100 important MVC Interview Question

By

Shivprasad Koirala
And
Sukesh Marla

Do not miss our 2 days (Saturday and Sunday) full hands on MVC training course in Mumbai call 022-66752917

Want to learn MVC online start from the below youtube video

https://www.youtube.com/watch?v=Lp7nSImO5vk

Contents

What is MVC?

MVC is an architectural pattern which separates the representation and the user interaction. It's divided in to three broader sections, "Model", "View" and "Controller". Below is how each one of them handles the task.

- The "View" is responsible for look and feel.
- "Model" represents the real world object and provides data to the "View".
- The "Controller" is responsible to take the end user request and load the appropriate "Model" and "View".

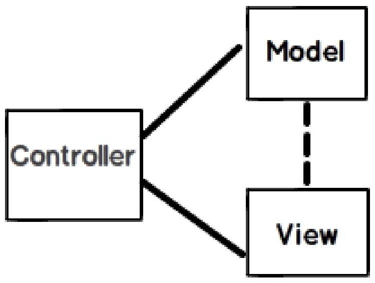
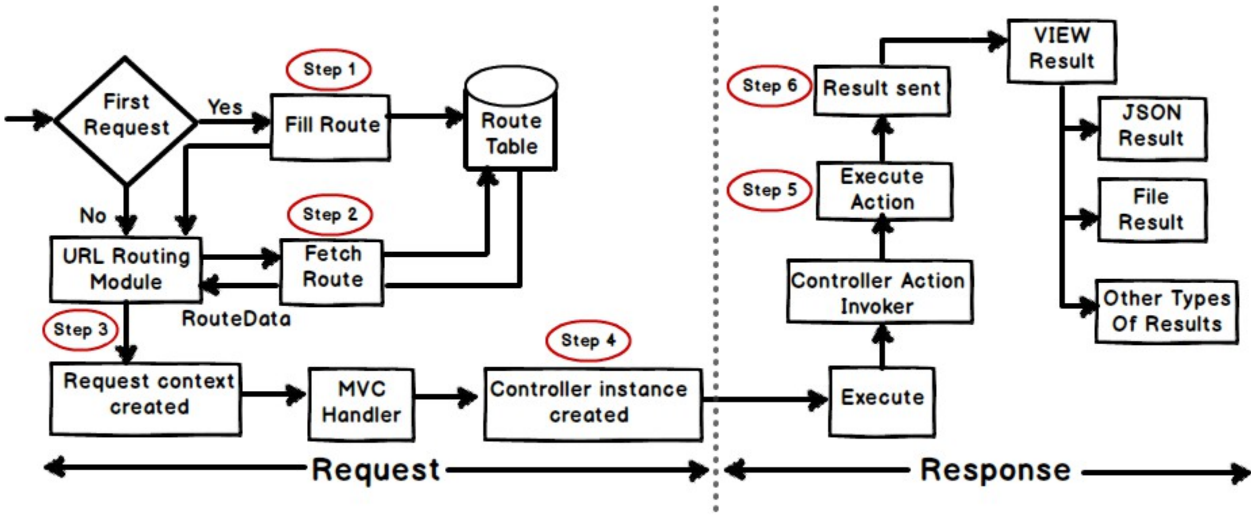


Figure 6.1:- MVC (Model view controller)

Explain MVC application life cycle?

```
Note: - There  is nothing as such called as MVC life cycle. I think lot of
people are obsessed with ASP.NET page life cycle and they think there is life
cycle in MVC as well. To be specific the MVC request goes through various
steps of execution and that's what is  termed as MVC application life cycle.
```



Any web application has two main execution steps first understanding the request and depending on the type of the request sending out appropriate response. MVC application life cycle is not different it has two main phases first creating the request object and second sending our response to the browser.

Creating the request object: - The request object creation has four major steps. Below is the detail explanation of the same.

Step 1 Fill route: - MVC requests are mapped to route tables which in turn specify which controller and action to be invoked. So if the request is the first request the first thing is to fill the route table with routes collection. This filling of route table happens in the global.asax file.

Step 2 Fetch route:- Depending on the URL sent "UrlRoutingModule" searches the route table to create "RouteData" object which has the details of which controller and action to invoke.

Step 3 Request context created: - The "RouteData" object is used to create the "RequestContext" object.

Step 4 Controller instance created: - This request object is sent to "MvcHandler" instance to create the controller class instance. Once the controller class object is created it calls the "Execute" method of the controller class.

Creating Response object: - This phase has two steps executing the action and finally sending the response as a result to the view.

Step 5 Execute Action: - The "ControllerActionInvoker" determines which action to executed and executes the action.

Step 6 Result sent: - The action method executes and creates the type of result which can be a view result , file result , JSON result etc.

So in all there are six broad steps which get executed in MVC application life cycle.

Note :- In case you are not able to remember the above steps during interview remember the acronym FFRCER(Fight For Respect Can Evoke Revolution).

Is MVC suitable for both windows and web application?

MVC architecture is suited for web application than windows. For window application MVP i.e. "Model view presenter" is more applicable.IfyouareusingWPFandSLMVVMismoresuitableduetobindings.

What are the benefits of using MVC?

There are two big benefits of MVC:-

- Separation of concerns is achieved as we are moving the code behind to a separate class file. By moving the binding code to a separate class file we can reuse the code to a great extent.
- Automated UI testing is possible because now the behind code (UI interaction code) has moved to a simple.NET class. This gives us opportunity to write unit tests and automate manual testing.

Is MVC different from a 3 layered architecture?

MVC is an evolution of a 3 layered traditional architecture. Many components of 3 layered architecture are part of MVC. So below is how the mapping goes.

Functionality	**3 layered / tiered architecture**	**Model view controller architecture**
Look and Feel	User interface.	View.
UI logic	User interface.	Controller
Business logic /validations	Middle layer	Model.
Request is first sent to	User interface	Controller.
Accessing data	Data access layer.	Data access layer.

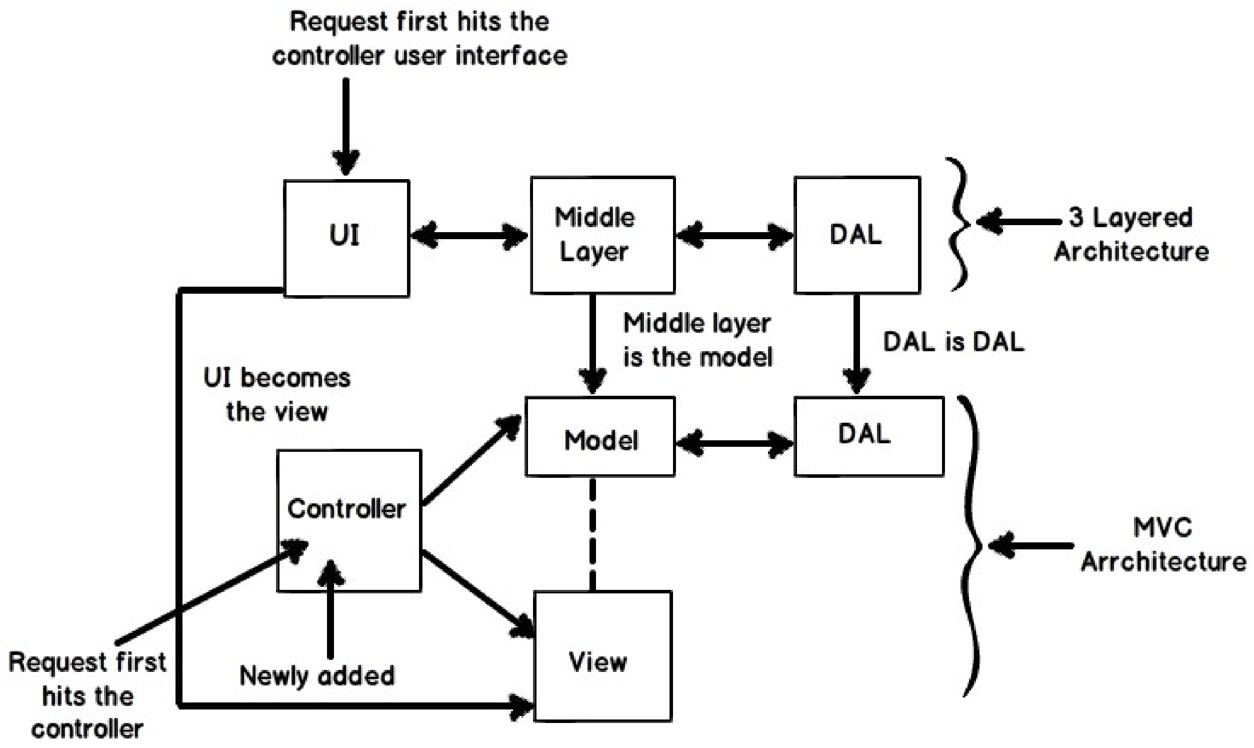


Figure 6.2:- 3 layered architecture

What is the latest version of MVC?

When this note was written, four versions where released of MVC. MVC 1 , MVC 2, MVC 3 and MVC 4. So the latest is MVC 5.

What is the difference between each version of MVC?

Below is a detail table of differences. But during interview it's difficult to talk about all of them due to time limitation. So I have highlighted important differences which you can run through before the interviewer.

MVC 2	MVC 3	MVC 4
Client-Side Validation **Templated Helpers** **Areas** **Asynchronous Controllers** Html.ValidationSummary Helper Method DefaultValueAttribute in Action-Method Parameters Binding Binary Data with Model Binders DataAnnotations Attributes	**Razor** Readymade project templates **HTML 5 enabled templates** **Support for Multiple View Engines** **JavaScript and Ajax** Model Validation Improvements	**ASP.NET Web API** Refreshed and modernized default project templates New mobile project template **Many new features to support mobile apps** Enhanced support for asynchronous methods

Model-Validator Providers New RequireHttpsAttribute Action Filter Templated Helpers Display Model-Level Errors		

What are HTML helpers in MVC?

HTML helpers help you to render HTML controls in the view. For instance if you want to display a HTML textbox on the view, below is the HTML helper code.

```
<%= Html.TextBox("LastName") %>
```

For checkbox below is the HTML helper code. In this way we have HTML helper methods for every HTML control that exists.

```
<%= Html.CheckBox("Married") %>
```

What is the difference between “HTML.TextBox” vs “HTML.TextBoxFor”?

