response it simply displays blank white screens. It simply represents "No Result".
In our example this situation won't happen. Just to make sure that all code paths returns a value EmptyResult statement was written.
Note: When ActionMethod return type is Void, it is equivalent to EmptyResult.

Lab 12 – Save records in database and update Grid

Step 1 – Create SaveEmployee in EmployeeBusinessLayer as follows

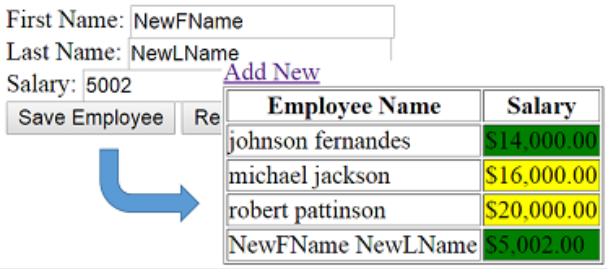
```
public Employee SaveEmployee(Employee e)
{
   SalesERPDAL salesDal = new SalesERPDAL();
   salesDal.Employees.Add(e);
   salesDal.SaveChanges();
   return e;
}
```

Step 2 – Change SaveEmployee Action method
In EmployeeController change the SaveEmployee action method code as follows.

```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
   switch (BtnSubmit)
   {
      case "Save Employee":
         EmployeeBusinessLayer empBal = new EmployeeBusinessLayer();
         empBal.SaveEmployee(e);
         return RedirectToAction("Index");
      case "Cancel":
         return RedirectToAction("Index");
   }
   return new EmptyResult();
}
```

Step 3 – Execute and Test
Press F5 and execute the application. Navigate to Data entry screen and put some valid values.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Lab 13 – Add Server side Validation

In Lab 10 we have seen basic functionality of Model Binder. Let understand a little more about same.

- Model binder updates the Employee object with the posted data.
- But this is not the only functionality performed by Model Binder. Model Binder also updates ModelState. ModelState encapsulates the state of the Model.
    - It have a property called IsValid which determines whether the Model (that is Employee object) gets successfully updated or not. Model won't update if any of the server side validation fails.
    - It holds validation errors.
      Example: ModelState["FirstName "].Errors will contain all errors related to First Name
    - It holds the incoming value(Posted data or Query String data)

In Asp.net MVC we use DataAnnotations to perform server side validations.

Before we get into Data Annotation lets understand few more things about Model Binder

How Model Binder work with primitive datatypes

When Action method contain primitive type parameter, Model Binder will compare name of the parameter with each key in the incoming data (Incoming data means either posted data or query string).

- When match is found, corresponding incoming data will be assigned to the parameter.
- When match is not found, parameter will be assigned with default value. (Default value – For integer it is 0 (zero), for string it is null etc.)
- In case assignment is not possible because of datatype mismatch exception will be thrown.

How Model Binder work with classes

When parameter is a Class parameter, Model Binder will iterate through all properties of all the class and compare each property name with each key in incoming data.

- When match is found,
    - If corresponding incoming value is empty, then
        - Null value will be assigned to property. If null assignment is not possible, default value will be set and ModelState.IsValid will be set to false.
        - If null assignment is possible but will be considered as invalid value because of the validation attached to the property then null be assigned as value and ModelState.IsValid will be set to false.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

- If corresponding incoming value is non empty,
  - In case assignment is not possible because of datatype mismatch or Server side validation failure null value will be assigned and ModelState.IsValid will be set to false.
    - If null assignment is not possible, default value will be set
- When match is not found, parameter will be assigned with default value. (Default value – For integer it is 0 (zero), for string it is null etc.) In this case ModelState.IsValid will remain unaffected.

Let's understand same by adding validation feature to our on-going project.

Step 1 – Decorate Properties with DataAnnotations

Open Employee class from Model folder and decorate FirstName and LastName property with DataAnnotation attribute as follows.

```
public class Employee
{
....
.....
   [Required(ErrorMessage="Enter First Name")]
   public string FirstName { get; set; }

   [StringLength(5,ErrorMessage="Last Name length should not be greater than 5")]
   public string LastName { get; set; }
.....
....
}
```

Step 2 – Change SaveEmployee Action method

Open EmplyeeController and Change SaveEmployee Action method as follows.

```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
   switch (BtnSubmit)
   {
      case "Save Employee":
         if (ModelState.IsValid)
         {
            EmployeeBusinessLayer empBal = new EmployeeBusinessLayer();
            empBal.SaveEmployee(e);
            return RedirectToAction("Index");
         }
         else
         {
            return View("CreateEmployee ");
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
        }
      case "Cancel":
        return RedirectToAction("Index");
    }
    return new EmptyResult();
}
```

Note: As you can see, When ModelState.IsValid is false response of SaveEmployee button click is ViewResult pointing to "CreateEmployee" view.

Step 3 – Display Error in the View

Change HTML in the "Views/Index/CreateEmployee.cshtml" to following.

This time we will format our UI a little with the help of "table" tag;

```
<table>
<tr>
  <td>
    First Name:
  </td>
  <td>
    <input type="text" id="TxtFName" name="FirstName" value="" />
  </td>
</tr>
<tr>
  <td colspan="2" align="right">
    @Html.ValidationMessage("FirstName")
  </td>
</tr>
<tr>
  <td>
    Last Name:
  </td>
  <td>
    <input type="text" id="TxtLName" name="LastName" value="" />
  </td>
</tr>
<tr>
  <td colspan="2" align="right">
    @Html.ValidationMessage("LastName")
  </td>
</tr>

<tr>
  <td>
    Salary:
  </td>
```

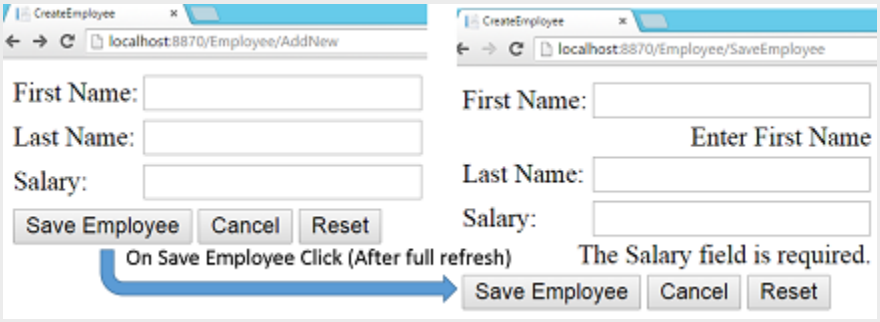
Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
    <td>
        <input type="text" id="TxtSalary" name="Salary" value="" />
    </td>
</tr>
<tr>
    <td colspan="2" align="right">
        @Html.ValidationMessage("Salary")
    </td>
</tr>

<tr>
    <td colspan="2">
        <input type="submit" name="BtnSubmit" value="Save Employee" />
        <input type="submit" name="BtnSubmit" value="Cancel" />
        <input type="button" name="BtnReset" value="Reset" onclick="ResetForm();" />
    </td>
</tr>
</table>
```

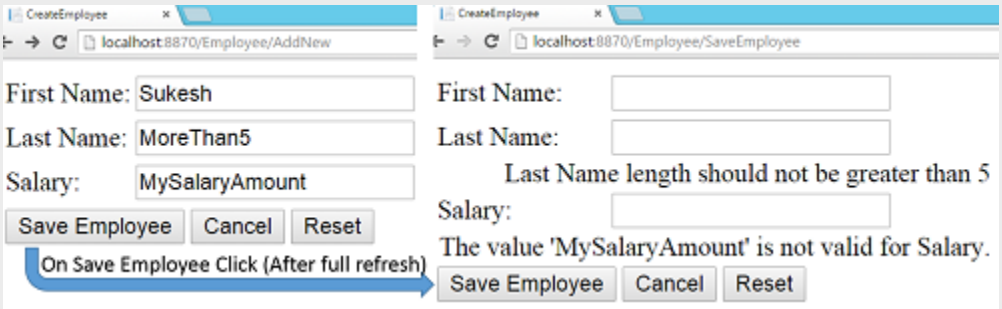
Step 4 – Execute and Test

Press F5 and execute the application. Navigate to "Employee/AddNew" action method and test the application.

Test 1



Test 2



Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Note: You may end up with following error.
"The model backing the 'SalesERPDAL' context has changed since the database was created. Consider using Code First Migrations to update the database."

To remove this error, simply add following statement in Application_Start in Global.asax file.

```
Database.SetInitializer(new DropCreateDatabaseIfModelChanges<SalesERPDAL>());
```

Database class exists inside System.Data.Entity namespace

Soon we are going to start a detail series on Entity Framework Code First approach where will speak about this in detail. This article is intended to MVC and we are try to stick with it :)

Talk on lab 13

What does @Html.ValidationMessage do?

- @ means it's a razor code
- Html is an instance of HtmlHelper class available inside view.
- ValidationMessage is a function of HtmlHelper class which displays the error message

How ValidationMessage function works?

ValidationMessage is a function. It executes at runtime. As we discussed earlier, Model Binder updates the ModelState. ValidationMessage displays the error message from ModelState based on Key. Example: ValidationMessage("FirstName") displays the error message related to First Name.

Do we have more attributes like required and StringLength?

Yes, here are some

- DataType – Make sure that data is of particular type like email, credit card number, URL etc.
- EnumDataTypeAttribute – Make sure that value exist in an enumeration.
- Range Attribute – Make sure that value is in a particular range.
- Regular expression- Validates the value against a particular regular expression.
- Required – Make sure that value is provided.
- StringthLength – Validates the string for maximum and minimum number of characters.

How Salary is getting validated?

We have not added any Data Annotation attribute to Salary attribute but still it's getting validated. Reason for that is, Model Binder also considers the datatype of a property while updating model.

In Test 1 – we had kept salary as empty string. Now in this case, as per the Model binder explanation we had (In Lab 13), ModelState.IsVaid will be false and ModelState will hold validation error related to Salary which will displayed in view because of Html.ValidationMessage("Salary")
In Test 2 – Salary data type is mismatched hence validation is failed.

Is that means, integer properties will be compulsory by default?

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Yes, Not only integers but all value types because they can't hold null values.

What if we want to have a non-required integer field?

Make it nullable?

```
public int? Salary{get;set;}
```

How to change the validation message specified for Salary?

Default validation support of Salary (because of int datatype) won't allow us to change the validation message. We achieve the same by using our own validation like regular expression, range or Custom Validator.

Why values are getting cleared when validation fails?

Because it's a new request. DataEntry view which was rendered in the beginning and which get rendered later are same from development perspective but are different from request perspective. We will learn how to maintain values in Day 4.

Can we explicitly ask Model Binder to execute?

Yes simply remove parameters from action method. It stops default model binder from executing by default.

In this case we can use UpdateModel function as follows.

```
Employee e = new Employee();
UpdateModel<Employee>(e);
```

Note: UpdateModel won't work with primitive datatypes.

What is the difference between UpdateModel and TryUpdateModel method?

TryUpdateModel will be same as UpdateModel except one added advantage.

UpdateModel will throw an exception if Model adaption fails because of any reason. In case of UpdateModel function ModelState.IsValid will not be of any use.

TryUpdateModel is exactly same as keeping Employee object as function parameter. If updating fails ModelState.IsValid will be false;

What about client side validation?

It should be done manually unless and until we are using HTML Helper classes.

We are going to talk about both manual client side validation and automatic client side validation with the help of HTML helper classes in day 4.

Can we attach more than one DataAnnotation attribute to same property?

Yes we can. In that case both validations will fire.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Lab 14 – Custom Server side validation

Step 1 – Create Custom Validation

Create a new class Called FirstNameValidation as follow

```
public class FirstNameValidation:ValidationAttribute
{
   protected override ValidationResult IsValid(object value, ValidationContext validationContext)
   {
      if (value == null) // Checking for Empty Value
      {
         return new ValidationResult("Please Provide First Name");
      }
      else
      {
         if (value.ToString().Contains("@"))
         {
            return new ValidationResult("First Name should contain @");
         }
      }
      return ValidationResult.Success;
   }
}
```

Step 2- Attach it to First Name

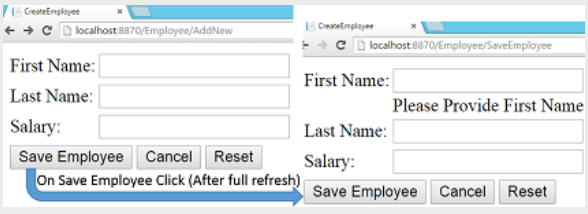
Open Employee class and remove the default "Required" attribute from FirstName property and attach FirstNameValidation as follows.

```
[FirstNameValidation]
public string FirstName { get; set; }
```

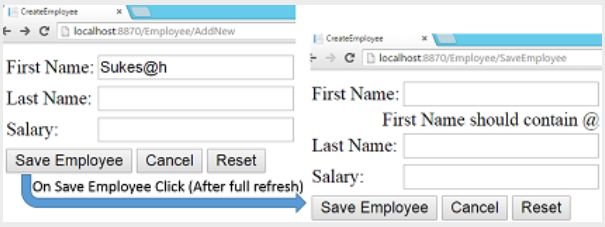
Step 3 – Execute and Test

Press F5. Navigate to "Employee/AddNew" action.

Test 1



Test 2



Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Learn MVC Project in 7 Days – Day 4

Lab 15 – Retaining values on Validation Error

In Lab 13 we introduced Server side validation and in Lab 14 we took it to next level by adding a Custom Validation support.

I strongly recommend you to rerun to the Lab 14 once again. Execute it and understand the code and output both perfectly.

In the Lab 15 we will learn how to repopulate values on validation failure.

Step 1 – Create CreateEmployeeViewModel

Create a new class in ViewModel folder as follows.

```
public class CreateEmployeeViewModel
{
   public string FirstName { get; set; }
   public string LastName { get; set; }
   public string Salary { get; set; }
}
```

Step 2 – Change SaveEmployee action method

For repopulation we will simply reuse the Employee object created by Model Binder. Change SaveEmployee Action method as follows.

```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
   switch (BtnSubmit)
   {
      case "Save Employee":
         if (ModelState.IsValid)
         {
            EmployeeBusinessLayer empBal = new EmployeeBusinessLayer();
            empBal.SaveEmployee(e);
            return RedirectToAction("Index");
         }
         else
         {
                  CreateEmployeeViewModel vm = new CreateEmployeeViewModel();
                  vm.FirstName = e.FirstName;
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
                vm.LastName = e.LastName;
                if (e.Salary.HasValue)
                {
                   vm.Salary = e.Salary.ToString();
                }
                else
                {
                   vm.Salary = ModelState["Salary"].Value.AttemptedValue;
                }
                return View("CreateEmployee", vm); // Day 4 Change - Passing e here
            }
        case "Cancel":
            return RedirectToAction("Index");
    }
    return new EmptyResult();
}
```

Step 3 –Repopulate values in View

Step 2.1 Make View a strongly typed view

Put following code in the top of CreateEmployee View.