Both of them provide the same HTML output, “HTML.TextBoxFor” is strongly typed while “HTML.TextBox” isn’t.

Below is a simple HTML code which just creates a simple textbox with “CustomerCode” as name.

```
Html.TextBox("CustomerCode")
```

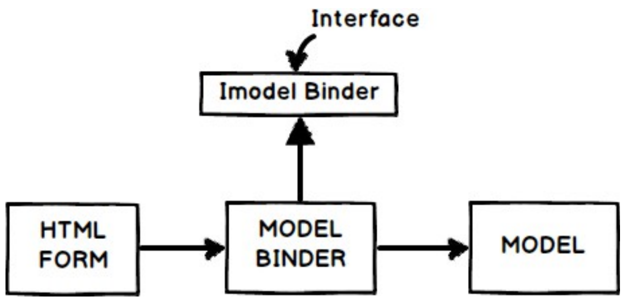
Below is “Html.TextBoxFor” code which creates HTML textbox using the property name ‘CustomerCode” from object “m”.

```
Html.TextBoxFor(m => m.CustomerCode)
```

In the same way we have for other HTML controls like for checkbox we have “Html.CheckBox” and “Html.CheckBoxFor”.

Explain the importance of MVC model binders?

Model binder maps HTML form elements to the model. It acts like a bridge between HTML UI and MVC model.



Take the below simple HTML form example:-

```
<form id="frm1" method=post action="/Customer/SubmitCustomer">
            Customer code :- <input name="CCode" type="text" />
            Customer name :- <input name="CName" type="text" />
           <input type=submit/>
    </form>
```

Now this form needs to fill the below "Customer" class model. If you see the HTML control name they are different from the class property name. For example HTML textbox control name is "CCode" and the class property name is "CustomerCode". This mapping code is written in HTML binder classes.

```
public class Customer
{
        public string CustomerCode { get; set; }
        public string CustomerName { get; set; }
}
```

To create a model binder we need to implement "IModelBinder" interface and mapping code needs to be written in the "BindModel" method as shown in the below code.

```
public class CustomerBinder : IModelBinder
{

public object BindModel(ControllerContext controllerContext, ModelBindingContext
bindingContext)
{
            HttpRequestBase request = controllerContext.HttpContext.Request;

            string strCustomerCode = request.Form.Get("CCode");
            string strCustomerName = request.Form.Get("CName");

            return new Customer
            {
                CustomerCode = strCustomerCode,
                CustomerName = strCustomerName
            };
}
}
```

Now in the action result method we need to use the “ModelBinder” attribute which will attach the binder with the class model.

```
public ActionResult SubmitCustomer([ModelBinder(typeof(CustomerBinder))]Customer obj)
{

            return View(“DisplayCustomer”);
}
```

What are routing in MVC?

Routing helps you to define user friendly URL structure and map those URL structure to the controller.

For instance let’s say we want that when any user types “http://localhost/View/ViewCustomer/” , it goes to the “Customer” Controller and invokes “DisplayCustomer” action. This is defined by adding an entry in to the “routes” collection using the “maproute” function. Below is the under lined code which shows how the URL structure and mapping with controller and action is defined.

```
routes.MapRoute(
                "View", // Route name
                "View/ViewCustomer/{id}", // URL with parameters
                new { controller = "Customer", action = "DisplayCustomer", id
= UrlParameter.Optional }); // Parameter defaults
```

Where is the route mapping code written?

The route mapping code is written in the “global.asax” file.

Can we map multiple URL’s to the same action?

Yes , you can , you just need to make two entries with different key names and specify the same controller and action.

How can we navigate from one view to other view using hyperlink?

By using “ActionLink” method as shown in the below code. The below code will create a simple URL which help to navigate to the “Home” controller and invoke the “GotoHome” action.

```
<%= Html.ActionLink("Home","Gotohome") %>
```

How can we restrict MVC actions to be invoked only by GET or POST?

We can decorate the MVC action by “HttpGet” or “HttpPost” attribute to restrict the type of HTTP calls. For instance you can see in the below code snippet the “DisplayCustomer” action can only be invoked by “HttpGet”. If we try to make Http post on “DisplayCustomer” it will throw an error.

```
        [HttpGet]
          public ViewResult DisplayCustomer(int id)
          {
              Customer objCustomer = Customers[id];

              return View("DisplayCustomer",objCustomer);
          }
```

How can we maintain session in MVC?

Sessions can be maintained in MVC by 3 ways tempdata , viewdata and viewbag.

What is the difference between tempdata , viewdata and viewbag?

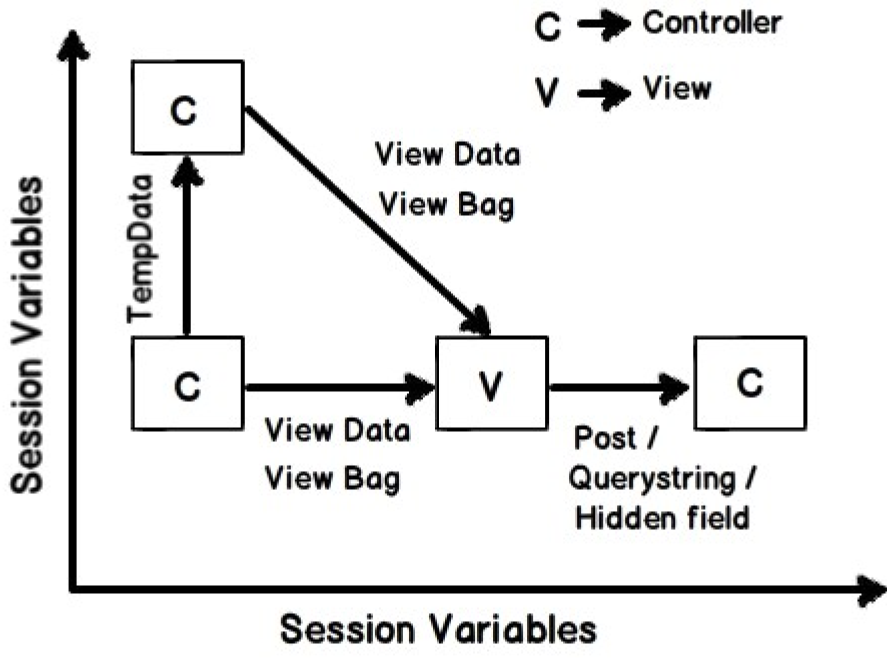


Figure 6.3:- difference between tempdata , viewdata and viewbag

Temp data: -Helps to maintain data when you move from one controller to other controller or from one action to other action. In other words when you redirect,"tempdata" helps to maintain data between those redirects. It internally uses session variables.

View data: - Helps to maintain data when you move from controller to view.

View Bag: - It's a dynamic wrapper around view data. When you use "Viewbag" type casting is not required. It uses the dynamic keyword internally.

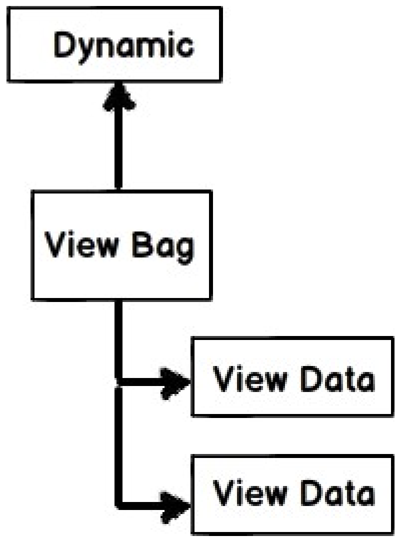


Figure 6.4:-dynamic keyword

Session variables: - By using session variables we can maintain data from any entity to any entity.

Hidden fields and HTML controls: - Helps to maintain data from UI to controller only. So you can send data from HTML controls or hidden fields to the controller using POST or GET HTTP methods.

Below is a summary table which shows different mechanism of persistence.

Maintains data between	**ViewData/ViewBag**	**TempData**	**Hidden fields**	**Session**
Controller to Controller	No	Yes	No	Yes
Controller to View	Yes	No	No	Yes
View to Controller	No	No	Yes	Yes

What is life of "TempData" ?

"TempData" is available for the current request and in the subsequent request it's available depending on whether "TempData" is read or not.

So if "TempData" is once read it will not be available in the subsequent request.

What is the use of Keep and Peek in "TempData"?

Once "TempData" is read in the current request it's not available in the subsequent request. If we want "TempData" to be read and also available in the subsequent request then after reading we need to call "Keep" method as shown in the code below.

```
@TempData["MyData"];

TempData.Keep("MyData");
```

The more shortcut way of achieving the same is by using "Peek". This function helps to read as well advices MVC to maintain "TempData" for the subsequent request.

```
string str = TempData.Peek("Td").ToString();
```

What are partial views in MVC?

Partial view is a reusable view (like a user control) which can be embedded inside other view. For example let's say all your pages of your site have a standard structure with left menu, header and footer as shown in the image below.

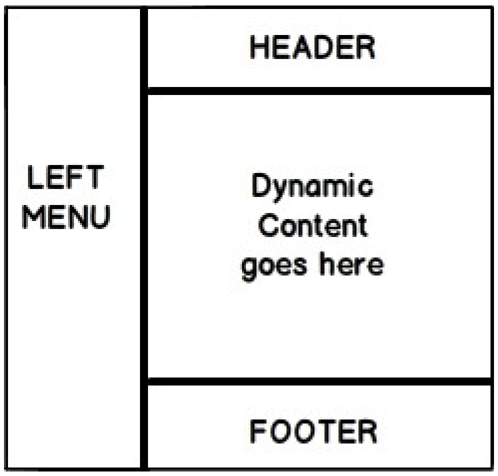
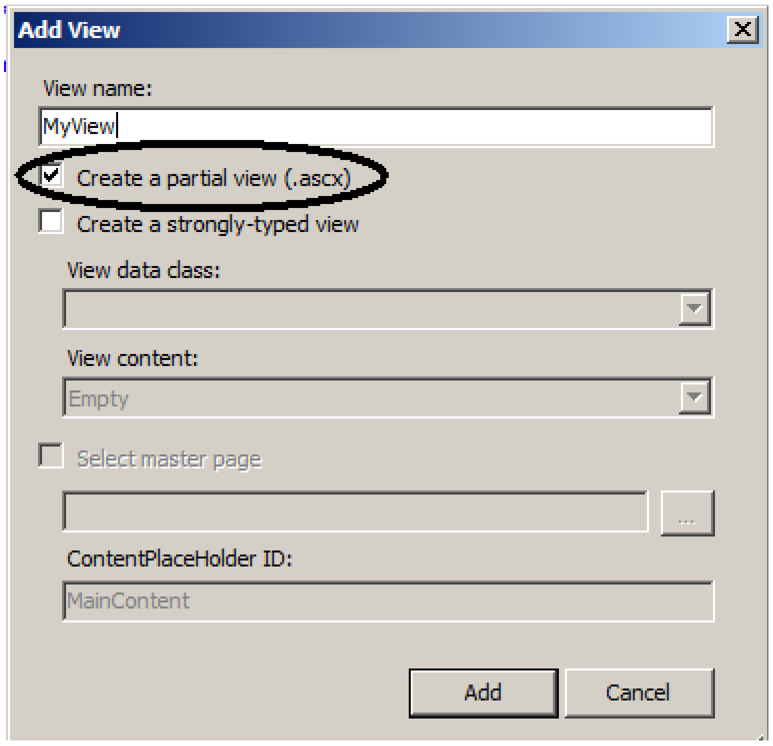


Figure 6.5:- partial views in MVC

For every page you would like to reuse the left menu, header and footer controls. So you can go and create partial views for each of these items and then you call that partial view in the main view.

How did you create partial view and consume the same?

When you add a view to your project you need to check the "Create partial view" check box.



MyView

[✓ Create a partial view (.ascx)]

Create a strongly-typed view

View data class:

View content: Empty

Select master page

ContentPlaceHolder ID: MainContent

Figure6.6:-createpartialview

Once the partial view is created you can then call the partial view in the main view using "Html.RenderPartial" method as shown in the below code snippet.

```
<body>
<div>
<% Html.RenderPartial("MyView"); %>
</div>
</body>
```

How can we do validations in MVC?

One of the easy ways of doing validation in MVC is by using data annotations. Data annotations are nothing but attributes which you can be applied on the model properties. For example in the below code snippet we have a simple "customer" class with a property "customercode".

This"CustomerCode" property is tagged with a "Required" data annotation attribute. In other words if this model is not provided customer code it will not accept the same.

```
public class Customer
{
        [Required(ErrorMessage="Customer code is required")]
        public string CustomerCode
        {
            set;
            get;
        }
}
```

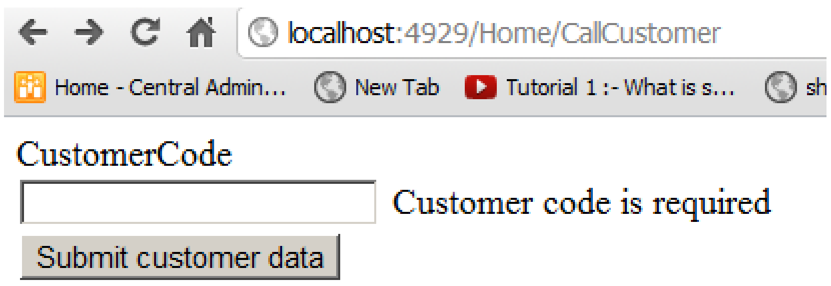
In order to display the validation error message we need to use "ValidateMessageFor" method which belongs to the "Html" helper class.

```
<% using (Html.BeginForm("PostCustomer", "Home", FormMethod.Post))
{ %>
<%=Html.TextBoxFor(m => m.CustomerCode)%>
<%=Html.ValidationMessageFor(m => m.CustomerCode)%>
<input type="submit" value="Submit customer data" />
<%}%>
```

Later in the controller we can check if the model is proper or not by using "ModelState.IsValid" property and accordingly we can take actions.

```
public ActionResult PostCustomer(Customer obj)
{
if (ModelState.IsValid)
{
                obj.Save();
                return View("Thanks");
}
else
{
                return View("Customer");
}
}
```

Below is a simple view of how the error message is displayed on the view.



CustomerCode

Customer code is required

Submit customer data

Figure 6.7:- validations in MVC

Can we display all errors in one go?

Yes we can, use "ValidationSummary" method from HTML helper class.

```
<%= Html.ValidationSummary() %>
```

What are the other data annotation attributes for validation in MVC?

If you want to check string length, you can use "StringLength".

```
[StringLength(160)]
public string FirstName { get; set; }
```

In case you want to use regular expression, you can use "RegularExpression" attribute.

```
[RegularExpression(@"[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}")]
public string Email { get; set; }
```

If you want to check whether the numbers are in range, you can use the "Range" attribute.

```
[Range(10,25)]
public int Age { get; set; }
```

Some time you would like to compare value of one field with other field, we can use the "Compare" attribute.

```
public string Password { get; set; }
[Compare("Password")]
public string ConfirmPass { get; set; }
```

In case you want to get a particular error message , you can use the "Errors" collection.

```
var ErrMessage = ModelState["Email"].Errors[0].ErrorMessage;
```

If you have created the model object yourself you can explicitly call “TryUpdateModel” in your controller to check if the object is valid or not.

```
TryUpdateModel(NewCustomer);
```

In case you want add errors in the controller you can use “AddModelError” function.

```

ModelState.AddModelError("FirstName", "This is my server-side error.");
```

How can we enable data annotation validation on client side?

It’s a two-step process first reference the necessary jquery files.

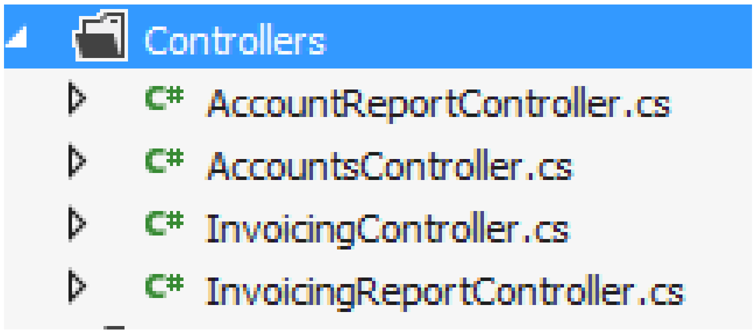
```
<script src="<%= Url.Content("~/Scripts/jquery-1.5.1.js") %>"
type="text/javascript"></script>
<script src="<%= Url.Content("~/Scripts/jquery.validate.js") %>"
type="text/javascript"></script>
<script src="<%= Url.Content("~/Scripts/jquery.validate.unobtrusive.js") %>"
type="text/javascript"></script>
```

Second step is to call “EnableClientValidation” method.

```
<% Html.EnableClientValidation(); %>
```

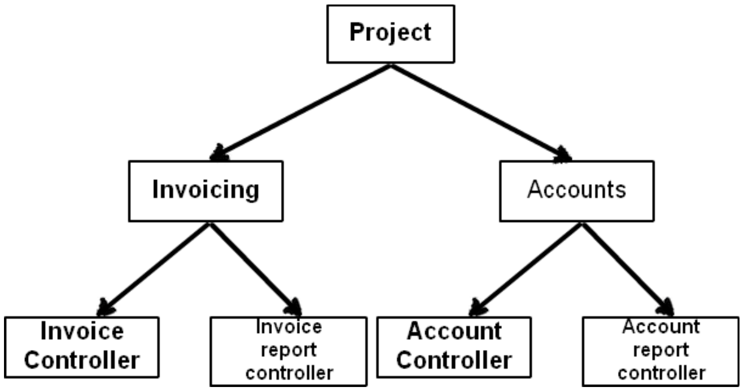
Explain Areas in MVC?

Areas help you to group functionalities in to independent modules thus making your project more organized. For example in the below MVC project we have four controller classes and as time passes by if more controller classes are added it will be difficult to manage. In bigger projects you will end up with 100’s of controller classes making life hell for maintenance.

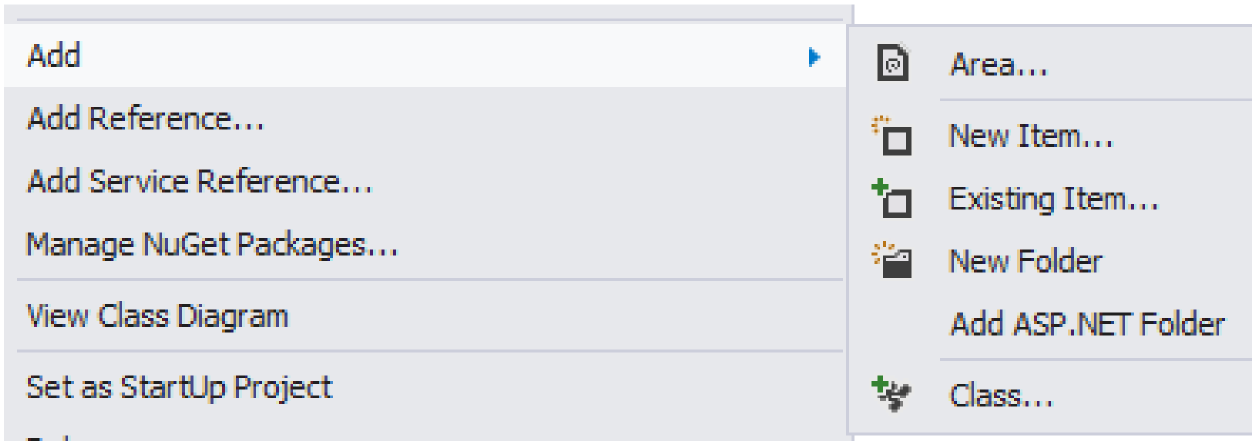


Controllers
AccountReportController.cs
AccountsController.cs
InvoicingController.cs
InvoicingReportController.cs