```
@using WebApplication1.ViewModels
@model CreateEmployeeViewModel
```

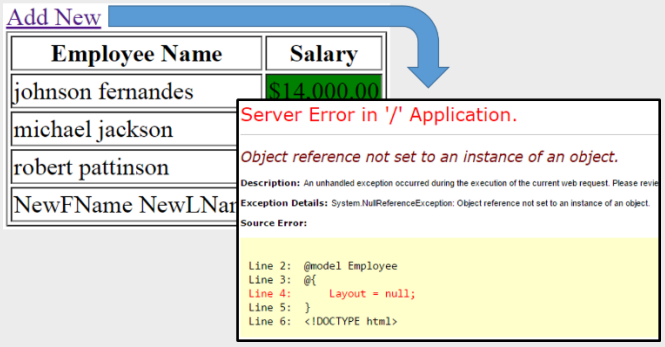
Step 2.2 Display values in corresponding controls from Model

```
...
...
...
<input type="text" id="TxtFName" name="FirstName" value="@Model.FirstName" />
...
...
...
<input type="text" id="TxtLName" name="LastName" value="@Model.LastName" />
...
...
...
<input type="text" id="TxtSalary" name="Salary" value="@Model.Salary" />
...
...
...
```

Step 3 – Execute and Test

Press F5 and execute the application. Navigate to the AddNew screen by clicking "Add New" link.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Reason for above error will be discussed at the end of this lab. Let's implement solution now.

Step 4 – Change AddNew Action method

```
public ActionResult AddNew()
{
   return View("CreateEmployee", new CreateEmployeeViewModel ());
}
```

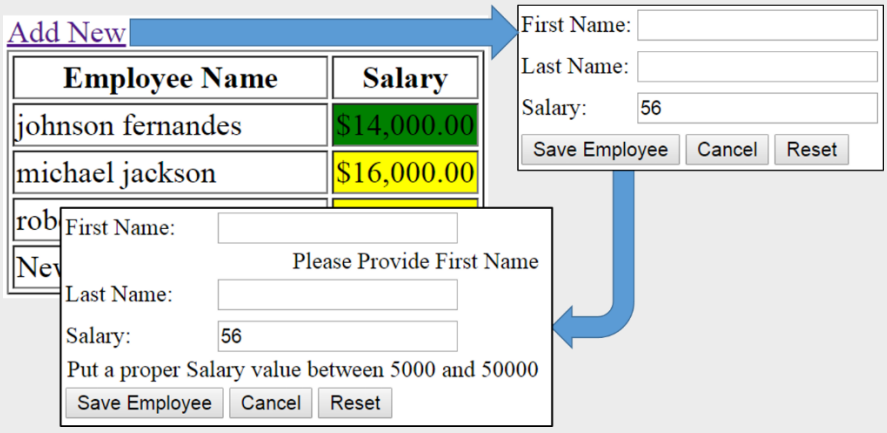
Step 5 – Execute and Test

Press F5 and execute the application.

Test 1

- Navigate to the AddNew screen by clicking "Add New" link.
- Keep First Name Empty
- Put Salary as 56.
- Click "Save Employee" button.

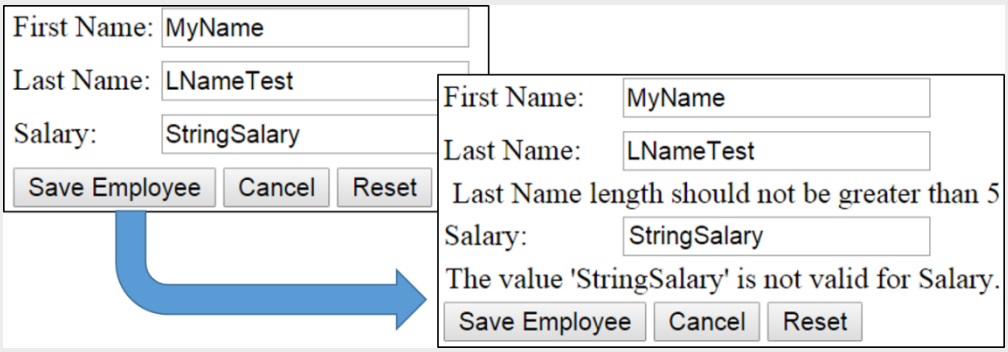
It will make two validations fail



As you can see 56 is maintained in Salary Textbox.

Test 2

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



As you can see FirstName and LastName textbox values are maintained.

Talk on Lab 15

Are we really retaining values here?

No, we are not. Here we are actually repopulating the values from the posted data.

Why it's required to pass "new CreateEmployeeViewModel ()" during initial request that is in "Add New" action method?

In the View we are trying to populate textbox with the values in model.

Example:

```
<input type="text" id="TxtSalary" name="Salary" value="@Model.Salary" />
```

As you can see, in the above code block we are accessing FirstName property of current Model. If Model is null, simply "Object reference not set to an instance of the class" exception will be thrown.

When "Add New" hyperlink is clicked, request will be handled by "Add New" action method. Earlier in this action method we were returning view without passing any data. It means Model property inside view be Null and Null. Something will throw "Object reference not set to an instance of the class". To solve this problems "new CreateEmployeeViewModel ()" was passed during initial request.

Do we have any automated way to achieve same functionality?

Yes, we can use HTML helper classes for that. We will talk about this in one of the upcoming lab.

Lab 16 – Adding Client side validation

First of all let's list down what all validations we require here.

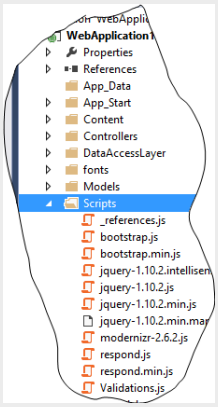
1. FirstName should not be empty.
2. LastName length should not be greater than 5.
3. Salary should not be empty.
4. Salary should be a proper number
5. FirstName should not contain "@" symbol

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Let's see how to implement it.

Step 1 – Create JavaScript validation file

Create a JavaScript File called "Validations.js" and keep it inside Scripts folder



Step 2 – Create Validation functions

In "Validations.js" file create validation functions as follows

```
function IsFirstNameEmpty() {
   if (document.getElementById('TxtFName').value == "") {
      return 'First Name should not be empty';
   }
   else { return ""; }
}

function IsFirstNameInValid() {
   if (document.getElementById('TxtFName').value.indexOf("@") != -1) {
      return 'First Name should not contain @';
   }
   else { return ""; }
}
function IsLastNameInValid() {
   if (document.getElementById('TxtLName').value.length>=5) {
      return 'Last Name should not contain more than 5 character';
   }
   else { return ""; }
}
function IsSalaryEmpty() {
   if (document.getElementById('TxtSalary').value=="") {
      return 'Salary should not be empty';
   }
   else { return ""; }
}
function IsSalaryInValid() {
   if (isNaN(document.getElementById('TxtSalary').value)) {
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
        return 'Enter valid salary';
    }
    else { return ""; }
}
function IsValid() {

    var FirstNameEmptyMessage = IsFirstNameEmpty();
    var FirstNameInValidMessage = IsFirstNameInValid();
    var LastNameInValidMessage = IsLastNameInValid();
    var SalaryEmptyMessage = IsSalaryEmpty();
    var SalaryInvalidMessage = IsSalaryInValid();

    var FinalErrorMessage = "Errors:";
    if (FirstNameEmptyMessage != "")
        FinalErrorMessage += "\n" + FirstNameEmptyMessage;
    if (FirstNameInValidMessage != "")
        FinalErrorMessage += "\n" + FirstNameInValidMessage;
    if (LastNameInValidMessage != "")
        FinalErrorMessage += "\n" + LastNameInValidMessage;
    if (SalaryEmptyMessage != "")
        FinalErrorMessage += "\n" + SalaryEmptyMessage;
    if (SalaryInvalidMessage != "")
        FinalErrorMessage += "\n" + SalaryInvalidMessage;

    if (FinalErrorMessage != "Errors:") {
        alert(FinalErrorMessage);
        return false;
    }
    else {
        return true;
    }
}
```

Step 3- Include Validation file in View

Simple put a reference of "Validations.js" file in the head section of "CreateEmployee" view as follows.

```
<script src="~/Scripts/Validations.js"></script>
```

Step 4 – Attach validations

Simply invoke IsValid function on SaveEmployee button click as follows.

```
<input type="submit" name="BtnSubmit" value="Save Employee" onclick="return IsValid();" />
```

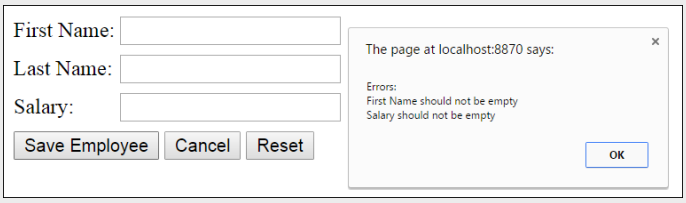
Step 5 – Execute and Test

Press F5 and execute the application

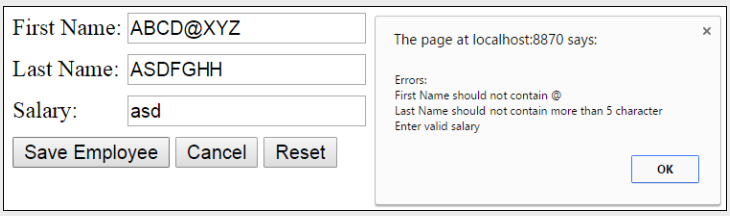
Navigate to the AddNew screen by clicking "Add New" link.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Test 1



Test 2



Talk on Lab 16

Why return keyword is required in onclick of SaveEmployee button click?

As we discussed in Lab 9, submit button will make a request to server when clicked. There is no point on making server request when validation fails. By writing "return false" in the onclick of submit button, we can stop the default server request.

In our case IsValid function will return false when validation fails and thus we achieve desired functionality.

Instead of alert can we show the error messages in the page itself?

Yes we can. Simply create a span tag for each error. Make it invisible (using css) in the beginning and on submit button click if validation fails make it visible using JavaScript.

Is there any way to get this client side validation automatically?

Yes, when we use HTML helpers we get automatic client side validation based on server side validation. We will discuss this in one of the future lab

Does server side validation is required anymore?

Yes, In case someone disables JavaScript, Server side validation keep everything in place.

Lab 17 – Adding Authentication

In this lab we will make our GetView action method secured. We will make sure that only valid user will be able to access the action method.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

In the Day 1 of this series we understood the real meaning of word Asp.net and MVC. We understood that Asp.Net MVC is part of Asp.net. Most of the features of Asp.Net are inherited in Asp.Net MVC. One of the feature is Forms Authentication.

Before we start with lab first let's understand how Forms Authentication works in Asp.Net

1. End user make a request to Forms authentication enabled application with the help of browser.
2. Browser will send all the associated cookies stored in the client machine with request.
3. When request is received as server end, server examines the request and check for the special cookie called "Authentication Cookie".
4. If valid authentication cookie is found, server confirms the identity of the user or in simple words, consider user as a valid user and make him go further.
5. If valid authentication cookie is not found server considers user as anonymous (or unauthenticated) user. In this case if the requested resource is marked as protected/secured user will be redirected to login page.

Step 1 – Create AuthenticationController and Login action method.

Right click controller folder and Select "Add New Controller" and create a controller called "Authentication". In this case full class name will be "AuthenticationController".
Inside controller create and Action method called Login as follows.

```
public class AuthenticationController : Controller
{
   // GET: Authentication
   public ActionResult Login()
   {
      return View();
   }
}
```

Step 2 – Create Model

Create new Model class called UserDetails inside Models folder as follows.

```
namespace WebApplication1.Models
{
   public class UserDetails
   {
      public string UserName { get; set; }
      public string Password { get; set; }
   }
}
```

Step 3 – Create Login View

Crete a new view called Login inside "~/Views/Authentication" folder. Make it a strongly typed view of type UserDetails.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Put following HTML inside View

```
@model WebApplication1.Models.UserDetails
@{
  Layout = null;
}
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>Login</title>
</head>
<body>
  <div>
    @using (Html.BeginForm("DoLogin", "Authentication", FormMethod.Post))
    {
      @Html.LabelFor(c=>c.UserName)
      @Html.TextBoxFor(x=>x.UserName)

      <br />
      @Html.LabelFor(c => c.Password)
      @Html.PasswordFor(x => x.Password)
      <br />

      <input type="submit" name="BtnSubmit" value="Login" />
    }
  </div>
</body>
</html>
```

As you can see, this time for generating View instead of Pure HTML we are using HtmlHelper class.

- In view we will get a readymade object of HtmlHelper class called "Html"
- HtmlHelper class functions simply returns html string.

Example 1:

```
@Html.TextBoxFor(x=>x.UserName)
```

Above code will generate following HTML.

```
<input id="UserName" name="UserName" type="text" value="" />
```

Example 2:

```
@using (Html.BeginForm("DoLogin", "Authentication", FormMethod.Post))
{
}
```

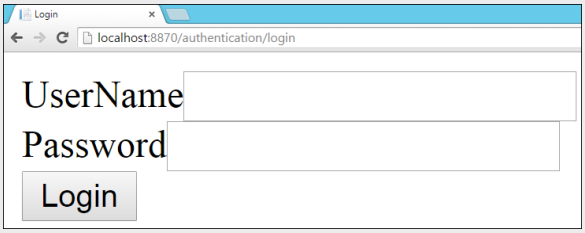
Above code will generate following HTML.

```
<form action="/Authentication/DoLogin" method="post">
</form>
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 4 – Execute and Test

Press F5 and execute the application. In the address put the URL of Login action method. In my case it will be "http://localhost:8870/Authentication/Login".



Step 5 – Enable Forms Authentication

Open Web.config file. Navigate to System.Web section. Find a child section called Authentication. If it won't exist create it. Set Authentication mode to Forms and Login URL to "Login" action method created in step 1.

```
<authentication mode="Forms">
  <forms loginUrl="~/Authentication/Login"></forms>
</authentication>
```

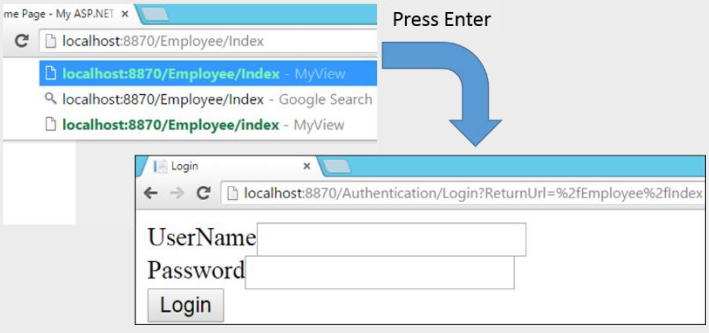
Step 6 – Make action method secured.

Open EmployeeController and attach Authorize attribute to Index action as follows.

```
[Authorize]
public ActionResult Index()
{
   EmployeeListViewModel employeeListViewModel = new EmployeeListViewModel();
.....
```

Step 7 –Execute and Test

Press F5 and execute the application. In the address bar put URL of Index action of EmployeeController. In my case it will be "http://localhost:8870/Employee/Index".



As you can see, request to Index action automatically redirected to login action.

Step 8 – Create business layer functionality

Open EmployeeBusinessLayer class and create a method called IsValidUser as follows.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
public bool IsValidUser(UserDetails u)
{
   if (u.UserName == "Admin" && u.Password == "Admin")
   {
      return true;
   }
   else
   {
      return false;
   }
}
```

Note: In business layer we are comparing username and password with hardcoded values. In real time we can make call to Database layer and compare it with real time values.

Step 9 – Create DoLogin Action method

Open AuthenticationController class and create a new action method called DoLogin.

This DoLogin action method will be invoked when Login button in Login view is clicked (Check Step 3).

Now let's list down the points need to be done in DoLogin

1. Check for validity of user by invoking business layer function.
2. If user is a valid user create an authentication cookie. It makes futures requests authenticated request.
3. If user is invalid, add a new error to current ModelState. This error will be displayed in View.

```
[HttpPost]
public ActionResult DoLogin(UserDetails u)
{
   EmployeeBusinessLayer bal = new EmployeeBusinessLayer();
   if (bal.IsValidUser(u))
   {
      FormsAuthentication.SetAuthCookie(u.UserName, false);
      return RedirectToAction("Index", "Employee");
   }
   else
   {
      ModelState.AddModelError("CredentialError", "Invalid Username or Password");
      return View("Login");
   }
}
```

Let's understand the above code block.

- If you remember in "Day 3 – Lab 13" we spoke about ModelState and understood that it encapsulates current state of the model. It contains all the errors related to current model. In above code snippet we are adding a new error when user is an invalid user (new error with key "CredentialError" and message "Invalid Username or Password").
- FormsAuthentication.SetAuthCookie will create a new cookie in client's machine.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 10– Display message in view

Open Login View and add following code just above the @Html.BeginForm

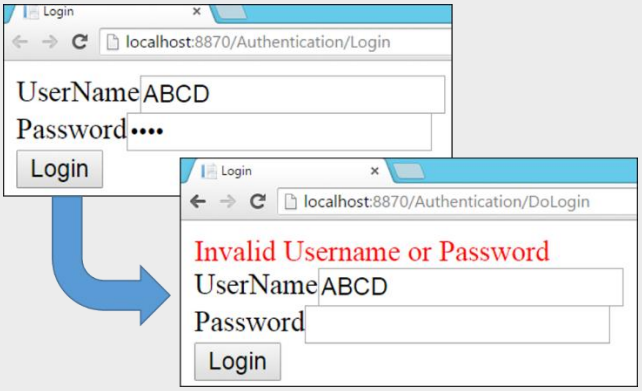
```
@Html.ValidationMessage("CredentialError", new {style="color:red;" })
@using (Html.BeginForm("DoLogin", "Authentication", FormMethod.Post))
{
```

Step 11 – Execute and Test

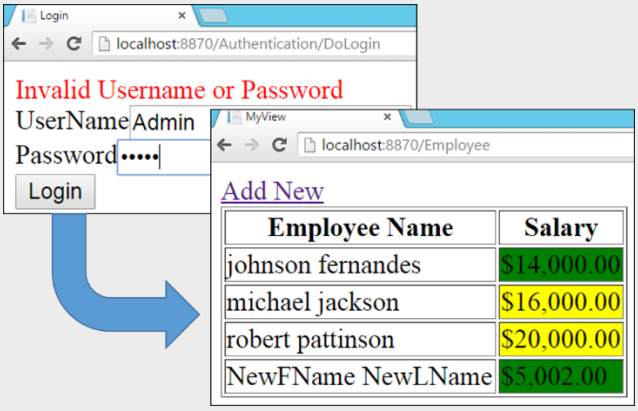
Press F5 and execute the application. Directly make a request to Login Action. I believe you know how to do it now.

Note: If you want you can make request to Index action of EmployeeController but it ultimately redirect you to Login Action.

Test 1



Test 2

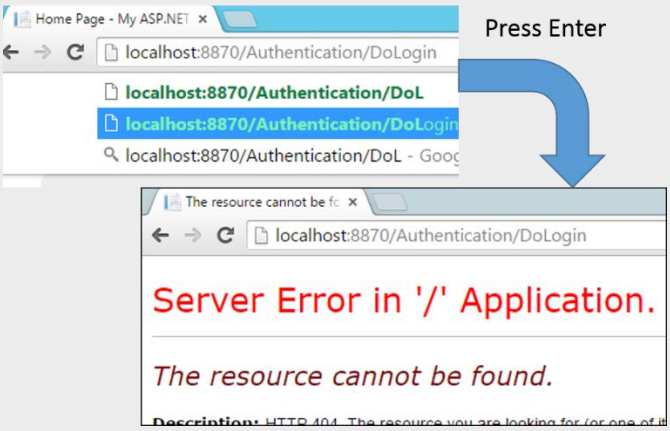


Talk on Lab 17

Why DoLogin is attached with HttpPost attribute?

This attribute makes DoLogin action method open for only Post request. If someone try to make a get request to DoLogin it won't work out.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Do we have more such attributes?

Yes. We have HttpGet,HttpPut and HttpDelete. As a best practice every action method must be attached with such attributes.

Note: To keep coding and learning simple and easy we have not followed such practice everywhere in this series but I recommend you to strictly follow it in your project.

As we move along we will keep on talking about such best practices.

Is it must to write FormsAuthentication.SetAuthCookie?

Yes.

Let's understand a small process.

- Client make a first request to server with the help of browser.
- When request is made via browser, all the associated cookies will be sent with the request.
- Server receives the request and prepares the response.
- Now as we know request-response happens via HTTP protocol and HTTP is a stateless protocol. For server every request will be a new request hence when the same client makes second request server won't recognize it. To solve this issue Server will add a cookie in the prepared response and send back.
- When client's browsers receives the response with cookies, it creates cookies in client's machine.
- Now if client make a request once again server will recognize him/her because request contains cookies.

FormsAuthentication.SetAuthCookie will add a special cookie called "Authentication" cookie to the response.

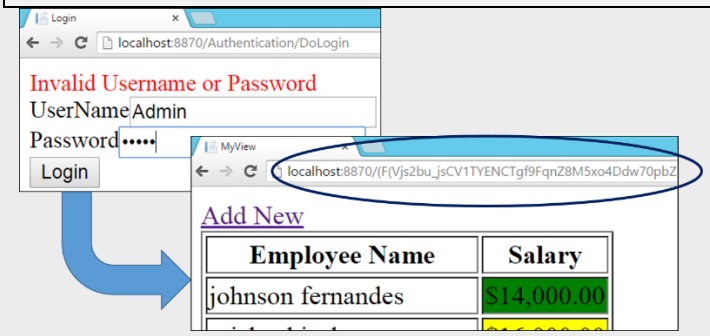
Does it means Forms Authentication won't work without cookies?

No. We have an alternative for it. We can use URI instead of cookies.

Open Web.Config and change "Authentication/Forms" section as follows.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
<forms loginUrl="~/Authentication/Login" cookieless="UseUri"></forms>
```



As you can see, now authentication cookie is passed in URL itself.

By default cookieless property is set to "AutoDetect". It means authentication works via cookies and in case cookies are not supported URL will do the required work.

What does second parameter of FormsAuthentication.SetAuthCookie do?
It will decide whether we want to create a persistent cookie or not. Non persistent cookies will get deleted automatically when browser is closed. Persistent cookies wont deleted automatically. We have to remove it manually either via coding or by using browser settings.

How to do logout via coding?
We will learn it in one of the future lab.

How come the UserName textbox is repopulated when credentials are wrong?
That's the magic of HTML helper classes. They will repopulate the values in the controls from the posted data. This is one of the advantage of using Helper classes.

What does @Html.ValidationMessage does?
We already discussed it in Lab 13 talks. It displays ModelState error based on key.

What does Authorize attribute do?
In Asp.net MVC there is a concept called Filters. Which will be used to filter out requests and response. There are four kind of filters. We will discuss each one of them in our 7 days journey.
Authorize attribute falls under Authorization filter. It will make sure that only authenticated requests are allowed for an action method.

Can we attach both HttpPost and Authorize attribute to same action method?
Yes we can.

Why there is no ViewModel in this example?

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

As per the discussion we had in Lab 6, View should not be connected to Model directly. We must always have ViewModel in between View and Model. It doesn't matter if view is a simple "display view" or "data entry view", it should always connected to ViewModel.
Reason for not using ViewModel in our project is simplicity. In real time project I strongly recommend you to have ViewModel everywhere.

Is it required to attach Authorize attribute to each and every action method?
No. We can attach it Controller level or Global level also.
When attached at controller level, it will be applicable for all the action methods in a controller.
When attached at Global level, it will be applicable for all the action method in all the controllers.

Controller Level

```
[Authorize]
public class EmployeeController : Controller
{
....
```

Global level
Step 1 - Open FilterConfig.cs file from App_start folder.
Step 2 - Add one more line RegisterGlobalFilters as follows.

```
public static void RegisterGlobalFilters(GlobalFilterCollection filters)
{
   filters.Add(new HandleErrorAttribute());//Old line
   filters.Add(new AuthorizeAttribute());//New Line
}
```

Step 3 - Attach AllowAnonymous attribute to Authentication controller.

```
[AllowAnonymous]
public class AuthenticationController : Controller
{
```

Step 4 – Execute and Test the application in the same way we did before.

What does filters.Add(new HandleErrorAttribute()) does?
We will discuss this in detail in future lab.

Why AllowAnonymous attribute is required for AuthenticationController.
We have attached Authorize filter at global level. That means now everything is protected including Login and DoLogin action methods. AllowAnonymous opens action method for non-authenticated requests.

How come this RegisterGlobalFilters method inside FilterConfig class invoked?
It was invoked in Application_Start event written inside Global.asax file.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Does FormsAuthentication is the only way to implement authentication in Asp.Net?

We have more ways to do authentication. We can perform windows authentication or can go with oAuth. In our sample project we have implemented Forms Autentication.

Lab 18 – Display UserName in the view.

In this lab we will display currently logged in User Name in View.

Step 1 – Add UserName in ViewModel

Open EmployeeListViewModel and add a new property called UserName as follows.

```
public class EmployeeListViewModel
{
   public List<EmployeeViewModel> Employees { get; set; }
   public string UserName { get; set; }
}
```

Step 2 – Set value to ViewModel UserName

Open EmployeeController and change Index as follows

```
public ActionResult Index()
{
   EmployeeListViewModel employeeListViewModel = new EmployeeListViewModel();
   employeeListViewModel.UserName = User.Identity.Name; //New Line
.....
```

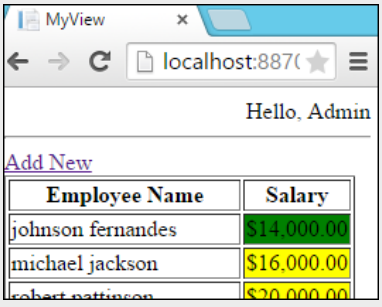
Step 3 – Display UserName in View

Open Index.cshtml view and display UserName as follows.

```
<body>
 <div style="text-align:right"> Hello, @Model.UserName </div>
 <hr />
 <a  href="/Employee/AddNew">Add New</a>
  <div>
    <table border="1">
```

Step 4 – Execute and Test

Press F5 and execute the application. Complete the login process and you will get to see following output.



Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Lab 19 – Implement Logout

Step 1 – Create Logout link

Open Index.cshtml and create Logout link as follows.

```
<body>
   <div style="text-align:right">Hello, @Model.UserName
   <a href="/Authentication/Logout">Logout</a></div>
   <hr />
   <a  href="/Employee/AddNew">Add New</a>
   <div>
      <table border="1">
```

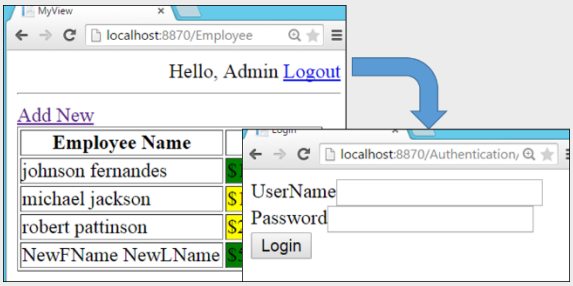
Step 2 – Create Logout Action method.

Open AuthenticationController and add new action method called Logout as follows.

```
public ActionResult Logout()
{
   FormsAuthentication.SignOut();
   return RedirectToAction("Login");
}
```

Step 3 – Execute and Test

Press F5 and execute the application.



Lab 20 – Implementing validation in Login Page

Step 1 – Add data annotations

Open UserDetails.cs and add Data Annotation as follows.

```
public class UserDetails
{

   [StringLength(7,MinimumLength=2, ErrorMessage = "UserName length should be between 2 and 7")]
   public string UserName { get; set; }
   public string Password { get; set; }
}
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 2 – Display error messages in view

Change Login.cshtml to display error messages.

```
@using (Html.BeginForm("DoLogin", "Authentication", FormMethod.Post))
{
   @Html.LabelFor(c=>c.UserName)
   @Html.TextBoxFor(x=>x.UserName)
   @Html.ValidationMessageFor(x=>x.UserName)
.....
```

Note: This time instead of Html.ValidationMessage we have used Html.ValidationMessageFor. Both will do same thing. Html.ValidationMessageFor can be used only when the view is strongly typed view.

Step 3 – Change DoLogin

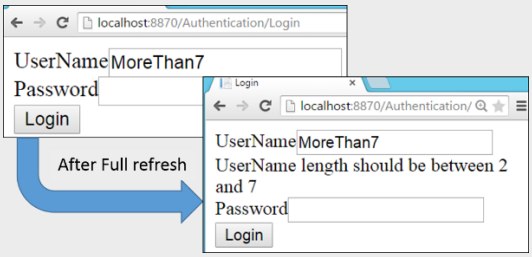
Change DoLogin action method as follows

```
[HttpPost]
public ActionResult DoLogin(UserDetails u)
{
   if (ModelState.IsValid)
   {
      EmployeeBusinessLayer bal = new EmployeeBusinessLayer();
      if (bal.IsValidUser(u))
      {
         FormsAuthentication.SetAuthCookie(u.UserName, false);
         return RedirectToAction("Index", "Employee");
      }
      else
      {
         ModelState.AddModelError("CredentialError", "Invalid Username or Password");
         return View("Login");
      }
   }
   else
   {
      return View("Login");
   }
}
```

Step 4- Execute and Test

Press F5 and execute the application.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



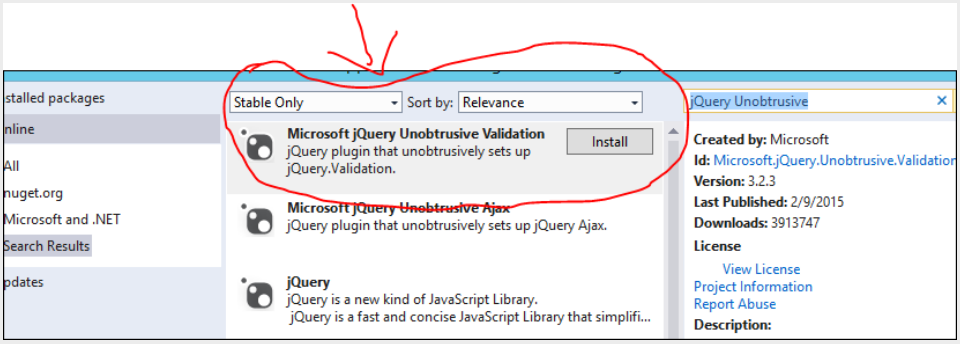
Lab 21 – Implementing Client side validation in Login Page

This time we will do client side validation in a different way.

Step 1 – Download jQuery unobtrusive Validation files.

Right click the project. Select "Manage Nuget packages".

Click on online and search for "jQuery Unobtrusive".



Install "Microsoft jQuery Unobtrusive Valiadtion

Step 2 – Include jQuery Validation files in View

Above steps adds three JavaScript files in Scripts folder.

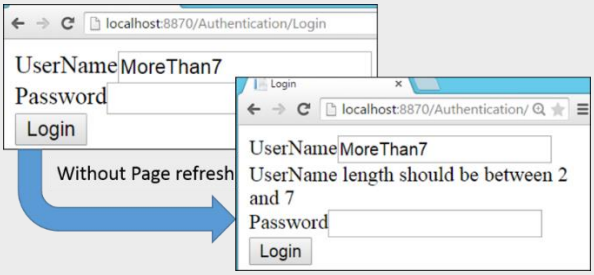
- jQuery-Someversion.js
- jQuery.valiadte.js
- jquery.validate.unobtrusive

Open Login.cshtml and in head section include all three js files (in same sequence)/

```
<script src="~/Scripts/jquery-1.8.0.js"></script>
<script src="~/Scripts/jquery.validate.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.js"></script>
```

Step 3 – Execute and Test.

Press F5 and execute the application.



Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Talk on Lab 21

How come client side validation is implemented?

As you can see, without much effort client side validation is implemented. In Login view, HTML elements are generated using HTML helper classes. Helper functions while generating HTML markup attach some attributes based on the data annotation attributes used.

Example:

```
@Html.TextBoxFor(x=>x.UserName)
@Html.ValidationMessageFor(x=>x.UserName)
```

Above code will generate following HTML.

```
<input data-val="true" data-val-length="UserName length should be between 2 and 7" data-val-length-max="7" data-val-length-min="2" id="UserName" name="UserName" type="text" value="" />
<span class="field-validation-error" data-valmsg-for="UserName" data-valmsg-replace="true"> </span>
```

These custom HTML attributes will be used by "jQuery Unobtrusive validation files" and thus validation get implemented at client side automatically.

Automatic client side validation is the second advantage of Html helper classes.

What is unobtrusive JavaScript means?

This is what Wikipedia says about it.

> Unobtrusive JavaScript is a general approach to the use of JavaScript in web pages. Though the term is not formally defined, its basic principles are generally understood to include:
>
> - Separation of functionality (the "behaviour layer") from a Web page's structure/content and presentation
> - Best practices to avoid the problems of traditional JavaScript programming (such as browser inconsistencies and lack of scalability)
> - Progressive enhancement to support user agents that may not support advanced JavaScript functionality

Let me define it in layman terms.

"Write your JavaScript in such way that, JavaScript won't be tightly connected to HTML. JavaScript may access DOM elements, JavaScript may manipulate DOM elements but won't directly connected to it."

In the above example, jQuery Unobtrusive JavaScript simply used some input element attributes and implemented client side validation.

Can we use these JavaScript validation without HTML helper classes?

Yes, for that we have to manually attach such attributes to elements.

What is more preferred, Html helper functions or pure HTML?

I personally prefer pure HTML because Html helper functions once again take "full control over HTML" away from us and we already discussed the problems with that.

Secondly let's talk about a project where instead of jQuery some other JavaScript frameworks/libraries are used. Some other framework like angular. In that case mostly we think about angular validation and in that case these custom HTML validation attributes will go in vain.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Learn MVC Project in 7 Days – Day 5

Lab 22 – Add Footer

In this lab we will add Footer to Employee Screen. Objective of this lab is understanding Partial Views.

What are "Partial Views"?

Logically, Partial View is a reusable view which will never get displayed directly. It will be included inside other views and get displayed as a part of that view. It resembles user controls in Asp.Net web forms, but without Code behind.

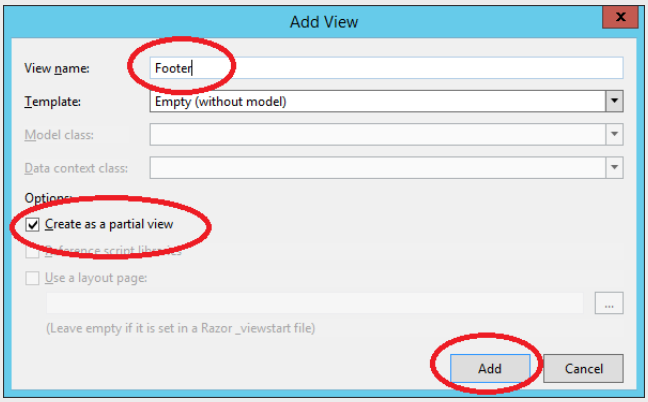
Step 1 – Create ViewModel for Partial View

Right click the ViewModel folder and create a class called FooterViewModel as follows.