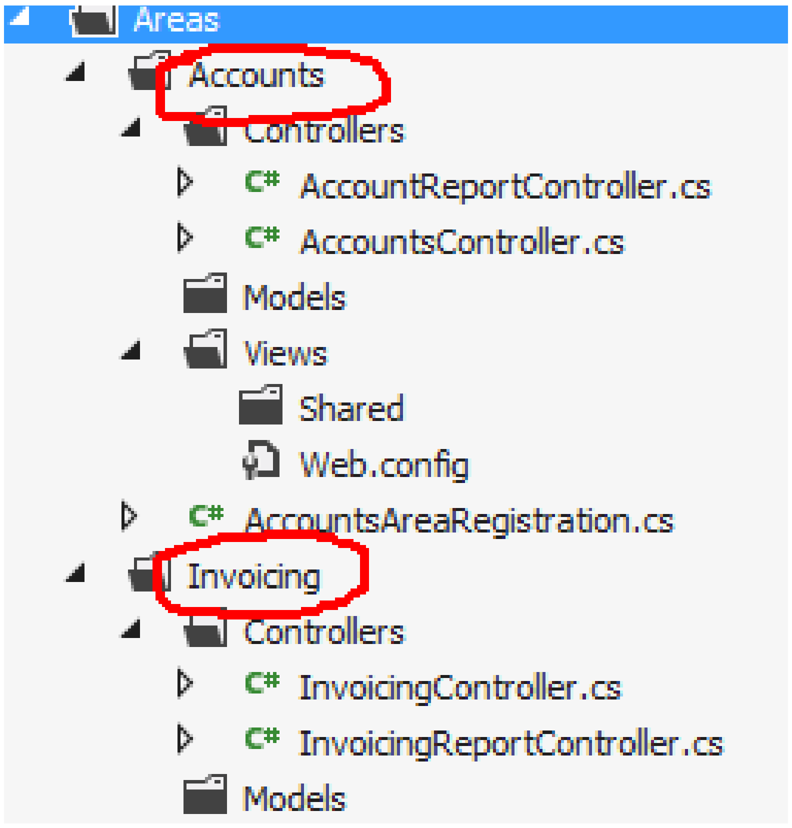
If we can group controller classes in to logical section like "Invoicing" and "Accounting" that would make life easier and that's what "Area" are meant to.



You can add an area by right clicking on the MVC solution and clicking on "Area" menu as shown in the below figure.



In the below image we have two "Areas" created "Account" and "Invoicing" and in that I have put the respective controllers. You can see how the project is looking more organized as compared to the previous state.



What is razor in MVC?

It's a light weight view engine. Till MVC we had only one view type i.e.ASPX, Razor was introduced in MVC 3.

Why razor when we already had ASPX?

Razor is clean, lightweight and syntaxes are easy as compared to ASPX. For example in ASPX to display simple time we need to write.

```
<%=DateTime.Now%>
```

In Razor it's just one line of code.

```

@DateTime.Now
```

So which is a better fit Razor or ASPX?

Microsoft razor is more preferred because it's light weight and has simple syntaxes.

Explain the difference between layout and master pages ?

Layout are like master pages in ASP.NET Web form. Master pages give a standard look and feel for Web form views while layout gives standard look and feel or acts like a template for razor views.

How to apply layout to Razor views?

So first we need to create a template file as shown in the below code.

```
<div>
@RenderSection("Header")
@RenderBody()
@RenderSection("Footer")
</div>
```

And then apply this template to the view as shown below and display data in those respective sections.

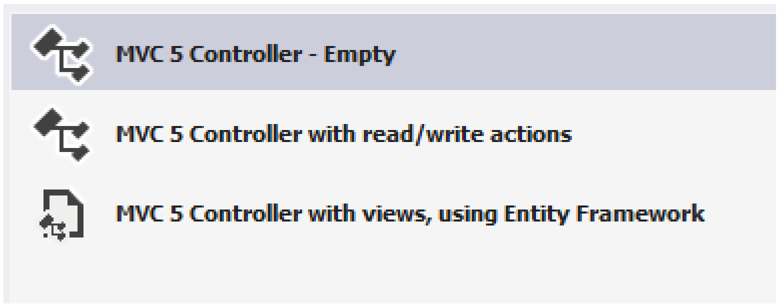
```
@{Layout = "~/Views/Default1/_LayoutPage1.cshtml";}
This is body
@section  Footer{Copyright 2015-2016}
@section Header{Welcome to my site}
```

Explain the concept of Scaffolding?

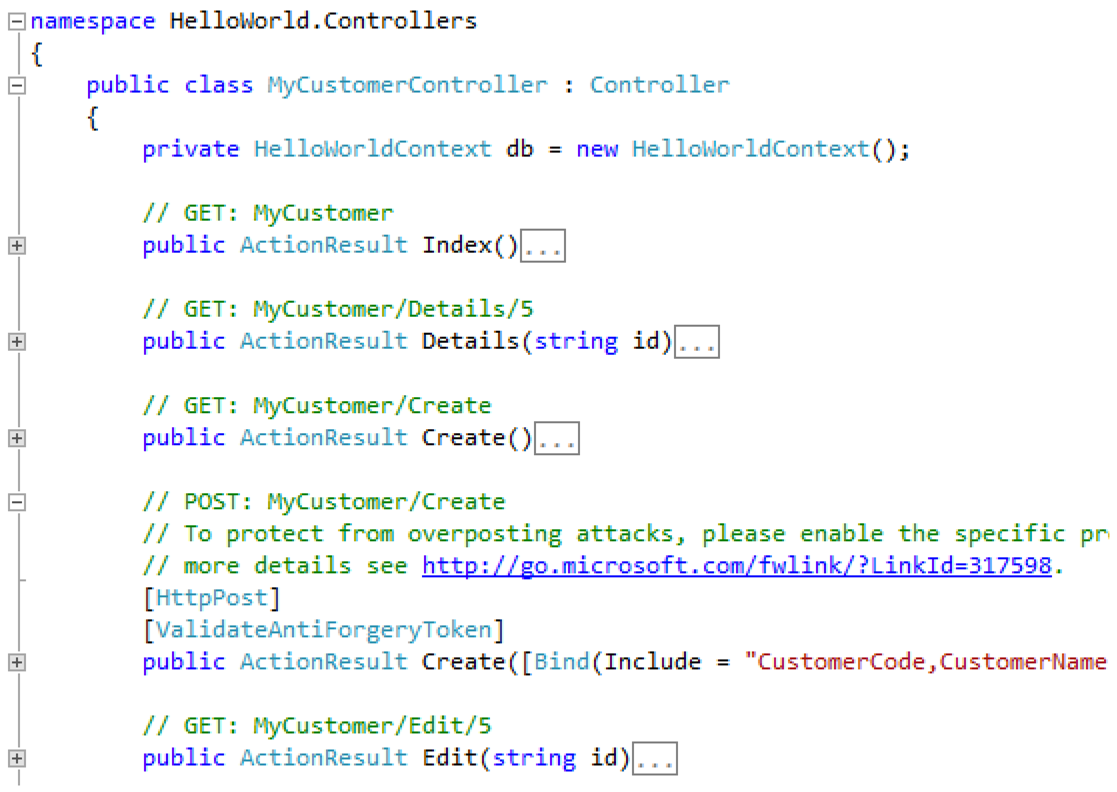
```
Note :- Do not get scared with the word. Its actually a very simple thing.
```

Scaffolding is a technique in which the MVC template helps to auto-generate CRUD code. CRUD stands for create, read, update and delete.

So to generate code using scaffolding technique we need to select one of the types of templates (leave the empty one).

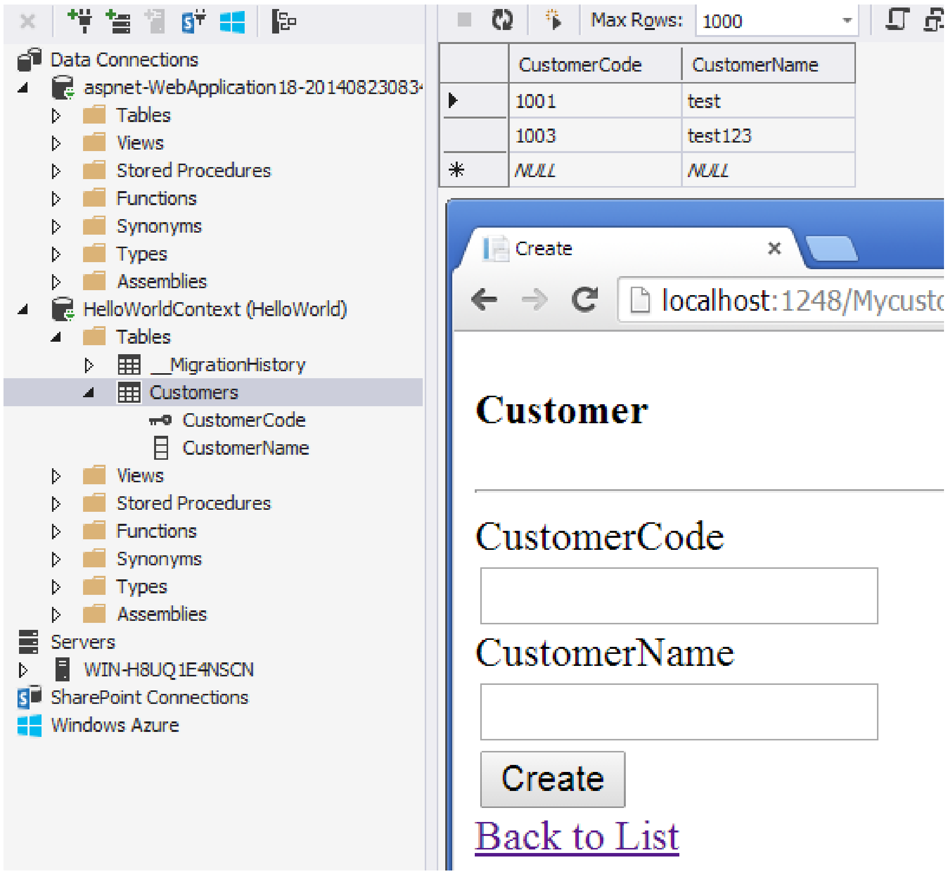

MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework

For instance if you choose “using Entity framework” template the following code is generated.



```
namespace HelloWorld.Controllers
{
    public class MyCustomerController : Controller
    {
        private HelloWorldContext db = new HelloWorldContext();

        // GET: MyCustomer
        public ActionResult Index()...

        // GET: MyCustomer/Details/5
        public ActionResult Details(string id)...

        // GET: MyCustomer/Create
        public ActionResult Create()...

        // POST: MyCustomer/Create
        // To protect from overposting attacks, please enable the specific pr
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "CustomerCode,CustomerName

        // GET: MyCustomer/Edit/5
        public ActionResult Edit(string id)...
```

It creates controller code, view and also table structure as shown in the below figure.