```
public class FooterViewModel
{
   public string CompanyName { get; set; }
   public string Year { get; set; }
}
```

Step 2 – Create Partial View

Right click the "~/Views/Shared" folder. Select Add>>View.



Put View name as Footer, Check "Create as a partial view" checkbox and click "Add".

Note: We already spoke about shared folder in Day 1. Shared folder contains views which will not be specific to a particular controller. Views inside Shared folder will be available to all the controllers.

Step 3 – Display data in the Partial View

Open Footer.cshtml and put following HTML in it.

```
@using WebApplication1.ViewModels
@model FooterViewModel
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
<div style="text-align:right;background-color: silver;color: darkcyan;border: 1px solid gray;margin-top:2px;padding-right:10px;">
  @Model.CompanyName © @Model.Year
</div>
```

Step 3 – Include Footer data in Main ViewModel

Open EmployeeListViewModel class and add a new property to hold Footer data as follows.

```
public class EmployeeListViewModel
{
   public List<EmployeeViewModel> Employees { get; set; }
   public string UserName { get; set; }

   public FooterViewModel Footer Data { get; set; }//New Property
}
```

In our example Footer (Partial View) is going to be displayed as a part of Index View.

We will pass necessary data to Footer from Index View.

Index View is a strongly typed view of type EmployeeListViewModel and hence all the data required for Footer view should be encapsulated in EmployeeListViewModel.

Step 4 – Set Footer Data

Open EmployeeController and in Index action method set value to Footer Data property as follows.

```
public ActionResult Index()
{
  ...
  ...
   employeeListViewModel.Footer Data = new FooterViewModel();
   employeeListViewModel.Footer Data.CompanyName = "StepByStepSchools";
   employeeListViewModel.Footer Data.Year = DateTime.Now.Year.ToString();
   return View("Index", employeeListViewModel);
}
```

Step 5 – Display Footer

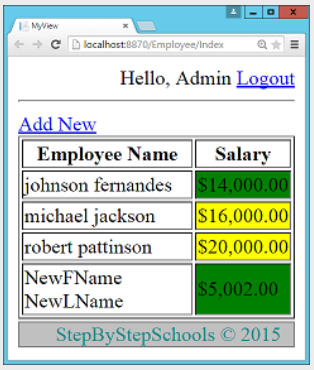
Open Index.cshtml and display Footer Partial View after table tag as follows.

```
    </table>
     @{
        Html.RenderPartial("Footer", Model.Footer Data);
     }
  </div>
</body>
</html>
```

Step 6 – Execute and Test

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Press F5. Navigate to Index view. (I believe, you know how to do it now.☺)



Talk on Lab 22

What does Html.Partial do?

Just like Html.RenderPartial, Html.Partial will be used to display Partial View in the View.

This is the syntax

```
@Html.Partial("Footer", Model.Footer Data);
```

Syntax is much simpler than earlier one.

What's the difference between two?

Html.RenderPartial will write result of the Partial View directly to the HTTP response stream whereas Html.Partial will return result as MvcHtmlString.

What is MvcHtmlString and why does Html.Partial return MvcHtmlString instead of string?

First let's understand what is MvcHtmlString?

As per MSDN "MvcHtmlString represents a HTML-encoded string that should not be encoded again".

Let's make this definition easier. Look at the following code.

```
@{
        string MyString = "<b>My Simple String</b>";
}
@MyString
```

It will generate following output

My Simple String

As you can see, razor displayed whole content as it is. Many people might have thought of seeing a bold string but Razor Html encoded the content before displaying and that's why instead of bold string we got pure content.

We use MvcHtmlString when we don't want razor to do the encoding. MvcHtmlString is an indication to razor that "string is already encoded, no more encoding is required".

For example look at the following code.

```
@{
   string MyString = "<b>My Simple String</b>";
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
}
@MvcHtmlString.Create(MyString)
```

It will generate following output

My Simple String

Why does Html.Partial return MvcHtmlString instead of string?

We already understood a fact that "razor will always encode strings but it never encodes MvcHtmlString". It doesn't make sense if Partial View contents are considered as pure string gets displayed as it is. We want it to be considered as a HTML content and for that we have to stop razor from encoding thus Partial method is designed to return MvcHtmlString.

What is recommended Html.RenderPartial or Html.Partial?

Html.RenderPartial is recommended because it is faster.

When Html.Partial will be preferred?

It is recommended when we want to change the result returned by Partial View before displaying.

Open Index.cshtml and open Footer code to below code and test.

```
@{
   MvcHtmlString result = Html.Partial ("Footer", Model.Footer Data);
   string finalResult = result.ToHtmlString().Replace("2015", "20000");
}
@MvcHtmlString.Create(finalResult)
```

Now footer will look like below.

StepByStepSchools © 20000

Why Partial View is placed inside Shared Folder?

Partial Views are meant for reusability hence the best place for them is Shared folder.

Can't we place Partial Views inside a specific controller folder, like Employee or Authentication?

We can do that but in that case it won't be available to only specific controller.

Example: When we keep Partial View inside Employee folder it won't be available for AuthenticationController or to Views related to AuthenticationController.

Why definition of Partial View contains word "Logically"?

In definition we have said that Partial View is a reusable view but it won't get executed by its own. It has to be placed in some other view and then displayed as a part of the view.

What we said about reusability is completely true but what we said about execution is only true logically. Technically it's not a correct statement. We can create an action method which will return a ViewResult as bellow.

```
public ActionResult MyFooter()
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
{
   FooterViewModel Footer Data = new FooterViewModel();
   Footer Data.CompanyName = "StepByStepSchools";//Can be set to dynamic value
   Footer Data.Year = DateTime.Now.Year.ToString();
   return View("Footer", Footer Data);
}
```

It will display following output

Although logically it doesn't make sense, technically it's possible. Footer.cshtml won't contain properly structured HTML. It meant to be displayed as a part of some other view. Hence I said "Logically it doesn't make sense".

Why Partial View is created instead of putting footer contents directly in the view?

Two advantages

1. Reusability – we can reuse the same Partial View in some other View.
2. Code Maintenance – Putting it in a separate file makes it easy to manage and manipulate.

Why Header is not created as Partial View?

As a best practice we must create Partial View for header also but to keep Initial labs simpler we had kept it inline.

Lab 23 – Implementing Role based security

In this lab we will implement Admin and Non-Admin login feature.

Requirement is very simple.

"Non Admin user won't be able to create new Employees".

With this lab we will understand two more topics in MVC.

- Session
- Action Filters

Let's start with our lab. 1

To make it simple lets break it into 2 parts.

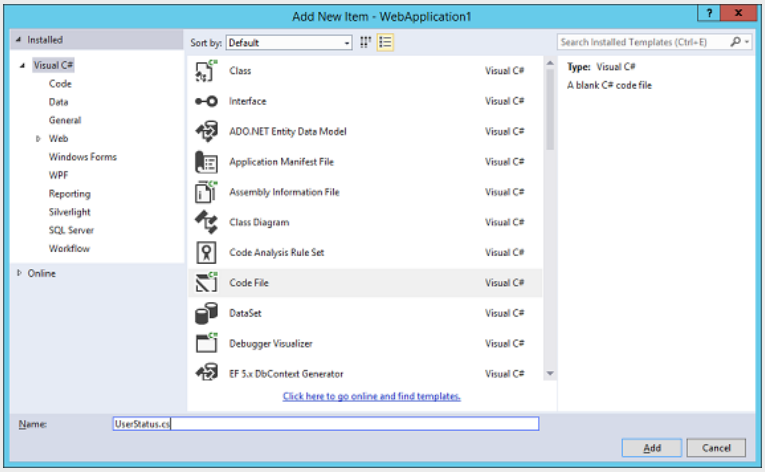
Part 1 – Hide AddNew link from Non-Admin User

Step 1 – Create Enum for identifying UserStatus

Right click the Models folder and select "Add New Item".

Select "Code File" option from the dialog box.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Put Name as "UserStatus" and click Add.
"Code File" option will create a blank ".cs" file.

Create an enum called UserStatus inside it as follows.

```
namespace WebApplication1.Models
{
   public enum UserStatus
   {
      AuthenticatedAdmin,
      AuthentucatedUser,
      NonAuthenticatedUser
   }
}
```

Step 2 – Change business layer functionality

Delete IsValidUser function and create a new function called GetUserValidity as follows.

```
public UserStatus GetUserValidity(UserDetails u)
{
   if (u.UserName == "Admin" && u.Password == "Admin")
   {
      return UserStatus.AuthenticatedAdmin;
   }
   else if (u.UserName == "Sukesh" && u.Password == "Sukesh")
   {
      return UserStatus.AuthentucatedUser;
   }
   else
   {
      return UserStatus.NonAuthenticatedUser;
   }
}
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 3 – Change DoLogin action method

Open AuthenticationController and change DoLogin action method as follows.

```
[HttpPost]
public ActionResult DoLogin(UserDetails u)
{
   if (ModelState.IsValid)
   {
      EmployeeBusinessLayer bal = new EmployeeBusinessLayer();

      //New Code Start
      UserStatus status = bal.GetUserValidity(u);
      bool IsAdmin = false;
      if (status==UserStatus.AuthenticatedAdmin)
      {
         IsAdmin = true;
      }
      else if (status == UserStatus.AuthentucatedUser)
      {
         IsAdmin = false;
      }
      else
      {
         ModelState.AddModelError("CredentialError", "Invalid Username or Password");
         return View("Login");
      }
      FormsAuthentication.SetAuthCookie(u.UserName, false);
      Session["IsAdmin"] = IsAdmin;
      return RedirectToAction("Index", "Employee");

      //New Code End
   }
   else
   {
      return View("Login");
   }
}
```

As you can see, we are using session variable for identifying whether the user is a admin user or non admin user.

Don't know Session?

Session is the feature of Asp.Net which is reused in Asp.Net MVC.

We use Session variables to hold user related data. Life of session variable is going to be life of a user. It will be available until current session ends.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

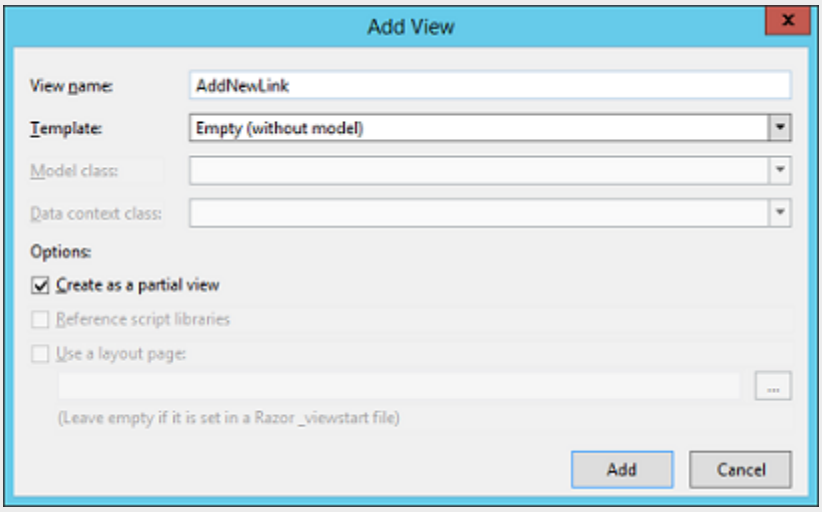
Step 4 – Remove Existing AddNew link

Open Index.cshtml view from "~/Views/Employee" folder and remove "Add New" hyperlink completely.

```
<!-- Remove following line from Index.cshtml -->
<a  href="/Employee/AddNew">Add New</a>
```

Step 5 – Crete partial view

Right click the "~/Views/Employee" folder and select Add>>View. Set View Name to "AddNewLink" and make sure to check "Create a partial View" checkbox.



Step 6 – Put contents in Partial View

Simply put following contents inside newly created Partial View.

```
<a  href="/Employee/AddNew">Add New</a>
```

Step 7 – Create Action Method

Open EmployeeController and create a new action method called "GetAddNewLink" as follows.

```
public ActionResult GetAddNewLink()
{
   if (Convert.ToBoolean(Session["IsAdmin"]))
   {
      return Partial View("AddNewLink");
   }
   else
   {
      return new EmptyResult();
   }
}
```

Step 8 – Display AddNew link

Open Index.html and simply put following code inside it.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

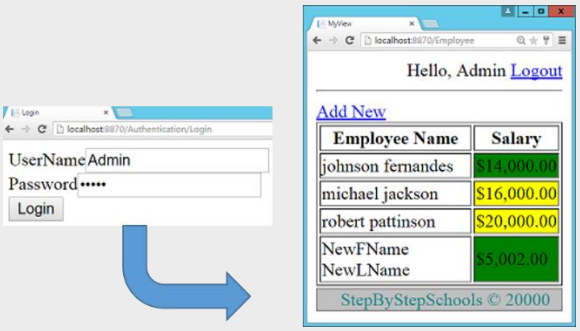
```
<a href="/Authentication/Logout">Logout</a>
</div>
<hr />
@{
   Html.RenderAction("GetAddNewLink");
}
<div>
   <table border="1">
      <tr>
```

Html.RenderAction executes the Action Method and writes result directly to response stream.

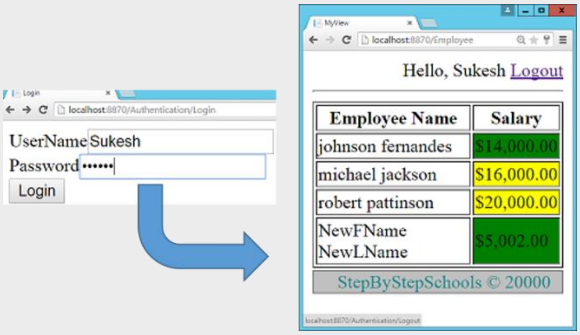
Step 9 – Execute and Test

Press F5 and execute the application.

Test 1



Test 2



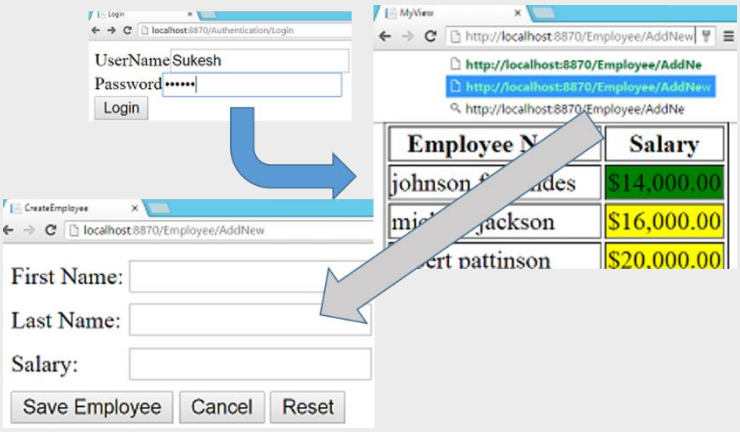
Part 2 – Direct URL security

With above logic one thing is guaranteed. Now a Non-Admin User won't be able to navigate to AddNew action via hyperlink.

Is it enough?

No, It not enough. What if a Non-Admin user directly try to navigate to AddNew action via URL.