	CustomerCode	CustomerName
▶	1001	test
	1003	test123
*	NULL	NULL

Customer

CustomerCode

CustomerName

What does scaffolding use internally to connect to database?

It uses Entity framework internally.

How can you do authentication and authorization in MVC?

You can use windows or forms authentication for MVC.

How to implement windows authentication for MVC?

For windows authentication you need to go and modify the “web.config” file and set authentication mode to windows.

```
<authentication mode="Windows"/>
<authorization>
     <deny users="?"/>
</authorization>
```

Then in the controller or on the action you can use the “Authorize” attribute which specifies which users have access to these controllers and actions. Below is the code snippet for the same. Now only the users specified in the controller and action can access the same.

```
[Authorize(Users= @"WIN-3LI600MWLQN\Administrator")]
    public class StartController : Controller
    {
        //
        // GET: /Start/
        [Authorize(Users = @"WIN-3LI600MWLQN\Administrator")]
        public ActionResult Index()
        {
            return View("MyView");
        }

    }
```

How do you implement forms authentication in MVC?

Forms authentication is implemented the same way as we do in ASP.NET. So the first step is to set authentication mode equal to forms. The “loginUrl” points to a controller here rather than page.

```
<authentication mode="Forms">
<forms loginUrl="~/Home/Login"  timeout="2880"/>
```

```
</authentication>
```

We also need to create a controller where we will check the user is proper or not. If the user is proper we will set the cookie value.

```
public ActionResult Login()
{
if ((Request.Form["txtUserName"] == "Shiv") && (Request.Form["txtPassword"]
== "Shiv@123"))
{
            FormsAuthentication.SetAuthCookie("Shiv",true);
            return View("About");
}
else
{
            return View("Index");
}

}
```

All the other actions need to be attributed with "Authorize" attribute so that any unauthorized user if he makes a call to these controllers it will redirect to the controller (in this case the controller is "Login") which will do authentication.

```
[Authorize]
PublicActionResult Default()
{
return View();
}

[Authorize]
publicActionResult About()
{
return View();
}
```

How to implement Ajax in MVC?

You can implement Ajax in two ways in MVC:-

- Ajax libraries
- Jquery

Below is a simple sample of how to implement Ajax by using “Ajax” helper library. In the below code you can see we have a simple form which is created by using “Ajax.BeginForm” syntax. This form calls a controller action called as “getCustomer”. So now the submit action click will be an asynchronous ajax call.

```
<script language="javascript">
function OnSuccess(data1)
{
// Do something here
}
</script>
<div>
<%
        var AjaxOpt = new AjaxOptions{OnSuccess="OnSuccess"};
    %>
<% using (Ajax.BeginForm("getCustomer","MyAjax",AjaxOpt)) { %>
<input id="txtCustomerCode" type="text" /><br />
<input id="txtCustomerName" type="text" /><br />
<input id="Submit2" type="submit" value="submit"/></div>
<%} %>
```

In case you want to make ajax calls on hyperlink clicks you can use “Ajax.ActionLink” function as shown in the below code.

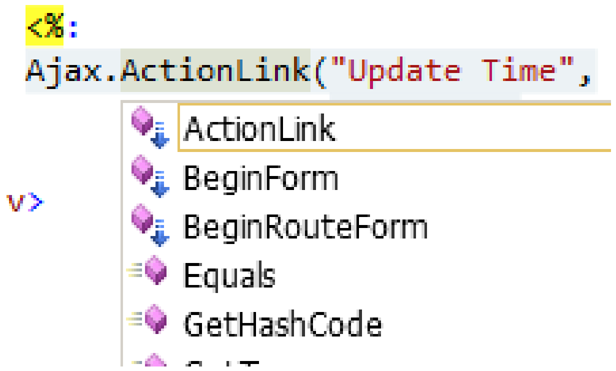

```
<%:
Ajax.ActionLink("Update Time",
```

Figure 6.8:- implement Ajax in MVC

So if you want to create Ajax asynchronous hyperlink by name “GetDate” which calls the “GetDate” function on the controller , below is the code for the same. Once the controller responds this data is displayed in the HTML DIV tag by name “DateDiv”.

```
<span id="DateDiv" />
<%:
Ajax.ActionLink("Get Date","GetDate",
new AjaxOptions {UpdateTargetId = "DateDiv" })
%>
```

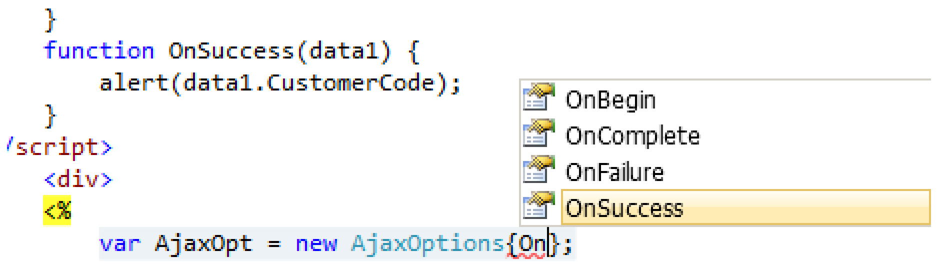
Below is the controller code. You can see how “GetDate” function has a pause of 10 seconds.

```
public class Default1Controller : Controller
{

       public string GetDate()
       {
           Thread.Sleep(10000);
           return DateTime.Now.ToString();
       }
}
```

The second way of making Ajax call in MVC is by using Jquery. In the below code you can see we are making an ajax POST call to a URL “/MyAjax/getCustomer”. This is done by using “$.post”. All this logic is put in to a function called as “GetData” and you can make a call to the “GetData” function on a button or a hyper link click event as you want.

```
function GetData()
    {
        var url = "/MyAjax/getCustomer";
        $.post(url, function (data)
        {

            $("#txtCustomerCode").val(data.CustomerCode);
            $("#txtCustomerName").val(data.CustomerName);
        }
        )
    }
```

What kind of events can be tracked in AJAX ?



```
    }
    function OnSuccess(data1) {
        alert(data1.CustomerCode);
    }
/script>
    <div>
    <%
        var AjaxOpt = new AjaxOptions{On};
```

Figure 6.9:- tracked in AJAX

What is the difference between “ActionResult” and “ViewResult”?

“ActionResult” is an abstract class while “ViewResult” derives from “ActionResult” class. “ActionResult” has several derived classes like “ViewResult” ,”JsonResult” , “FileStreamResult” and so on.

“ActionResult” can be used to exploit polymorphism and dynamism. So if you are returning different types of view dynamically “ActionResult” is the best thing. For example in the below code snippet you can see we have a simple action called as “DynamicView”. Depending on the flag (“IsHtmlView”) it will either return “ViewResult” or “JsonResult”.

```
public ActionResult DynamicView(bool IsHtmlView)
{
   if (IsHtmlView)
     return View(); // returns simple ViewResult
   else
     return Json(); // returns JsonResult view
}
```

What are the different types of results in MVC?

```
Note:-It’s difficult to remember all the 12 types. But some important ones
you can remember for the interview are “ActionResult”, “ViewResult” and
“JsonResult”. Below is a detailed list for your interest.
```

There 12 kinds of results in MVC, at the top is “ActionResult”class which is a base class that can have 11 subtypes’sas listed below: -

1. ViewResult - Renders a specified view to the response stream
2. PartialViewResult - Renders a specified partial view to the response stream
3. EmptyResult - An empty response is returned
4. RedirectResult - Performs an HTTP redirection to a specified URL
5. RedirectToRouteResult - Performs an HTTP redirection to a URL that is determined by the routing engine, based on given route data
6. JsonResult - Serializes a given object to JSON format
7. JavaScriptResult - Returns a piece of JavaScript code that can be executed on the client
8. ContentResult - Writes content to the response stream without requiring a view
9. FileContentResult - Returns a file to the client
10. FileStreamResult - Returns a file to the client, which is provided by a Stream
11. FilePathResult - Returns a file to the client

What are “ActionFilters”in MVC?

“ActionFilters” helps you to perform logic while MVC action is executing or after a MVC action has executed.

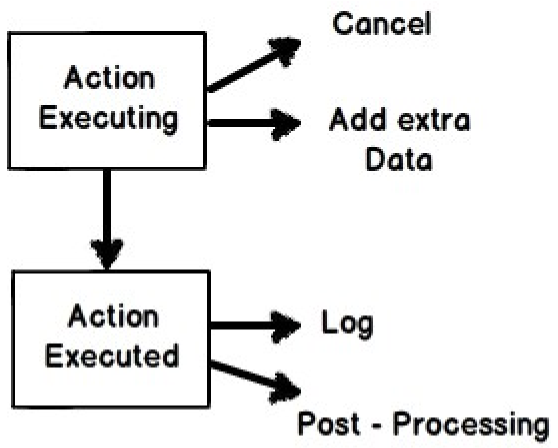


Figure 6.10:- “ActionFilters”in MVC

Action filters are useful in the following scenarios:-

1. Implement post-processinglogic beforethe action happens.
2. Cancel a current execution.

3. Inspect the returned value.
4. Provide extra data to the action.

You can create action filters by two ways:-

- Inline action filter.
- Creating an “ActionFilter” attribute.

To create a inline action attribute we need to implement “IActionFilter” interface.The “IActionFilter” interface has two methods “OnActionExecuted” and “OnActionExecuting”. We can implement pre-processing logic or cancellation logic in these methods.

```
public class Default1Controller : Controller , IActionFilter
    {
        public ActionResult Index(Customer obj)
        {
            return View(obj);
        }
        void IActionFilter.OnActionExecuted(ActionExecutedContext
filterContext)
        {
            Trace.WriteLine("Action Executed");
        }
        void IActionFilter.OnActionExecuting(ActionExecutingContext
filterContext)
        {
            Trace.WriteLine("Action is executing");
        }
    }
```

The problem with inline action attribute is that it cannot be reused across controllers. So we can convert the inline action filter to an action filter attribute. To create an action filter attribute we need to inherit from “ActionFilterAttribute” and implement “IActionFilter” interface as shown in the below code.

```
public class MyActionAttribute : ActionFilterAttribute , IActionFilter
{
void IActionFilter.OnActionExecuted(ActionExecutedContext filterContext)
{
     Trace.WriteLine("Action Executed");
}
void IActionFilter.OnActionExecuting(ActionExecutingContext filterContext)
{
      Trace.WriteLine("Action executing");
}
}
```

Later we can decorate the controllers on which we want the action attribute to execute. You can see in the below code I have decorated the "Default1Controller" with "MyActionAttribute" class which was created in the previous code.

```
[MyActionAttribute]
public class Default1Controller : Controller
{
 public ActionResult Index(Customer obj)
 {
 return View(obj);
 }
}
```

Can we create our custom view engine using MVC?