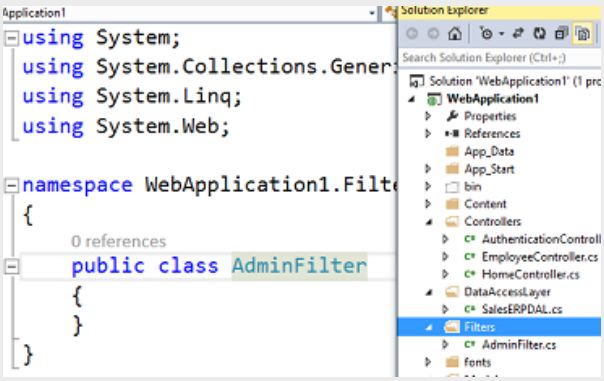
Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



As you can see in the above example, a Non-Admin user is able to access the AddNew action.
To solve this problem we will use MVC ActionFilters. Action Filters let us add some pre-processing and post-processing logic to our action methods. In this lab we will look after pre-processing support of ActionFilters and in coming up lab we will look into post-processing functionality as well.

Step 1 – Setup Filter.

Create a new folder called Filters in Project and create a new class called AdminFilter.



Step 2 – Create Filter

Upgrade simple AdminFilter class to ActionFilter by inheriting it from ActionFilterAttribute class as follows.

```
public class AdminFilter:ActionFilterAttribute
{
}
```

Note: To use ActionFilterAttribute you have to put using System.Web.Mvc in the top.

Step 3 – Add Security validation logic

Inside ActionFilter override OnActionExecuting as follows.

```
public override void OnActionExecuting(ActionExecutingContext filterContext)
{
   if (!Convert.ToBoolean(filterContext.HttpContext.Session["IsAdmin"]))
   {
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
        filterContext.Result = new ContentResult()
        {
            Content="Unauthorized to access specified resource."
        };
    }
}
```

OnActionExecuting will executes before action method executes. Here we can validate request for some special conditions and may stop an action method from being execute.
In our example, ActionFilter makes sure that a request made by Non-Admin user leads to ContentResult.

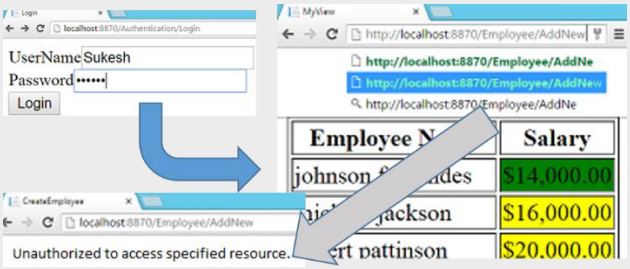
Step 4 – Attach Filter

Attach filter to both AddNew and SaveEmployee Action Method as follows.

```
[AdminFilter]
public ActionResult AddNew()
{
   return View("CreateEmployee",new Employee());
}
...
...
[AdminFilter]
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
   switch (BtnSubmit)
   {
      case "Save Employee":
         if (ModelState.IsValid)
         {
            EmployeeBusinessLayer empBal = new EmployeeBusinessLayer();
....
....
```

Step 5 – Execute and Test

Press F5 and execute the application. Login using Non-Admin credentials and try to navigate to AddNew action method by putting URL of AddNew action.



As you can see, now your Action methods are completely secured.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Note: Whatever strategy and logic we have used in this lab for implementing Role based security may not be the best solution. You may have some better logic to implement such behaviour. It's just one of the way to achieve it.

Talk on Lab 23

Can we invoke GetAddNewLink directly via browser Address bar?

Yes, we already spoke about such behavior in "Talk on Lab 22" section.

Is it possible to stop direct execution of GetAddNewLink?

For that decorate GetAddNewLink with ChildActionOnly attribute.

```
[ChildActionOnly]
public ActionResult GetAddNewLink()
{
   if (Convert.ToBoolean(Session["IsAdmin"]))
   {
```

What does Html.Action do?

Just like Html.RenderAction, Html.Action will execute the action method and display the result in View.

This is the syntax

```
@Html.Action("GetAddNewLink ");
```

Syntax is much simpler than earlier one.

What's the difference between two?

Html.RenderAction will write result of the action method execution directly to the HTTP response stream whereas Html.Action will return result as MvcHtmlString.

What is recommended Html.RenderAction or Html.Action?

Html.RenderAction is recommended because it is faster.

When Html.Action will be preferred?

It is recommended when we want to change the result returned by action method execution before displaying.

What is ActionFilter?

Just like AuthorizationFilter ActionFilter is kind of Filter in Asp.Net MVC. It allows us to add pre-processing and post-processing logic to action method execution.

Note: After each lab we are trying our best to cover each and every question a developer may come up. In case you believe some more questions need to be included please free to send your questions to SukeshMarla@Gmail.com

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Lab 24 - Assignment Lab – Handle CSRF attack

From safety point of view we must also handle CSRF attacks to the project. This one I will leave to you guys.

I recommend you to read this article and implement same to our SaveEmployee action method.

http://www.codeproject.com/Articles/994759/What-is-CSRF-attack-and-how-can-we-prevent-the-sam

Lab 25 – Implement Consistent look across project

In Asp.Net world consistent layout means MasterPage.

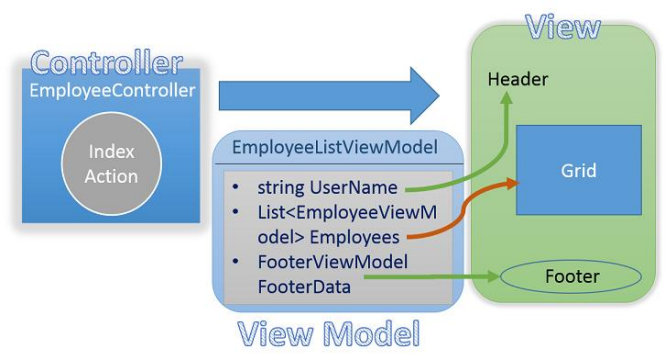
Asp.Net MVC is not different from it. In Razor Master Pages are called as Layout Pages.

Before we go and start with our lab, first let's discuss what all things we have to place in Layout Page

1. Header with Welcome Message
2. Footer with Footer Data

Biggest problem?

Data for Footer and Header is passed to view from Controller as a part of ViewModel.



Now the big question is, how data will be passed from View to Layout Page after header and footer are moved to Layout Page.

Solution – Inheritance

We can simply follow the Object oriented Inheritance principle here. Let's understand it with a lab.

Step 1 – Create Base class ViewModel

Create a new ViewModel called BaseViewModel class in ViewModel folder as follows.

```
public class BaseViewModel
{
   public string UserName { get; set; }
   public FooterViewModel Footer Data { get; set; }//New Property
}
```

As you can see BaseViewModel encapsulates everything required for Layout Page

Step 2 – Prepare EmployeeListViewModel

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Remove UserName and Footer Data properties from EmployeeListViewModel class and inherit it from BaseViewModel.

```
public class EmployeeListViewModel:BaseViewModel
{
   public List<EmployeeViewModel> Employees { get; set; }
}
```

Step 3 – Create Layout Page

Right click shared folder. Select Add>>MVC 5 Layout Page. Put name as MyLayout and click ok.
It will create a structure like below.

```
<!DOCTYPE html>

<html>
<head>
   <meta name="viewport" content="width=device-width" />
   <title>@ViewBag.Title</title>
</head>
<body>
   <div>
      @RenderBody()
   </div>
</body>
</html>
```

Step 4 – Convert Layout to strongly typed Layout

Simply put following statements in the top of the layout page and make it strongly typed layout.

```
@using WebApplication1.ViewModels
@model BaseViewModel
```

Step 5 – Design Layout Page

In the layout page add header, footer and three sections for contents as below.

```
<html>
<head>
   <meta name="viewport" content="width=device-width" />
   <title>@RenderSection("TitleSection")</title>
   @RenderSection("HeaderSection",false)
</head>
<body>
   <div style="text-align:right">
      Hello, @Model.UserName
      <a href="/Authentication/Logout">Logout</a>
   </div>
   <hr />
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
  <div>
  @RenderSection("ContentBody")
  </div>
  @Html.Partial("Footer",Model.Footer Data)
</body>
</html>
```

As you can see we have created three sections in the layout page. TitleSection, HeaderSection and ContentBody. Content pages will use these sections for defining appropriate contents.

Note: While defining HeaderSection second parameter is passed. This parameter decides whether it's the optional section or compulsory section. False indicates it's an optional section.

Step 6 – Attach Layout Page to Index View

Open Index.cshtml and in the top you will find following code.

```
@{
  Layout = null;
}
```

Change it to below code.

```
@{
  Layout = "~/Views/Shared/MyLayout.cshtml";
}
```

Step 7 – Design Index View

1. Take out Headers and Footers from Index View.
2. Copy remaining contents in the body tag and keep it somewhere
3. Now copy contents of title tag.
4. Remove all the HTML contents from the view. Make sure you just remove HTML, @model and layout statement shouldn't be touched.
5. Define TitleSection and Contentbody with the contents copied earlier.

Complete View will look like below.

```
@using WebApplication1.ViewModels
@model EmployeeListViewModel
@{
  Layout = "~/Views/Shared/MyLayout.cshtml";
}

@section TitleSection{
  MyView
}
@section ContentBody{
  <div>
    @{
```

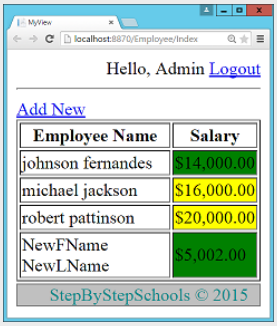
Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
        Html.RenderAction("GetAddNewLink");
    }
    <table border="1">
        <tr>
            <th>Employee Name</th>
            <th>Salary</th>
        </tr>
        @foreach (EmployeeViewModel item in Model.Employees)
        {
            <tr>
                <td>@item.EmployeeName</td>
                <td style="background-color:@item.SalaryColor">@item.Salary</td>
            </tr>
        }
    </table>
  </div>
}
```

As you can see, everything in the view is defined inside some section.

Step 8 – Execute and Test

Press F5 and execute the application. Navigate to Index action.



Step 9 – Attach Layout Page to CreateEmployee View

Open Index.cshtml and in the top you will find following code.

```
@{
  Layout = null;
}
```

Change it to below code.

```
@{
  Layout = "~/Views/Shared/MyLayout.cshtml";
}
```

Step 10 – Design CreateEmployee View

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Follow the same Step 7 procedure and define sections in CreateEmployee View. This time there will be one addition. We will also define HeaderSection.

Complete HTML look like below.

```
@using WebApplication1.Models
@model Employee
@{
  Layout = "~/Views/Shared/MyLayout.cshtml";
}

@section TitleSection{
  CreateEmployee
}

@section HeaderSection{
<script src="~/Scripts/Validations.js"></script>
<script>
  function ResetForm() {
    document.getElementById('TxtFName').value = "";
    document.getElementById('TxtLName').value = "";
    document.getElementById('TxtSalary').value = "";
  }
</script>
}
@section ContentBody{
  <div>
    <form action="/Employee/SaveEmployee" method="post" id="EmployeeForm">
      <table>
      <tr>
        <td>
          First Name:
        </td>
        <td>
          <input type="text" id="TxtFName" name="FirstName" value="@Model.FirstName" />
        </td>
      </tr>
      <tr>
        <td colspan="2" align="right">
          @Html.ValidationMessage("FirstName")
        </td>
      </tr>
      <tr>
        <td>
          Last Name:
```

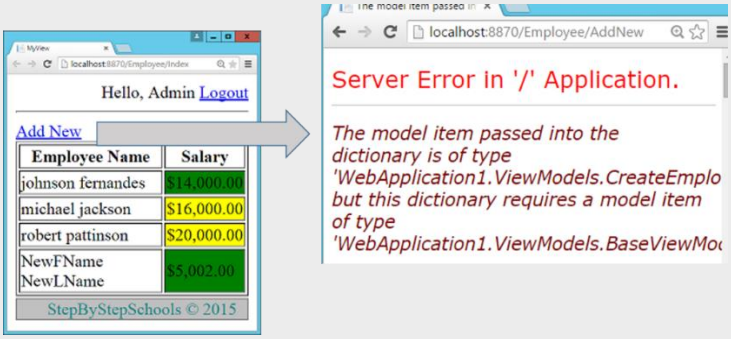
Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
            </td>
            <td>
                <input type="text" id="TxtLName" name="LastName" value="@Model.LastName" />
            </td>
        </tr>
        <tr>
            <td colspan="2" align="right">
                @Html.ValidationMessage("LastName")
            </td>
        </tr>

        <tr>
            <td>
                Salary:
            </td>
            <td>
                <input type="text" id="TxtSalary" name="Salary" value="@Model.Salary" />
            </td>
        </tr>
        <tr>
            <td colspan="2" align="right">
                @Html.ValidationMessage("Salary")
            </td>
        </tr>

        <tr>
            <td colspan="2">

                <input type="submit" name="BtnSubmit" value="Save Employee" onclick="return IsValid();"
/>
                <input type="submit" name="BtnSubmit" value="Cancel" />
                <input type="button" name="BtnReset" value="Reset" onclick="ResetForm();" />
            </td>
        </tr>
        </table>
    </div>
}
```

Step 11 – Execute and Test

Press F5 and execute the application and then try to navigate to AddNew Action via hyperlink.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Index View was a strongly typed view of type EmployeeListViewModel which is a child of BaseViewModel and that's why it's worked. CreateEmployee View is a strongly typed view of type CreateEmployeeViewModel and it's not the child of BaseViewModel and hence such error occurred.

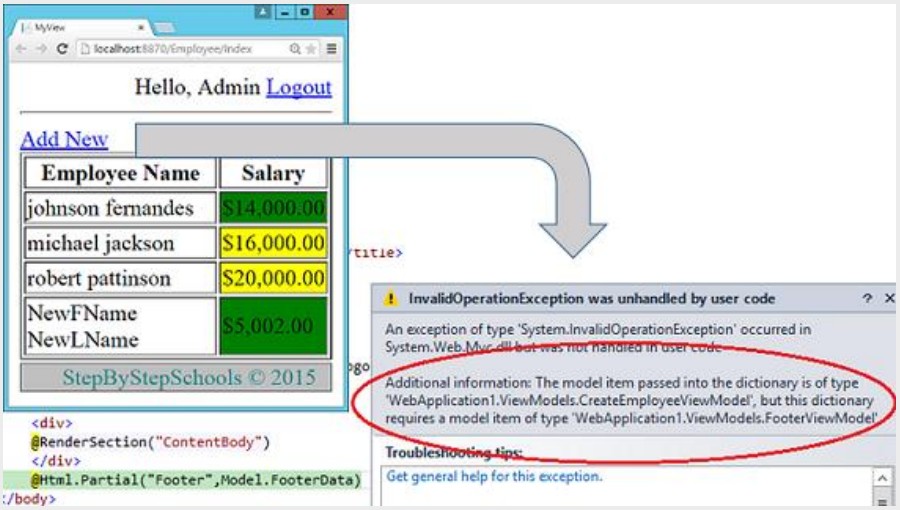
Step 12– Prepare CreateEmployeeViewModel

Inherit CreateEmployeeViewModel from BaseViewModel as follow

```
public class CreateEmployeeViewModel:BaseViewModel
{
...
```

Step 13 – Execute and Test

Do the testing once again.



This error looks very different than it is really ☺

Real reason for such error is, we have not initialized Header and Footer Data in AddNew action.