Yes, we can create our own custom view engine in MVC. To create our own custom view engine we need to follow 3 steps:-

Let' say we want to create a custom view engine where in the user can type a command like "<DateTime>" and it should display the current date and time.

Step 1:- We need to create a class which implements "IView" interface. In this class we should write the logic of how the view will be rendered in the "render" function. Below is a simple code snippet for the same.

```
public class MyCustomView : IView
    {

        private string _FolderPath; // Define where  our views are stored

        public string FolderPath
        {
            get { return _FolderPath; }
            set { _FolderPath = value; }
        }



        public void Render(ViewContext viewContext, System.IO.TextWriter
writer)
        {
           // Parsing logic <dateTime>
            // read the view file
            string strFileData = File.ReadAllText(_FolderPath);
            // we need to and replace <datetime> datetime.now value
            string strFinal = strFileData.Replace("<DateTime>",
DateTime.Now.ToString());
            // this replaced data has to sent for display
            writer.Write(strFinal);


        }
    }
```

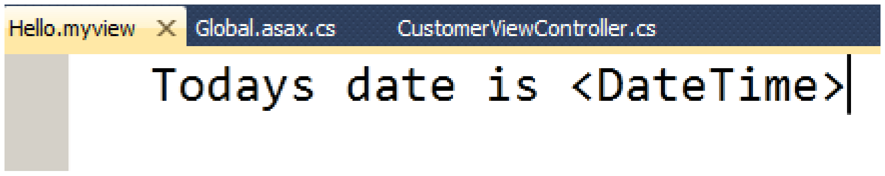
Step 2 :-We need to create a class which inherits from "VirtualPathProviderViewEngine" and in this class we need to provide the folder path and the extension of the view name. For instance for razor the extension is "cshtml" , for aspx the view extension is ".aspx" , so in the same way for our custom view we need to provide an extension. Below is how the code looks like. You can see the "ViewLocationFormats" is set to the "Views" folder and the extension is ".myview".

```
public class MyViewEngineProvider : VirtualPathProviderViewEngine
    {
        // We will create the object of Mycustome view
        public MyViewEngineProvider() // constructor
        {
            // Define the location of the View file
            this.ViewLocationFormats = new string[] { "~/Views/{1}/
{0}.myview", "~/Views/Shared/{0}.myview" }; //location and extension of our
views
        }


        protected override IView CreateView(ControllerContext
controllerContext, string viewPath, string masterPath)
        {
            var physicalpath = 
controllerContext.HttpContext.Server.MapPath(viewPath);
            MyCustomView obj = new MyCustomView(); // Custom view engine 
class
            obj.FolderPath = physicalpath; // set the path where the views 
will be stored
            return obj; // returned this view paresing logic so that it can 
be registered in the view engine collection
        }
        protected override IView CreatePartialView(ControllerContext 
controllerContext, string partialPath)
        {
            var physicalpath = 
controllerContext.HttpContext.Server.MapPath(partialPath);
            MyCustomView obj = new MyCustomView(); // Custom view engine 
class
            obj.FolderPath = physicalpath; // set the path where the views 
will be stored
            return obj; // returned this view paresing logic so that it can 
be registered in the view engine collection
        }
    }
```

Step 3:- We need to register the view in the custom view collection. The best place to register the custom view engine in the "ViewEngines" collection is the "global.asax" file. Below is the code snippet for the same.

```
protected void Application_Start()
 {
            // Step3 :-  register this object in the view engine collection
            ViewEngines.Engines.Add(new MyViewEngineProvider());
     …..
 }
```

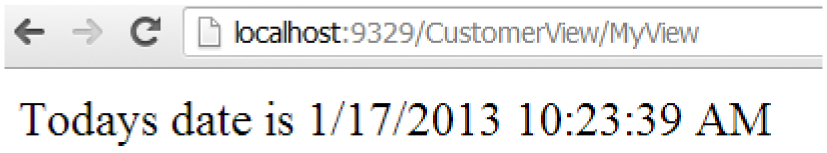
Below is a simple output of the custom view written using the commands defined at the top.



```
Todays date is <DateTime>
```

Figure6.11:-customviewengineusingMVC

If you invoke this view you should see the following output.



```
Todays date is 1/17/2013 10:23:39 AM
```

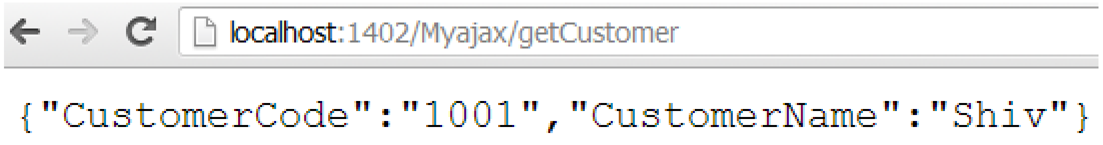
Figure 6.12:- output

How to send result back in JSON format in MVC?

In MVC we have “JsonResult” class by which we can return back data in JSON format. Below is a simple sample code which returns back “Customer” object in JSON format using “JsonResult”.

```
public JsonResult getCustomer()
{
Customer obj = new Customer();
obj.CustomerCode = "1001";
obj.CustomerName = "Shiv";
 return Json(obj,JsonRequestBehavior.AllowGet);
}
```

Below is the JSON output of the above code if you invoke the action via the browser.



```
{"CustomerCode":"1001","CustomerName":"Shiv"}
```

Figure 6.13:- JSON format in MVC

What is “WebAPI”?

HTTP is the most used protocol.For past many years browser was the most preferred client by which we can consume data exposed over HTTP. But as years passed by client variety started spreading out. We had demand to consume data on HTTP from clients like mobile,javascripts,windows application etc.

For satisfying the broad range of client “REST” was the proposed approach. You can read more about “REST” from WCF chapter.

“WebAPI” is the technology by which you can expose data over HTTP following REST principles.

How does WCF differ from WEB API ?

	WCF	**WEB API**
Multi-protocol hosting	Heavy weight because of complicated WSDL structure.	Light weight, only the necessary information is transferred.
Protocol	Independent of protocols.	Only for HTTP protocol
Formats	To parse SOAP message, the client needs to understand WSDL format. Writing custom code for parsing WSDL is a heavy duty task. If your client is smart enough to create proxy objects like how we have in .NET (add reference) then SOAP is easier to consume and call.	Output of “WebAPI” are simple string message,JSON,Simple XML format etc. So writing parsing logic for the same in very easy.
Principles	SOAP follows WS-* specification.	WEB API follows REST principles. (Please refer about REST in WCF chapter).

With WCF also you can implement REST,So why "WebAPI"?

WCF was brought in to implement SOA, never the intention was to implement REST."WebAPI'" is built from scratch and the only goal is to create HTTP services using REST. Due to the one point focus for creating “REST” service “WebAPI” is more preferred.

How to implement “WebAPI” in MVC?

Below are the steps to implement "webAPI" :-

Step1:-Create the project using the "WebAPI" template.

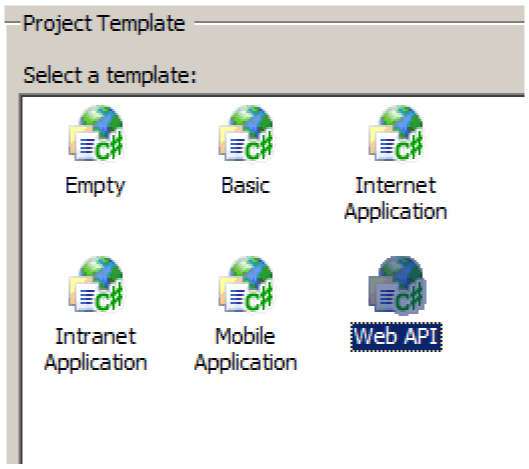
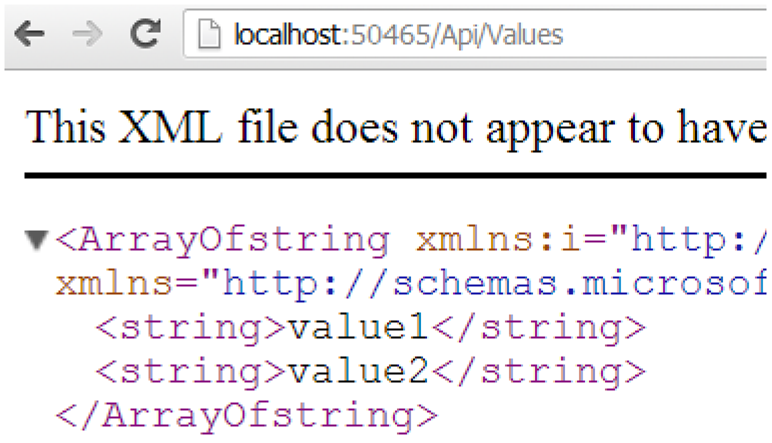


Figure 6.14:- implement "WebAPI" in MVC

Step 2:- Once you have created the project you will notice that the controller now inherits from "ApiController" and you can now implement "post","get","put" and "delete" methods of HTTP protocol.

```
public class ValuesController : ApiController
    {
        // GET api/values
        public IEnumerable<string> Get()
        {
            return new string[] { "value1", "value2" };
        }

        // GET api/values/5
        public string Get(int id)
        {
            return "value";
        }

        // POST api/values
        public void Post([FromBody]string value)
        {
        }

        // PUT api/values/5
        public void Put(int id, [FromBody]string value)
        {
        }

        // DELETE api/values/5
        public void Delete(int id)
        {
        }
    }
```

Step 3:-If you make a HTTP GET call you should get the below results.



localhost:50465/Api/Values

This XML file does not appear to have

```
▼<ArrayOfstring xmlns:i="http:/
 xmlns="http://schemas.microsof
   <string>value1</string>
   <string>value2</string>
 </ArrayOfstring>
```

Figure 6.15:- HTTP

How can we detect that a MVC controller is called by POST or GET ?