Step 14 – Initialize Header and Footer Data

Change AddNew action method code to following.

```
public ActionResult AddNew()
{
   CreateEmployeeViewModel employeeListViewModel = new CreateEmployeeViewModel();
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
    employeeListViewModel.Footer Data = new FooterViewModel();
    employeeListViewModel.Footer Data.CompanyName = "StepByStepSchools";//Can be set to dynamic
value
    employeeListViewModel.Footer Data.Year = DateTime.Now.Year.ToString();
    employeeListViewModel.UserName = User.Identity.Name; //New Line
    return View("CreateEmployee", employeeListViewModel);
}
```

Step 15 – Initialize Header and Footer Data in SaveEmployee

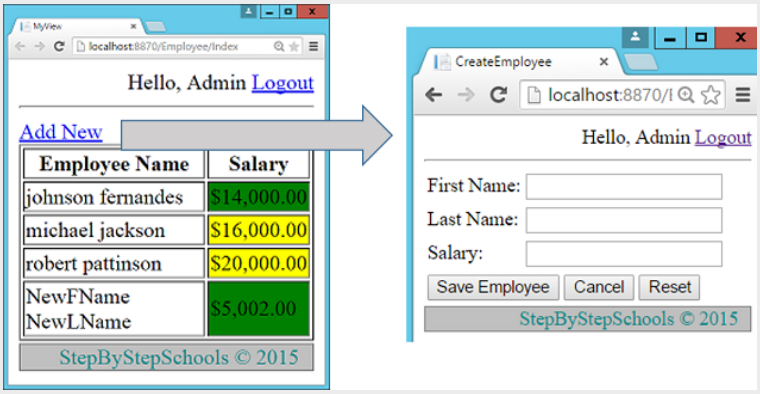
Do the same thing in SaveEmployee action method.

```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
    switch (BtnSubmit)
    {
        case "Save Employee":
            if (ModelState.IsValid)
            {
                ...
            }
            else
            {
                CreateEmployeeViewModel vm = new CreateEmployeeViewModel();
                ...
                vm.Footer Data = new FooterViewModel();
                vm.Footer Data.CompanyName = "StepByStepSchools";//Can be set to dynamic value
                vm.Footer Data.Year = DateTime.Now.Year.ToString();
                vm.UserName = User.Identity.Name; //New Line
                return View("CreateEmployee", vm); // Day 4 Change - Passing e here
            }
        case "Cancel":
            return RedirectToAction("Index");
    }
    return new EmptyResult();
}
```

Step 16 – Execute and Test

Press F5 and execute the application.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Talk on Lab 25

What does RenderBody do?

When we first created in the Layout page it had one razor stamen something like below.

```
@Html.RenderBody()
```

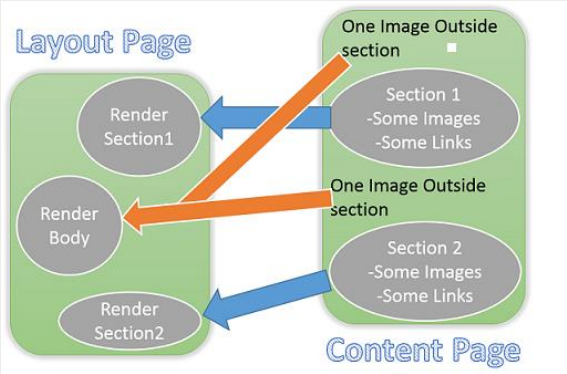
Later we removed it and defined sections manually. Let's understand what does it do?

In content pages we normally define sections which are declared in the Layout Page.

But a strange thing is, Razor allow us to define some contents outside the section too.

All the non-section contents in content page will be rendered by RenderBody function

Below image explains it better.



Can we have nested layouts?

Yes we can. We can create a layout page which will use some other layout page. Syntax will be same.

Is it required to specify layout page in each and every view?

You will find a special view called __ViewStart.cshtml inside Views folder. Setting defined inside this will get applied to all Views.

Example – simply put following code in __ViewStart.cshtml and it will set layout page to all other views.

```
@{
   Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Is it required to put Header and Footer Data code in each and every Action Method?

No, it's not required. We will make it reusable in next lab.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Is it compulsory to define all the sections in child view?

Yes if section is declared as a required section. Default value will be true.

```
@RenderSection("HeaderSection",false) // Not required
@RenderSection("HeaderSection",true) // required
@RenderSection("HeaderSection") // required
```

Lab 26 – Making Header and Footer Data code more efficient with Action Filter

In Lab 23 we had seen one advantage of ActionFilter now it's time for second.

Step 1 – Remove redundant code from Action Methods

Remove Header and Footer Data code from Index, AddNew and SaveEmployee methods (in the Employee Controller).

For the reference, Header Code will look like this

```
bvm.UserName = HttpContext.Current.User.Identity.Name;
```

Footer Code will look like this

```
bvm.Footer Data = new FooterViewModel();
bvm.Footer Data.CompanyName = "StepByStepSchools";//Can be set to dynamic value
bvm.Footer Data.Year = DateTime.Now.Year.ToString();
```

Step 2 – Create HeaderFooterFilter

Create a new class called HeaderFooterFilter in Filters folder and upgrade it to Action filter by inheriting it from ActionFilterAttribute class

Step 2 - Upgrade ViewModel

Override OnActionExecuted in HeaderFooterFilter class. In this method get the current view model and attach Header and Footer Data.

```
public class HeaderFooterFilter : ActionFilterAttribute
{
   public override void OnActionExecuted(ActionExecutedContext filterContext)
   {
      ViewResult v = filterContext.Result as ViewResult;
      if(v!=null) // v will null when v is not a ViewResult
      {
            BaseViewModel bvm = v.Model as BaseViewModel;
            if(bvm!=null)// bvm will be null when we want a view without Header and footer
            {
                  bvm.UserName = HttpContext.Current.User.Identity.Name;
                  bvm.Footer Data = new FooterViewModel();
                  bvm.Footer Data.CompanyName = "StepByStepSchools";//Can be set to dynamic value
                  bvm.Footer Data.Year = DateTime.Now.Year.ToString();
            }
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
        }
    }
}
```

OnActionExecuted will be used to add post processing logic to action method execution.

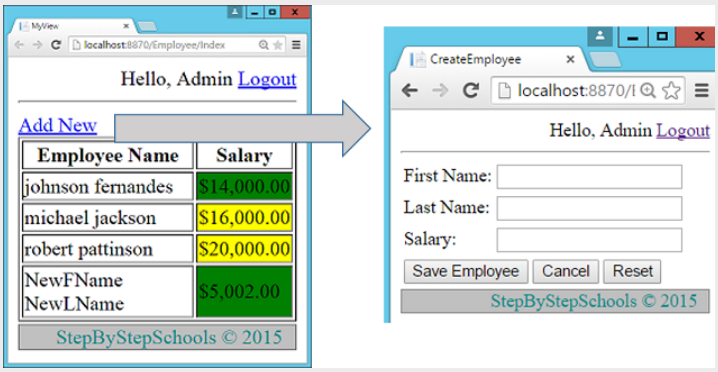
Step 4 – Attach Filter

Attach HeaderFooterFilter to Index, AddNew and SaveEmployee action methods.

```
[HeaderFooterFilter]
public ActionResult Index()
{
    EmployeeListViewModel employeeListViewModel = new EmployeeListViewModel();
...
}
...
[AdminFilter]
[HeaderFooterFilter]
public ActionResult AddNew()
{
    CreateEmployeeViewModel employeeListViewModel = new CreateEmployeeViewModel();
    //employeeListViewModel.Footer Data = new FooterViewModel();
    //employeeListViewModel.Footer Data.CompanyName = "StepByStepSchools";
...
}
...
[AdminFilter]
[HeaderFooterFilter]
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
    switch (BtnSubmit)
    {
        ...
```

Step 5 – Execute and Test

Press F5 and execute the application.



Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Learn MVC Project in 7 Days – Day 6

Lab 27 – Add Bulk upload option

In this lab we will create an option for uploading multiple employees from a CSV file.
We will learn two new things here.

1. How to use File upload control
2. Asynchronous controllers.

Step 1 – Create FileUploadViewModel

Create a new class called FileUploadViewModel in ViewModels folder as follows.

```
public class FileUploadViewModel: BaseViewModel
{
   public HttpPostedFileBase fileUpload {get; set ;}
}
```

HttpPostedFileBase will provide the access to the uploaded file by client.

Step 2- Create BulkUploadController and Index Action

Create a new controller called BulkUploadController and an action method called Index Action as follows.

```
public class BulkUploadController : Controller
{
      [HeaderFooterFilter]
      [AdminFilter]
      public ActionResult Index()
      {
         return View(new FileUploadViewModel());
      }
}
```

As you can see, Index action is attached with HeaderFooterFilter and AdminFilter attributes. HeaderFooterFilter makes sure that correct header and footer data is passed to the ViewModel and AdminFilter restricts access to action method by Non-Admin user.

Step 3 – Create Upload View

Create a view for above action method.
Kindly note that view name should be index.cshtml and should be placed in "~/Views/BulkUpload" folder.

Step 4 – Design Upload View

Put following contents inside the view.

```
@using WebApplication1.ViewModels
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
@model FileUploadViewModel
@{
   Layout = "~/Views/Shared/MyLayout.cshtml";
}

@section TitleSection{
   Bulk Upload
}
@section ContentBody{
   <div>
   <a href="/Employee/Index">Back</a>
      <form action="/BulkUpload/Upload" method="post" enctype="multipart/form-data">
         Select File : <input type="file" name="fileUpload" value="" />
         <input type="submit" name="name" value="Upload" />
      </form>
   </div>
}
```

As you can see name of the property in FileUploadViewModel and name of the input [type="file"] are same. It is "fileUpload". We spoke on the importance of name attribute in Model Binder lab.
Note: In form tag one additional attribute that is enctype is specified. We will talk about it in the end of the lab.

Step 5 - Create Business Layer Upload method

Create a new method called UploadEmployees in EmployeeBusinessLayer as follows.

```
public void UploadEmployees(List<Employee> employees)
{
   SalesERPDAL salesDal = new SalesERPDAL();
   salesDal.Employees.AddRange(employees);
   salesDal.SaveChanges();
}
```

Step 6 – Create Upload Action method

Create a new action method called Upload inside BulkUploadController as follows.

```
[AdminFilter]
public ActionResult Upload(FileUploadViewModel model)
{
   List<Employee> employees = GetEmployees(model);
   EmployeeBusinessLayer bal = new EmployeeBusinessLayer();
   bal.UploadEmployees(employees);
   return RedirectToAction("Index","Employee");
}

private List<Employee> GetEmployees(FileUploadViewModel model)
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
{
   List<Employee> employees = new List<Employee>();
   StreamReader csvreader = new StreamReader(model.fileUpload.InputStream);
   csvreader.ReadLine(); // Assuming first line is header
   while (!csvreader.EndOfStream)
   {
      var line = csvreader.ReadLine();
      var values = line.Split(',');//Values are comma separated
      Employee e = new Employee();
      e.FirstName = values[0];
      e.LastName = values[1];
      e.Salary = int.Parse(values[2]);
      employees.Add(e);
   }
   return employees;
}
```

AdminFilter attached to the Upload Action restrict access to Non-Admin user.

Step 7 – Create a link for BulkUpload

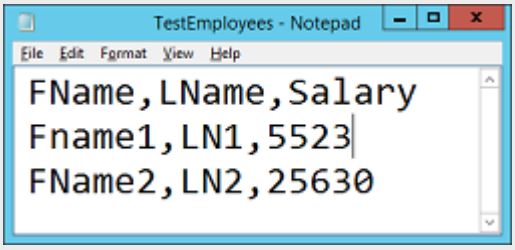
Open AddNewLink.cshtml from "Views/Employee" folder and put a link for BulkUpload as follows.

```
<a href="/Employee/AddNew">Add New</a>
 
 
<a href="/BulkUpload/Index">BulkUpload</a>
```

Step 8 – Execute and Test

Step 8.1 – Create a sample file for testing

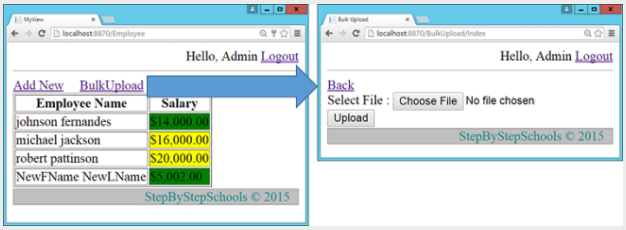
Create a sample file like below and save it somewhere in the computer.



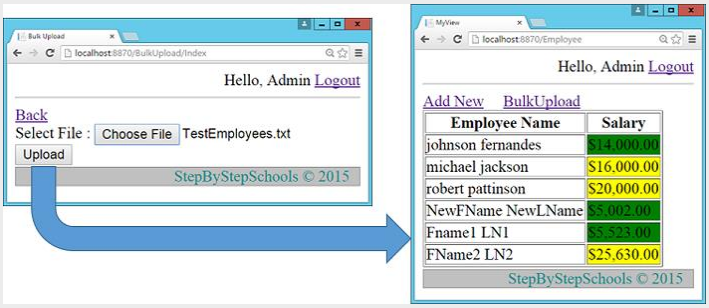
Step 8.2 – Execute and Test

Press F5 and execute the application. Complete the login process and navigate to BulkUpload option by clicking link.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Select File and Click on Upload.



Talk on Lab 27

Why don't we have validations here?

Adding client side and server side validation to this option will be an assignment for readers. I will give you a hint.

- For Server side validation use Data Annotations.
- For client side either you can leverage data annotation and implement jQuery unobtrusive validation. Obviously this time you have to set custom data attributes manually because we don't have readymade Htmlhelper method for file input.
  Note: If you didn't understood this point, I recommend you to go through "implanting client side validation in Login view" again
- For client side validation you can write custom JavaScript and invoke it on button click. This won't be much difficult because file input is an input control at the end of the day and its value can be retrieved inside JavaScript and can be validated.

What is HttpPostedFileBase?

HttpPostedFileBase will provide the access to the file uploaded by client. Model binder will update the value of all properties FileUploadViewModel class during post request. Right now we have only one property inside FileUploadViewModel and Model Binder will set it to file uploaded by client.

Is it possible to provide multiple file input control?

Yes, we can achieve it in two ways.

1. Create multiple file input controls. Each control must have unique name. Now in FileUploadViewModel class create a property of type HttpPostedFileBase one for each control. Each property name should match with the name of one control. Remaining magic will be done by Model Binder. ☺

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

2. Create multiple file input controls. Each control must have same name. Now instead of creating multiple properties of type HttpPostedFileBase, create one of type List< HttpPostedFileBase>.

Note: Above case is true for all controls. When you have multiple controls with same name ModelBinder update the property with the value of first control if property is simple parameter. ModelBinder will put values of each control in a list if property is a list property.

What does enctype="multipart/form-data" will do?

Well this is not a very important thing to know but definitely good to know.
This attribute specifies the encoding type to be used while posting data.
The default value for this attribute is "application/x-www-form-urlencoded"
Example – Our login form will send following post request to the server

```
POST /Authentication/DoLogin HTTP/1.1
Host: localhost:8870
Connection: keep-alive
Content-Length: 44
Content-Type: application/x-www-form-urlencoded
...
...
UserName=Admin&Passsword=Admin& BtnSubmi=Login
```

All input values are sent as one part in the form of key/value pair connected via "&".

When enctype="multipart/form-data" attribute is added to form tag, following post request will be sent to the server.

```
POST /Authentication/DoLogin HTTP/1.1
Host: localhost:8870
Connection: keep-alive
Content-Length: 452
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarywHxplIF8cR8KNjeJ
...
...
------WebKitFormBoundary7hciuLuSNglCR8WC
Content-Disposition: form-data; name="UserName"

Admin
------WebKitFormBoundary7hciuLuSNglCR8WC
Content-Disposition: form-data; name="Password"

Admin
------WebKitFormBoundary7hciuLuSNglCR8WC
Content-Disposition: form-data; name="BtnSubmi"
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
Login
------WebKitFormBoundary7hciuLuSNglCR8WC--
```

As you can see, form is posted in multiple part. Each part is separated by a boundary defined by Content-Type and each part contain one value.
encType must be set to "multipart/form-data" if form tag contains file input control.
Note: boundary will be generated randomly every time request is made. You may see some different boundary.

Why don't we always set encType to "multipart/form-data"?
When encType is set to "multipart/form-data", it will do both the things – Post the data and upload the file. Then why don't we always set it as "multipart/form-data".
Answer is, it will also increase the overall size of the request. More size of the request means less performance. Hence as a best practice we should set it to default that is "application/x-www-form-urlencoded".

Why we have created ViewModel here?
We had only one control in our View. We can achieve same result by directly adding a parameter of type HttpPostedFileBase with name fileUpload in Upload action method Instead of creating a separate ViewModel. Look at the following code.