To detect if the call on the controller is "POST" action or a "GET" action we can use "Request.HttpMethod" property as shown in the below code snippet.

```
public ActionResult SomeAction()
{
            if (Request.HttpMethod == "POST")
            {
                return View("SomePage");
            }
            else
            {
                return View("SomeOtherPage");
            }


}
```

What is Bundling and minification in MVC?

Bundling and minification helps us to improve request load time of a page thus increasing performance.

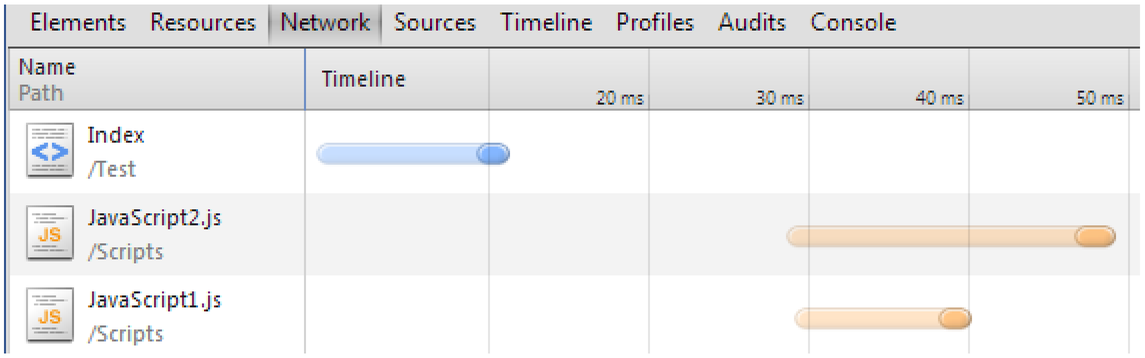
How does bundling increase performance?

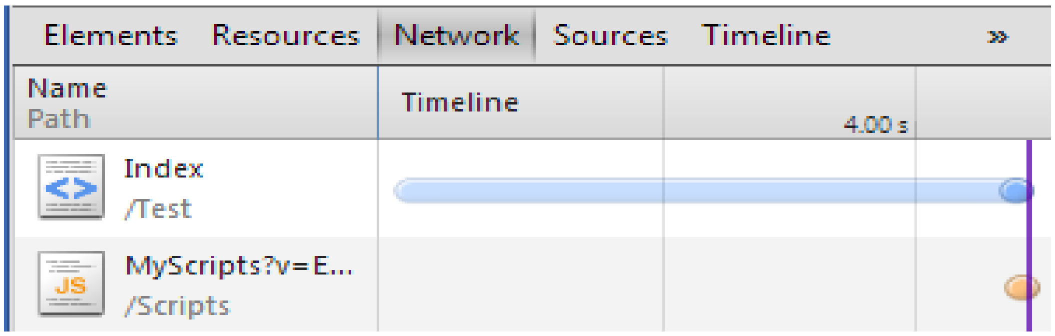
Web projects always need CSS and script files.Bundling helps us to combine to multiple javascript and CSS file in to a single entity thus minimizing multiple requests in to a single request.

For example consider the below web request to a page . This page consumes two JavaScript files “Javascript1.js” and “Javascript2.js”. So when this is page is requested it makes three request calls:-

- One for the “Index” page.
- Two requests for the other two JavaScript files “Javascript1.js” and “Javascript2.js”.

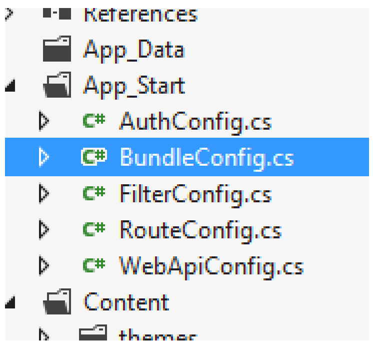
Below scenario can become worst if we have lot of javascript files resulting in many multiple requests thus decreasing performance. If we can somehow combine all the JS files in to a single bundle and request them as a single unit that would result in increased performance (See the next figure which has a single request).





So how do we implement bundling in MVC ?

Open the “BundleConfig.cs” from the “App_Start” folder.



In the “BundleConfig.cs” add the JS files which you want bundle in to single entity in to the bundles collection. In the below code we are combining all the javascript JS files which are existing in the “Scripts” folder as a single unit in to the bundle collection.

```
bundles.Add(new ScriptBundle("~/Scripts/MyScripts").Include(
"~/Scripts/*.js"));
```

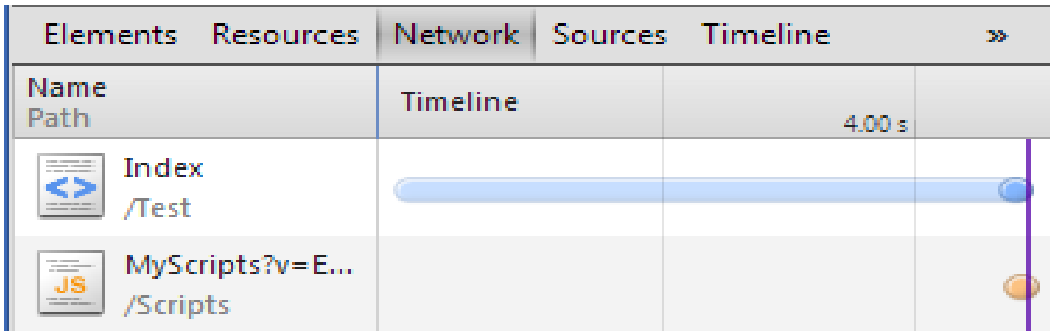
Below is how your “BundleConfig.cs” file will look like.

```
public  class BundleConfig
{
        public static void RegisterBundles(BundleCollection bundles)
        {


            bundles.Add(new ScriptBundle("~/Scripts/MyScripts").Include(
               "~/Scripts/*.js"));
            BundleTable.EnableOptimizations = true;
        }
}
```

Once you have combined your scripts in to one single unit we then to include all the JS files in to the view using the below code. The below code needs to put in the ASPX or RAZOR view.

```
<%= Scripts.Render("~/Scripts/MyScripts")  %>
```

If you now see your page requests you would see that script request is combined in to a one request.



How can you test bundling in debug mode ?

If you are in a debug mode you need to set "EnableOptimizations" to true in the "bundleconfig.cs" file or else you will not see the bundling effect in the page requests.

```
BundleTable.EnableOptimizations = true;
```

Explain minification and how to implement the same ?

Minification reduces the size of script and CSS files by removing blank spaces , comments etc. For example below is a simple javascript code with comments.

```
// This is test
var x = 0;
x = x + 1;
x = x * 2;
```

After implementing minification the javascript code looks something as below. You can see how whitespaces and comments are removed to minimize file size and thus increasing performance.

```
var x=0;x=x+1;x=x*2;
```

How to implement minification ?

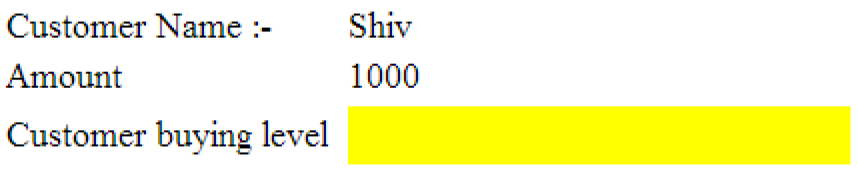
When you implement bundling minification is implemented by itself. In other words steps to implement bundling and minification are same.

Explain the concept of View Model in MVC ?

A view model is a simple class which represents data to be displayed on the view.
For example below is a simple customer model object with “CustomerName” and “Amount” property.

```
CustomerViewModel obj = new CustomerViewModel();
obj.Customer.CustomerName = "Shiv";
obj.Customer.Amount = 1000;
```

But when this “Customer” model object is displayed on the MVC view it looks something as shown in the below figure. It has “CustomerName” , “Amount” plus “**Customer Buying Level**” fields on the view / screen. “Customer buying Level” is a color indication which indicates how aggressive the customer is buying.
“Customer buying level” color depends on the value of the “Amount property. If the amount is greater than 2000 then color is red , if amount is greater than 1500 then color is orange or else the color is yellow.
In other words “Customer buying level” is an extra property which is calculated on the basis of amount.



Customer Name :-	Shiv
Amount	1000
Customer buying level	🟨

So the Customer viewmodel class has three properties

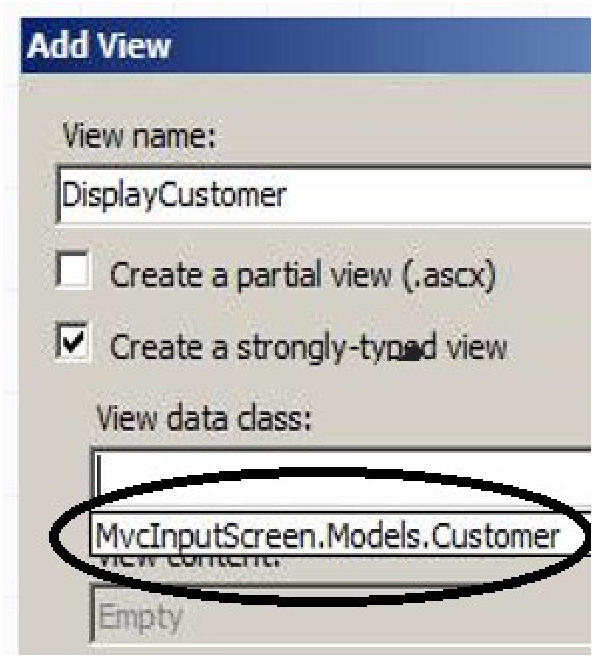
- “TxtCustomerName” textbox takes data from “CustomerName” property as it is.
- “TxtAmount” textbox takes data from “Amount” property of model as it is.
- “CustomerBuyingLevelColor” displays color value depending on the “Amount “ value.

Customer Model	**Customer ViewModel**

CustomerName	TxtCustomerName
Amount	TxtAmount
	CustomerBuyingLevelColor

How can we use two (multiple) models with a single view?

Let us first try to understand what the interviewer is asking. When we bind a model with a view we use the model dropdown as shown in the below figure. In the below figure we can only select one model.


Add View
View name:
DisplayCustomer
Create a partial view (.ascx)
Create a strongly-typed view
View data class:
MvcInputScreen.Models.Customer
Empty

But what if we want to bind “Customer” as well as “Order” class to the view.