```
public ActionResult Upload(HttpPostedFileBase fileUpload)
{
}
```

Then why we have created a separate class.
Creating ViewModel is a best practice. Controller should always send data to the view in the form of ViewModel and data sent from view should come to controller as ViewModel.

Problem in the above solution

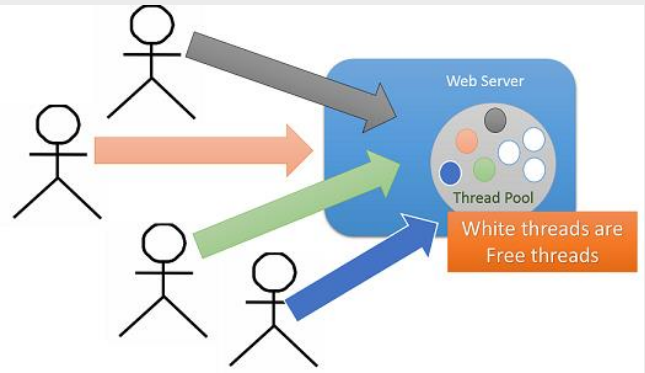
Did you ever wondered how you get response when you send a request?
Now don't say, action method receive request and blah blah blah!!! ☺

Although it's the correct answer I was expecting a little different answer.
My question is what happen in the beginning.

A simple programming rule – everything in a program is executed by a thread even a request.

In case of Asp.net on the webserver .net framework maintains a pool of threads.
Each time a request is sent to the webserver a free thread from the pool is allocated to serve the request. This thread will be called as worker thread.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Worker thread will be blocked while the request is being processed and cannot serve another request. Now let's say an application receives too many requests and each request will take long time to get completely processed. In this case we may end up at a point where new request will get into a state where there will be no worker thread available to serve that request. This is called as Thread Starvation.

In our case sample file had 2 employee records but in real time it may contain thousands or may be lacks of records. It means request will take huge amount of time to complete the processing. It may leads to Thread Starvation.

Solution

Now the request which we had discussed so far is of type synchronous request.
Instead of synchronous if client makes an asynchronous request, problem of thread starvation get solved.

- In case of asynchronous request as usual worker thread from thread pool get allocated to serve the request.
- Worker thread initiates the asynchronous operation and returned to thread pool to serve another request. Asynchronous operation now will be continued by CLR thread.
- Now the problem is, CLR thread can't return response so once it completes the asynchronous operation it notifies Asp.Net.
- Webserver again gets a worker thread from thread pool and processes the remaining request and renders the response.

In this entire scenario two times worker thread is retrieved from thread pool. Now both of them may be same thread or they may not be.

Now in our example file reading is an I/O bound operation which is not required to be processed by worker thread. So it's a best place to convert synchronous requests to asynchronous requests.

Does asynchronous request improves response time?
No, response time will remain same. Here thread will get freed for serving other requests.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Lab 28 – Solve thread starvation problem

In Asp.Net MVC we can convert synchronous requests to asynchronous requests by converting synchronous action methods to asynchronous action methods.

Step 1 - Create asynchronous Controller

Change base class of UploadController to AsynController from Controller.

```
{
   public class BulkUploadController : AsyncController
   {
```

Step 2 – Convert synchronous action method to asynchronous action method

It can be done easily with the help of two keywords – async and await.

```
[AdminFilter]
public async Task<ActionResult> Upload(FileUploadViewModel model)
{
   int t1 = Thread.CurrentThread.ManagedThreadId;
   List<Employee> employees = await Task.Factory.StartNew<List<Employee>>
      (() => GetEmployees(model));
   int t2 = Thread.CurrentThread.ManagedThreadId;
   EmployeeBusinessLayer bal = new EmployeeBusinessLayer();
   bal.UploadEmployees(employees);
   return RedirectToAction("Index", "Employee");
}
```

As you can see we are storing thread id in a variable in the beginning and in the end of action method.

Let's understand the code.

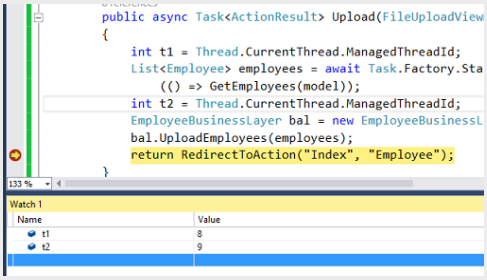
- When upload button is clicked by the client, new request will be sent to the server.
- Webserver will take a worker thread from thread pool and allocate it to serve the request.
- Worker thread make action method to execute.
- Worker method starts an asynchronous operation with the help of Task.Factory.StartNew method.
- As you can see action method is marked as asynchronous with the help of async keyword. It will make sure that worker thread get released as soon as asynchronous operation starts. Now logically asynchronous operation will continue its execution in the background by a separate CLR thread.
- Now asynchronous operation call is marked with await keyword. It will make sure that next line wont executes unless asynchronous operation completes.
- Once Asynchronous operation completes next statement in the action method should execute and for that again a worker thread is required. Hence webserver will simply take up a new free worker thread from thread pool and allocate it to serve the remaining request and to render response.

Step 3 – Execute and Test

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Execute the application. Navigate to BulkUpload option.
Now before you do anything more in the output, navigate to code and put a breakpoint at the last line.

Now select the sample file and click on Upload.



As you can see we have different thread id in the beginning and different in end.
Output is going to be same as previous lab.

Lab 29 – Exception Handling – Display Custom error page

A project will not be considered as a complete project without a proper exception handling.

So far we have spoken about two filters in Asp.Net MVC – Action Filters and Authorization Filters.
Now it's time for third one – Exception Filters.

What are Exception Filters?
Exception Filters will be used in the same way other filters are used. We will use them as an attribute.
Steps for using exception filter

- Enable them.
- Apply them as an attribute to an action method or to a controller. We can also apply exception filter at global level.

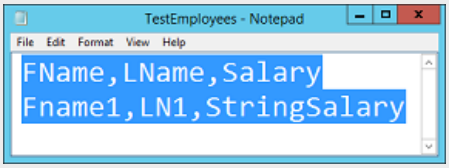
What they will do?
Exception filter will take control of the execution and start executing code written inside it automatically as soon exception occurs inside action method,.
Is there any automation?
Asp.Net MVC provides us one readymade Exception Filter called HandeError.
As we said before it will execute when exception occurs in an action method. This filter will find a view inside "~/Views/[current controller]" or "~/Views/Shared" folder with name "Error" , create the ViewResult of that view and return as a response.

Let's see a demo and understand it better.
In the last lab in our project we have implemented BulkUpload option. Now there is high possibility of error in the input file.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 1 – Create a sample Upload file with Error

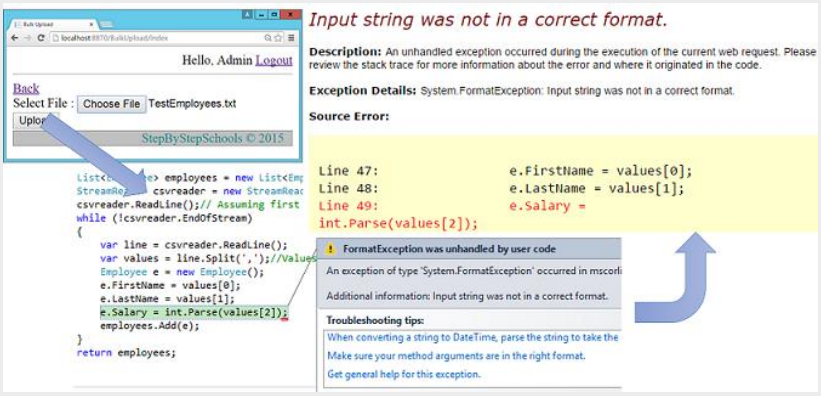
Create a sample upload file just like before but this time put some invalid value.



As you can see, Salary is invalid.

Step 2 – Execute and Test for exception

Press F5 and execute the application. Navigate to Bulk Upload Option. Select above file and click on Upload.



Step 3 – Enable Exception Filters

Exception filters get enabled when custom exception is enabled. To enable Custom Exception, open web.config file and navigate to System.Web Section. Add a new section for custom errors as below.

```
<system.web>
<customErrors mode="On"></customErrors>
```

Step 4 – Create Error View

In "~/Views/Shared" folder you will find a view called "Error.cshtml". This file was created as a part of MVC template in the beginning only. In case if it is not created ,create it manually.

```
@{
  Layout = null;
}

<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>Error</title>
</head>
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
<body>
  <hgroup>
    <h1>Error.</h1>
    <h2>An error occurred while processing your request.</h2>
  </hgroup>
</body>
</html>
```

Step 5 – Attach Exception filter

As we discussed before after we enable exception filter we attach them to action method or to controller.

Good news.☺ It's not required to manually attach it.

Open FilterConfig.cs file from App_Start folder. In RegisterGlobalFilters method you will see that HandleError filter is already attached at global level.

```
public static void RegisterGlobalFilters(GlobalFilterCollection filters)
{
  filters.Add(new HandleErrorAttribute());//ExceptionFilter
  filters.Add(new AuthorizeAttribute());
}
```

If required remove global filter and attach it at action or controller level as below.

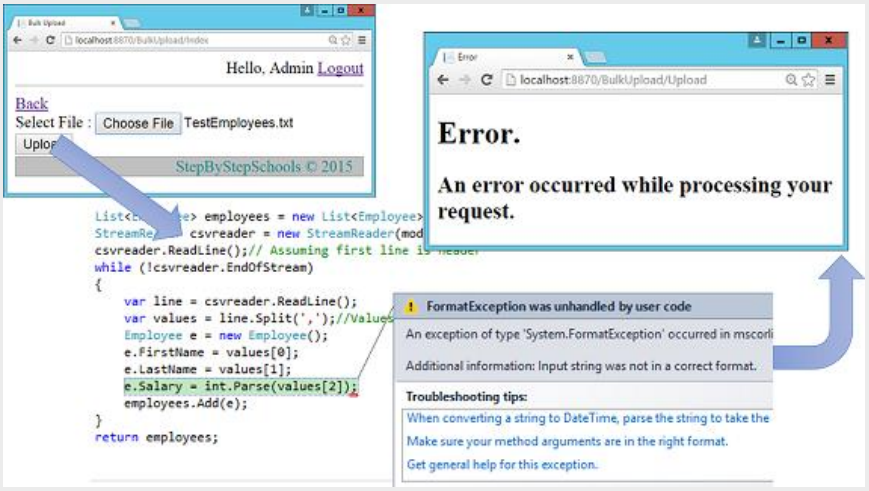
```
[AdminFilter]
[HandleError]
public async Task<ActionResult> Upload(FileUploadViewModel model)
{
```

It's not recommended to do it. It's better to apply at global level.

Step 6 – Execute and Test

Let's test the application in the same way we did before.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



Step 7 – Display Error message in the view

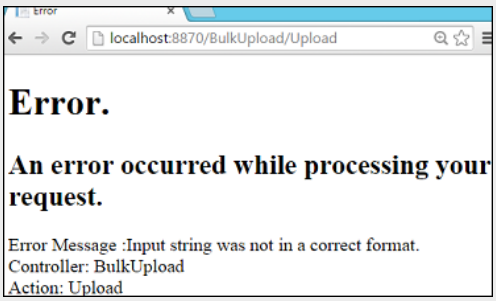
In order to achieve this convert Error view to strongly typed view of type HandleErrorInfo class and then display error messages in the view.

```
@model HandleErrorInfo
@{
   Layout = null;
}

<!DOCTYPE html>
<html>
<head>
   <meta name="viewport" content="width=device-width" />
   <title>Error</title>
</head>
<body>
   <hgroup>
      <h1>Error.</h1>
      <h2>An error occurred while processing your request.</h2>
   </hgroup>
      Error Message :@Model.Exception.Message<br />
      Controller: @Model.ControllerName<br />
      Action: @Model.ActionName
</body>
</html>
```

Step 8 – Execute and Test

Perform the same testing but this time we will get following Error view.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

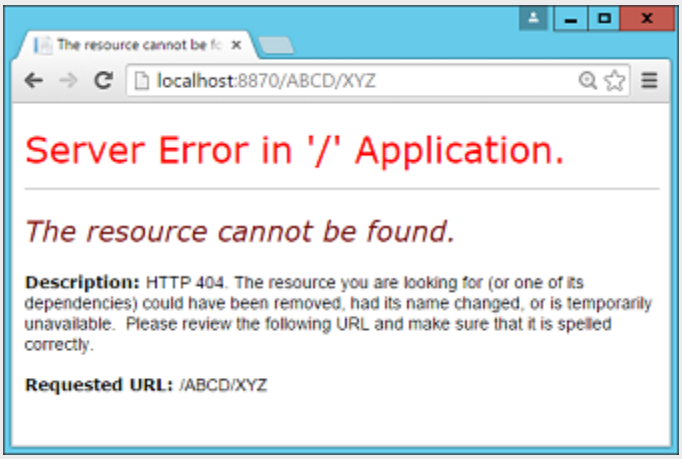


Are we missing something?

Handle error attribute make sure that custom view get displayed whenever exception occurs in an action method. But Its power is limited to controller and action methods.

It will not handle "Resource not found" error.

Execute the application and put something weird in URL



Step 9 – Create ErrorController as follows

Create a new controller called ErrorController in Controller folder and create an action method called Index as follows.

```
public class ErrorController : Controller
{
   // GET: Error
   public ActionResult Index()
   {
      Exception e=new Exception("Invalid Controller or/and Action Name");
      HandleErrorInfo eInfo = new HandleErrorInfo(e, "Unknown", "Unknown");
      return View("Error", eInfo);
   }
}
```

HandleErrorInfo constructor takes 3 arguments – Exception object, controller name and Action method Name.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 10 – Display Custom Error View on Invalid URL

In web.config define setting for "Resource not found error" as follows.

```
<system.web>
  <customErrors mode="On">
   <error statusCode="404" redirect="~/Error/Index"/>
  </customErrors>
```

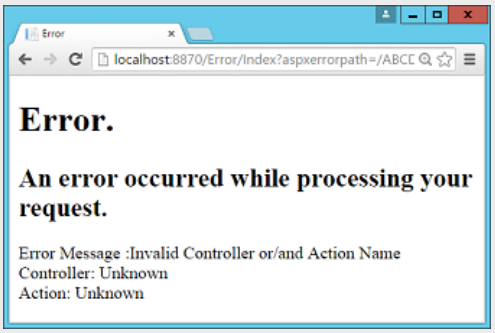
Step 11- Make ErrorController accessible to everyone

Apply AllowAnonymous attribute to ErrorController because error controller and index action should not be bound to an authenticated user. It may be possible that user has entered invalid URL before login.

```
[AllowAnonymous]
public class ErrorController : Controller
{
```

Step 12- Execute and Test

Execute the application and put some invalid URL in the address bar.



Talk on Lab 29

Is it possible to change the view name?

Yes, it's not required to keep view name as "Error" always.

In that case we have to specify view name while attaching HandlError filter.

```
[HandleError(View="MyError")]
Or
filters.Add(new HandleErrorAttribute()
          {
             View="MyError"
          });
```

Is it possible to get different error view for different exceptions?

Yes, it is possible. In that case we have to apply handle error filter multiple times.

```
[HandleError(View="DivideError",ExceptionType=typeof(DivideByZeroException))]
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
[HandleError(View = "NotFiniteError", ExceptionType = typeof(NotFiniteNumberException))]
[HandleError]

OR

filters.Add(new HandleErrorAttribute()
   {
      ExceptionType = typeof(DivideByZeroException),
      View = "DivideError"
   });
filters.Add(new HandleErrorAttribute()
{
   ExceptionType = typeof(NotFiniteNumberException),
   View = "NotFiniteError"
});
filters.Add(new HandleErrorAttribute());
```

In above case we are adding Handle Error filter thrice. First two are specific to exception whereas last one is more general one and it will display Error View for all other exceptions.

Understand limitation in above lab

The only limitation with above lab is we are not logging our exception anywhere.

Lab 30 – Exception Handling – Log Exception

Step 1 – Create Logger class

Create a new Folder called Logger in root location of the project.

Create a new class called FileLogger as follows inside Logger folder.

```
namespace WebApplication1.Logger
{
   public class FileLogger
   {
      public void LogException(Exception e)
      {
         File.WriteAllLines("C://Error//" + DateTime.Now.ToString("dd-MM-yyyy mm hh ss")+".txt",
            new string[]
            {
               "Message:"+e.Message,
               "Stacktrace:"+e.StackTrace
            });
      }
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
    }
}
```

Step 2 – Create EmployeeExceptionFilter class

Create a new class called EmployeeExceptionFilter inside Filters folder as follows.

```
namespace WebApplication1.Filters
{
    public class EmployeeExceptionFilter
    {
    }
}
```

Step 3 - Extend Handle Error to implement Logging

Inherit EmployeeExceptionFilter from HandleErrorAttribute class and override OnException method as follows.

```
public class EmployeeExceptionFilter:HandleErrorAttribute
{
    public override void OnException(ExceptionContext filterContext)
    {
        base.OnException(filterContext);
    }
}
```

Note: Make sure to put using System.Web.MVC in the top. HandleErrorAttribute class exists inside this namespace.

Step 4 – Define OnException method

Include Exception logging code inside OnException method as follows.