For that we need to create a view model which aggregates both the classes as shown in the below code. And then bind that view model with the view.

```
public class CustOrderVM
{
public  Customer cust = new Customer();
public Order Ord = new Order();
}
```

In the view we can refer both the model using the view model as shown in the below code.

```
<%= model.cust.Name %>
<%= model.Ord.Number %>
```

What kind of logic view model class will have ?

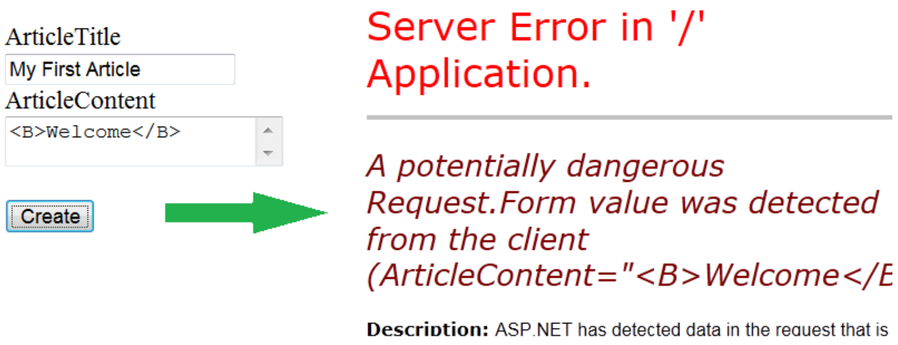
As the name says view model this class has the gel code or connection code which connects the view and the model.

So the view model class can have following kind of logics:-

- **Color transformation logic**: - For example you have a “Grade” property in model and you would like your UI to display “red” color for high level grade, “yellow” color for low level grade and “green” color of ok grade.
- **Data format transformation logic** :- Your model has a property “Status” with “Married” and “Unmarried” value. In the UI you would like to display it as a checkbox which is checked if “married” and unchecked if “unmarried”.
- **Aggregation logic: -** You have two different Customer and Address model classes and you have view which displays both “Customer” and “Address” data on one go.
- **Structure downsizing: -** You have “Customer” model with “customerCode” and “CustomerName” and you want to display just “CustomerName”. So you can create a wrapper around model and expose the necessary properties.

What is the use of “AllowHTML” and “ValidateInput” attributes?

While working with MVC you may have noticed that MVC actions doesn’t allow posting HTML values . This is done to avoid cross site scripting security issues.Look at the following example where we are trying to post HTML to the MVC action.



Server Error in '/' Application.

A potentially dangerous Request.Form value was detected from the client (ArticleContent="<B>Welcome</E

Description: ASP.NET has detected data in the request that is

But you can bypass this restriction by using one of the following 2 attributes

1) ValidateInput attribute at controller level or action level

```
[HttpPost()]
[ValidateInput(false)]
public ActionResult Index(ArticleData o)
{
    return View(o);
}
```

2) AllowHtml attribute at Mode Level

```
public class ArticleData
{
    [AllowHtml()]
    public string ArticleContent { get; set; }
    public string ArticleTitle { get; set; }
}
```

Explain unobtrusive JavaScript?

Unobtrusive JavaScript helps you to decouple presentation and behavior. Consider the below button code it has two parts one is the UI i.e. the button itself and second is the behavior i.e. “ShowAlert()”.

In other words the button is tied up with the “ShowAlert” action. It would be great if we could decouple the behavior from the button so that any other behavior can be attached with the button.

```
<input type="button" onclick="ShowAlert()" id="btn" />
<script>
function ShowAlert()
{
alert("Hello");
}
</script>
```

Look at this button code you can see no action is attached with the below button element.

```
<input type="button" id="btn" />
```

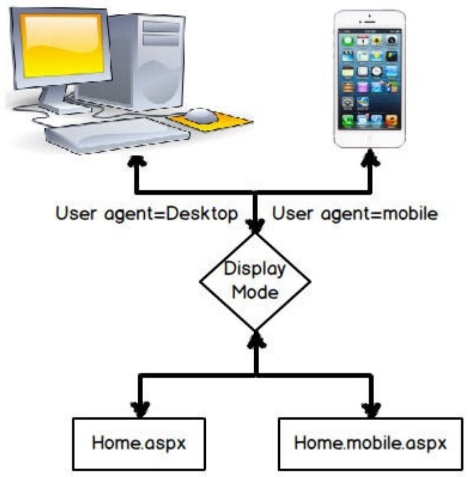
Action to the button is attached on runtime as shown in the below code. If needed tomorrow we can attach some other behavior with “btn” button . So unobtrusive javascript is nothing but decoupling the presentation from the behavior.

```
<script>
function ShowAlert()
{
alert("Hello");
}
var el = document.getElementById("btn");
el.onclick = ShowAlert;
</script>
```

Explain the need of display mode in MVC?

Display mode displays views depending on the device the user has logged in with. So we can create different views for different devices and display mode will handle the rest.

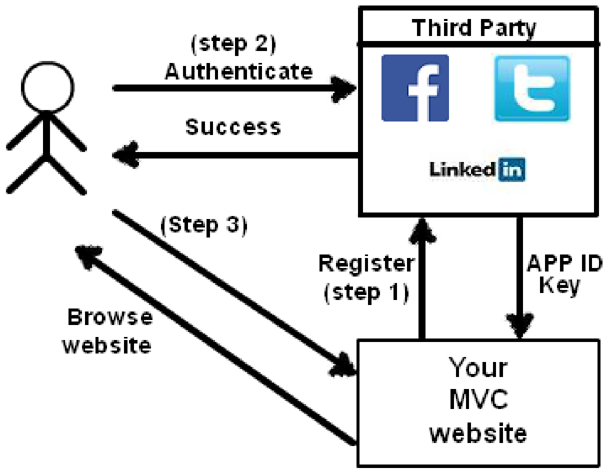
For example we can create a view “Home.aspx” which will render for the desktop computers andHome.Mobile.aspx for mobile devices. Now when an end user sends a request to the MVC application, display mode checks the “user agent” headers and renders the appropriate view to the device accordingly.



How can we validate using facebook or twitter accounts (MVC OAuth) ?

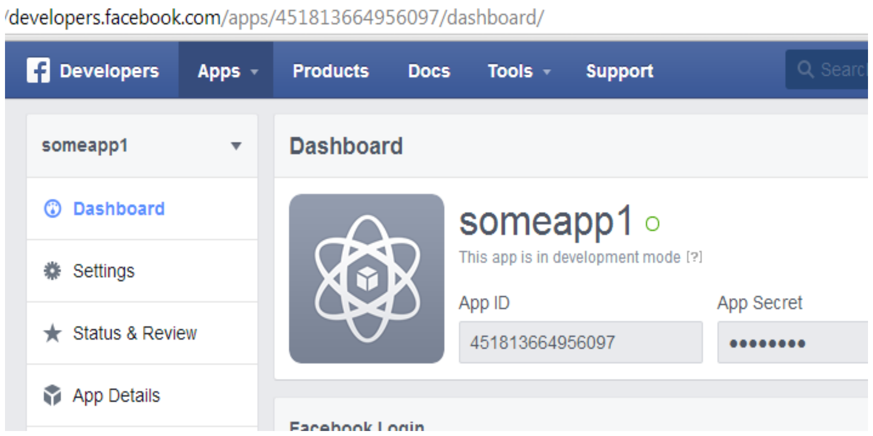
One of the most boring process for an end user is registering on a site. Sometimes those long forms and email validation just puts off the user. So how about making things easy by validating the users using their existing facebook / twitter / linkedin / etc accounts. So the user uses something which he already has while the site is assured that this user is a proper user.

This is achieved by using MVC OAuth (Open standard for authorization).



Using MVC Oauth is a three step process:-

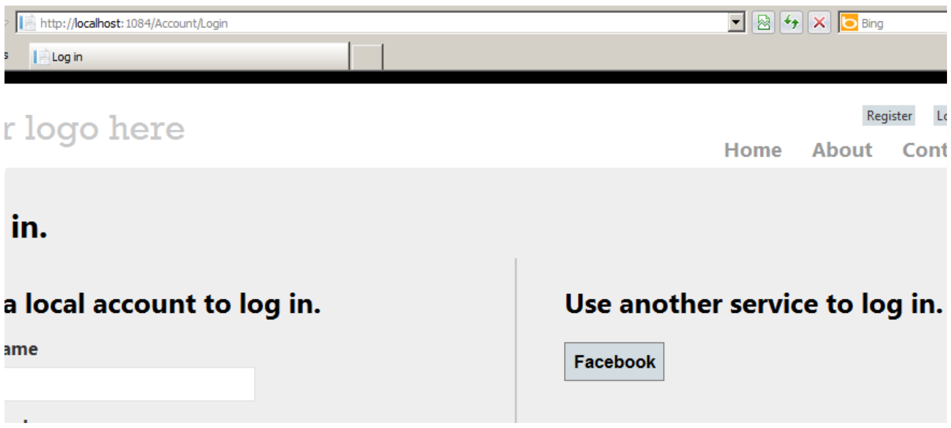
- Register your MVC application / website with the external site i.e. facebook , twitter etc. Once you register your app you get an ID and key from the external site. Below is snap shot of how facebook gives the ID and Key. In FB they term the key as the “App secret”. This process varies from site to site. So FB can have X steps while twitter can X+1.



- Next step is to open “AuthConfig.cs” and you will see lot of readymade code to user your ID and Key. So use the appropriate method as per site and provide the ID and Key.

```
OAuthWebSecurity.RegisterMicrosoftClient(
                clientId: "",
                clientSecret: "");

OAuthWebSecurity.RegisterTwitterClient(
                consumerKey: "",
                consumerSecret: "");

OAuthWebSecurity.RegisterFacebookClient(
                appId: "",
                appSecret: "");
OAuthWebSecurity.RegisterGoogleClient();
```

- Now if you run your application you should get a link to FB login as shown in the below screen. So if you login in FB it will redirect to your MVC application.


http://localhost:1084/Account/Login
Log in
r logo here
Register
Home
About
Cont
in.
a local account to log in.
ame
Use another service to log in.
Facebook

What is new in MVC 5 ?

Below are the new features in MVC 5 :-

- One ASP.NET
- Attribute based routing
- Asp.Net Identity
- Bootstrap in the MVC template
- Authentication Filters
- Filter overrides
- New Scaffolding system