```
public override void OnException(ExceptionContext filterContext)
{
    FileLogger logger = new FileLogger();
    logger.LogException(filterContext.Exception);
    base.OnException(filterContext);
}
```

Step 5 – Change Default Exception filter

Open FilterConfig.cs file and remove HandErrorAtrribute and attach the one we created in last step as follows.

```
public static void RegisterGlobalFilters(GlobalFilterCollection filters)
{
    //filters.Add(new HandleErrorAttribute());//ExceptionFilter
    filters.Add(new EmployeeExceptionFilter());
    filters.Add(new AuthorizeAttribute());
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

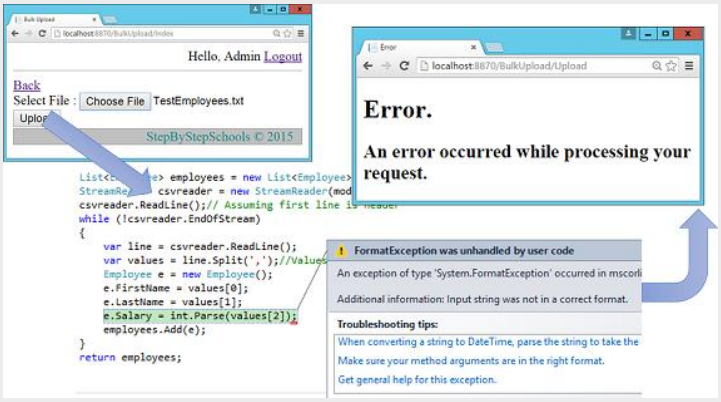
```
}
```

Step 6 – Execute and Test

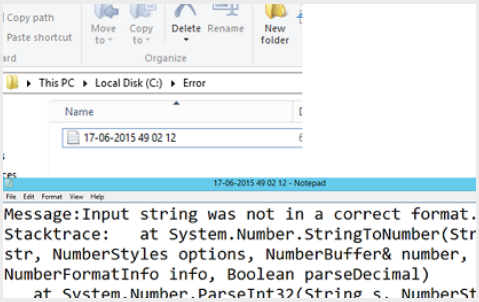
First of all create a folder called "Error" in C drive because that's the place where error files are going to be placed.

Note: If required change the path to your desired one.

Press F5 and execute the application. Navigate to Bulk Upload Option. Select above file and click on Upload.



Output won't be different this time. We will get a same Error View like before. Only difference will be this time we will also find an error file created in "C:\\Errors" folder.



Talk on Lab 30

How come Error view is returned as a response when exception occurs?

In above lab we have overridden the OnException method and implemented exception logging functionality. Now the question is, how come the default handle error filter is still working then? It's simple. Check the last line in the OnException method.

```
base.OnException(filterContext);
```

It means, let base class OnException do the reaming work and base class OnException will return ViewResult of the Error View.

Can we return some other result inside OnException?

Yes. Look at the following code.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
public override void OnException(ExceptionContext filterContext)
{
   FileLogger logger = new FileLogger();
   logger.LogException(filterContext.Exception);
   //base.OnException(filterContext);
   filterContext.ExceptionHandled = true;
   filterContext.Result = new ContentResult()
   {
      Content="Sorry for the Error"
   };
}
```

When we want to return custom response the first thing we should do is, inform MVC engine that we have handled exception manually so don't perform the default behaviour that is don't display the default error screen. This will be done with following statement.

```
filterContext.ExceptionHandled = true
```

You can read more about exception handling in Asp.net MVC here.

Routing

So far we have discussed many concepts, we answered many questions in MVC except one basic and important one.

"What exactly happens when end user makes a request?"

Well answer is definitely "Action method executes". But my exact question is how come controller and action method are identified for a particular URL request.

Before we start with our lab "Implement User Friendly URLs", first let's find out answer for above question. You might be wondering why this topic is coming in the ending. I purposely kept this topic at near end because I wanted people to know MVC well before understanding internals.

Understand RouteTable

In Asp.Net MVC there is a concept called RouteTable which will store URL routes for an application.
In simple words it holds a collection defining possible URL patterns of an application.

By default one route will be added to it as a part of project template. To check it open Global.asax file. In Application_Start you will find a statement something like below.

```
RouteConfig.RegisterRoutes(RouteTable.Routes);
```

You will find RouteConfig.cs file inside App_Start folder which contain following code block.

```
namespace WebApplication1
{
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        );
    }
}
```

As you can see in RegisterRoutes method already one default route is defined with the help of routes.MapRoute method.
Routes defined inside RegisterRoutes method will be used later in the Asp.Net MVC request cycle to determine the exact controller and action method to be executed.

If it's required we can create more than one route using route.MapRoute function. Internally defining route means creating Route object.
MapRoute function also attach RouteHandler to route object which will be MVCRouteHandler in case of Asp.Net MVC by default.

Understand Asp.Net MVC request cycle

Before you start let me make you clear that we are not going to explain 100% request cycle here. We will touch pass only important ones.

Step 1 – UrlRoutingModule
When end user make a request it first pass through UrlRoutingModule Object.
UrlRoutingModule is a HTTP Module.

Step 2 – Routing
UrlRoutingModule will get the first matching Route object from the route table collection.
Now for matching, request URL will be compared with the URL pattern defined in the route.
Following rules will be considered while matching.

- Number of parameters in the requests URL(other than domain name) and in the URL pattern defined in the route.
  Example:

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Route Pattern - {controller}/{action}/{id}	
Request URL	Matching
http://localhost:8870/BulkUpload/Upload/5	yes
http://localhost:8870/BulkUpload/Upload	No
http://localhost:8870/BulkUpload/Upload/5/6	No
http://localhost:8870/1/2/3	Yes

- Optional parameters defined in the URL pattern.
  Example:

Route Pattern - {controller}/{action}/{id} Defaults - new { controller = "Home", action = "Index", id = UrlParameter.Optional }	
Request URL	Matching
http://localhost:8870/BulkUpload/Upload/5	yes
http://localhost:8870/BulkUpload/Upload	yes
http://localhost:8870/BulkUpload	yes
http://localhost:8870	yes
http://localhost:8870/1/2/3	yes
http://localhost:8870/Upload/BulkUpload/5	yes
http://localhost:8870/BulkUpload/Upload/5/6	No

- Static parameters defined in the parameter.
  Example:

Route Pattern - ABC/{controller}/PQR/{action}/XYZ	
Request URL	Matching
http://localhost:8870/A/B/C/D/E	No
http://localhost:8870/ABC/A/B/C/D	No
http://localhost:8870/ABC/A/PQR/C/XYZ	yes

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 3 – Create MVC Route Handler

Once the Route object is selected, UrlRoutingModule will obtain MvcRouteHandler object from Route object.

Step 4 – Create RouteData and RequestContext

UrlRoutingModule object will create RouteData using Route object, which it then uses to create RequestContext.

RouteData encapsulates information about routes like name of the controller, name of the action and values of route parameters.

Controller Name

In order to get the controller name from the request URL following simple rule is followed.

"In the URL pattern {controller} is the keyword to identify Controller Name".

Example:

- When URL pattern is **{controller}/{action}/{id}** and request URL is "http://localhost:8870/BulkUpload/Upload/5", BulkUpload will be name of the controller.
- When URL pattern is **{action}/{controller}/{id}** and request URL is "http://localhost:8870/BulkUpload/Upload/5", Upload will be name of the controller.

Action Method Name

In order to get the action method name from the request URL following simple rule is followed.

"In the URL pattern {action} is the keyword to identify action method Name".

Example:

- When URL pattern is **{controller}/{action}/{id}** and request URL is "http://localhost:8870/BulkUpload/Upload/5", Upload will be name of the action method.
- When URL pattern is **{action}/{controller}/{id}** and request URL is "http://localhost:8870/BulkUpload/Upload/5", Upload will be name of the action method.

Route Parameters

Basically a URL pattern can contain following four things

1. {controller} -> Identifies controller name
2. {action} -> Identifies action method name.
3. SomeString -> Example – "MyCompany/{controller}/{action}" -> in this pattern "MyCompany" becomes compulsory string.
4. {Something} -> Example – "{controller}/{action}/{id}" -> In this pattern "id" is the route parameter. Route parameter can be used to get the value in the URL itself at the time of request.

Look at the following example.

Route pattern - > "{controller}/{action}/{id}"

Request URL -> http://localhost:8870/BulkUpload/Upload/5

Testing 1

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
public class BulkUploadController : Controller
{
   public ActionResult Upload (string id)
   {
     //value of id will be 5 -> string 5
     ...
   }
}
```

Testing 2

```
public class BulkUploadController : Controller
{
   public ActionResult Upload (int id)
   {
     //value of id will be 5 -> int 5
     ...
   }
}
```

Testing 3

```
public class BulkUploadController : Controller
{
   public ActionResult Upload (string MyId)
   {
     //value of MyId will be null
     ...
   }
}
```

Step 5 – Create MVCHandler

MvcRouteHandler will create the instance of MVCHandler passing RequestContext object.

Step 6 – Create Controller instance

MVCHandler will create Controller instance with the help of ControllerFactory (by default it will be DefaultControllerFactory).

Step 7 – Execute method

MVCHandler will invoke Controller's execute method. Execute method is defined inside controller base class.

Step 8 – Invoke Action method

Every controller is associated with a ControllerActionInvoker object. Inside execute method ControllerActionInvoker object invoke the correct action method.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

Step 9 – Execute result.

Action method receives the user input and prepares the appropriate response data and then executes the result by returning a return type. Now that return type may be ViewResult, may be RedirectToRoute Result or may be something else.

Now I believe you have good understanding the concept of Routing so let make our Project URLs more user-friendly with the help of routing.

Lab 31 – Implement User friendly URLs

Step 1 – Redefine RegisterRoutes method

Include additional routes in the RegisterRoutes method as follows.

```
public static void RegisterRoutes(RouteCollection routes)
{
   routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

   routes.MapRoute(
   name: "Upload",
   url: "Employee/BulkUpload",
   defaults: new { controller = "BulkUpload", action = "Index" }
   );

   routes.MapRoute(
      name: "Default",
      url: "{controller}/{action}/{id}",
      defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
   );
}
```

As you can see now we have more than one route defined.
(Default Route is kept untouched.)

Step 2 – Change URL references

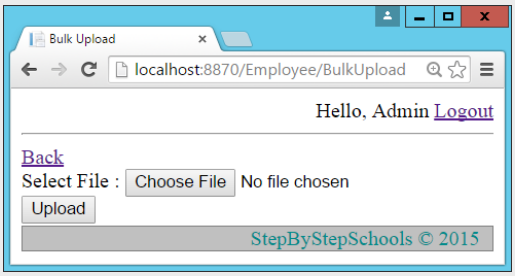
Open AddNewLink.cshtml from "~/Views/Employee" folder and change BulkUpload link as follows.

```
 
<a href="/Employee/BulkUpload">BulkUpload</a>
```

Step 3 – Execute and Test

Execute the application and see the magic.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917



As you can see URL is no more in the form of "Controller/Action". Rather it is more user friendly but output is same.

I recommend you to defines some more routes and try some more URLs.

Talk on Lab 31

Does earlier URL work now?

Yes, earlier URL will work too.

Now Index action in the BulkUploadController is accessible from two URLs

1. "http://localhost:8870/Employee/BulkUpload"
2. "http://localhost:8870/BulkUpload/Index"

What is "id" in default route?

We already spoke about it. It's called Route Parameter. It can be used to get values via URL.

It's a replacement for Query String.

What is the difference between Route Parameter and Query String?

- Query String have size limitation whereas we can define any number of Route Parameters.
- We cannot add constraints to Query String values but we can add constraints to Route Parameters.
- Default Value for Route Parameter is possible whereas default value for Query String is not possible.
- Query String makes URL cluttered whereas Route Parameter keep it clean.

How to apply constraints to Route Parameter?

It can be done with the help of regular expressions.

Example: look at the following route.

```
routes.MapRoute(
   "MyRoute",
   "Employee/{EmpId}",
   new {controller=" Employee ", action="GetEmployeeById"},
   new { EmpId = @"\d+" }
 );
```

Action method will look like below.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
public ActionResult GetEmployeeById(int EmpId)
{
  ...
}
```

Now when someone make a request with URL "http://..../Employee/1" or "http://..../Employee/111", action method will get executed but when someone make a request with URL "http://..../Employee/Sukesh" he/she will get "Resource Not Found" Error.

Is it a required to keep parameter name in action method same as Route Parameter Name?

Basically a single Route pattern may contain one or more RouteParameters involved in it.
To identify each route parameter independently it is must to keep parameter name in action method same as Route Parameter Name.

Does sequence while defining custom routes matters?

Yes, it matters. If you remember UrlRoutingModule will take first matching route object.

In the above lab we have defined two routes. One custom route and one default route.
Now let say for an instance default route is defined first and custom route is defined second.

In this case when end user make a request with URL "http://.../Employee/BulkUpload" in the comparison phase UrlRoutingModule finds that requested URL matches with the default route pattern and it will consider "Employee" as the controller name and "BulkUpload" as the action method name.

Hence sequence is very important while defining routes. Most generic route should be kept at the end.

Is there an easier way to define URL pattern for Action method?

We can use Attribute based routing for that.
Let's try it.

Step 1 – Enable Attribute based routing.

In RegisterRoutes method keep following line after IgnoreRoute statement.

```
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

routes.MapMvcAttributeRoutes();

routes.MapRoute(
...
```

Step 2 – Define route pattern for an action method

Simply attach Route attribute to Action method to Index action of EmployeeController as follows.

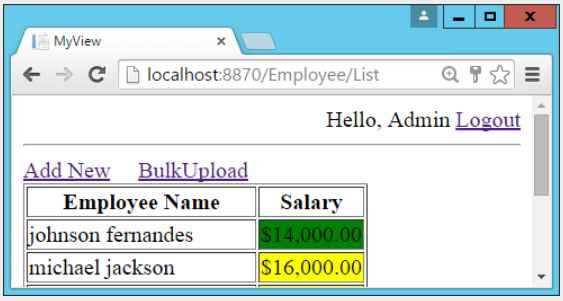
```
[Route("Employee/List")]
public ActionResult Index()
```

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

```
{
```

Step 3 - Execute and Test

Execute the application and complete login process.



As you can see, we have same output but with different more User Friendly URL.

Can we define Route Parameters with attribute based routing?

Yes, look at the following syntax.

```
[Route("Employee/List/{id}")]
public ActionResult Index (string id) { ... }
```

What about the constraints in this case?

It will be easier.

```
[Route("Employee/List/{id:int}")]
```

We can have following constraints

1. {x:alpha} – string validation
2. {x:bool} – Boolean validation
3. {x:datetime} – Date Time validation
4. {x:decimal} – Decimal validation
5. {x:double} – 64 bit float point value validation
6. {x:float} – 32 bit float point value validation
7. {x:guid} – GUID validation
8. {x:length(6)} –length validation
9. {x:length(1,20)} – Min and Max length validation
10. {x:long} – 64 int validation
11. {x:max(10)} – Max integer number validation
12. {x:maxlength(10)} – Max length validation
13. {x:min(10)} – Min Integer number validation
14. {x:minlength(10)} – Min length validation
15. {x:range(10,50)} – Integer range validation
16. {x:regex(SomeRegularExpression)} – Regular Expression validation

What does IgnoreRoutes do in RegisterRoutes method?

IgnoreRoutes will be used when we don't want to use routing for a particular extension.

Learn MVC quickly and easily this Saturday and Sunday in Mumbai with our special "Learn MVC in two days offline session". For detail call us on 022-66752917

As a part of MVC template following statement is already written in RegisterRoutes method.

```
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
```

It means if end user make a request with an extension ".axd" then there won't be any routing operation. Request will directly reach to physical resource.

We can define our own IgnoreRoute Statement as well.

Conclusion

With Day 6 we have completed our sample MVC project. Hope you have enjoyed the complete series.

Wait!!! Where is Day 7?

Day 7 will be there my friends. In day 7 we will create a Single Page Application using MVC, jQuery and Ajax. It will be more fun and more challenge. Stay tuned ☺

Your comments, Mails always motivates us do more. Put your thoughts and comments below or send mail to SukeshMarla@Gmail.com

Connect us on Facebook, LinkedIn or twitter to stay updated about new releases.

For Offline Technical trainings in Mumbai visit StepByStepSchools.Net
For Online Trainings visit JustCompile.com or www.Sukesh-Marla.com