ASP.NET MVC interview questions with answers

Shivprasad koirala, Marla Sukesh, 23 Dec 2014 CPOL

★★★★★ 4.87 (485 votes)

The whole purpose of this article is to quickly brush up your MVC knowledge from ASP.NET MVC interview perspective.

Download 100 MVC Interview questions with answers Ebook - 771.8 KB

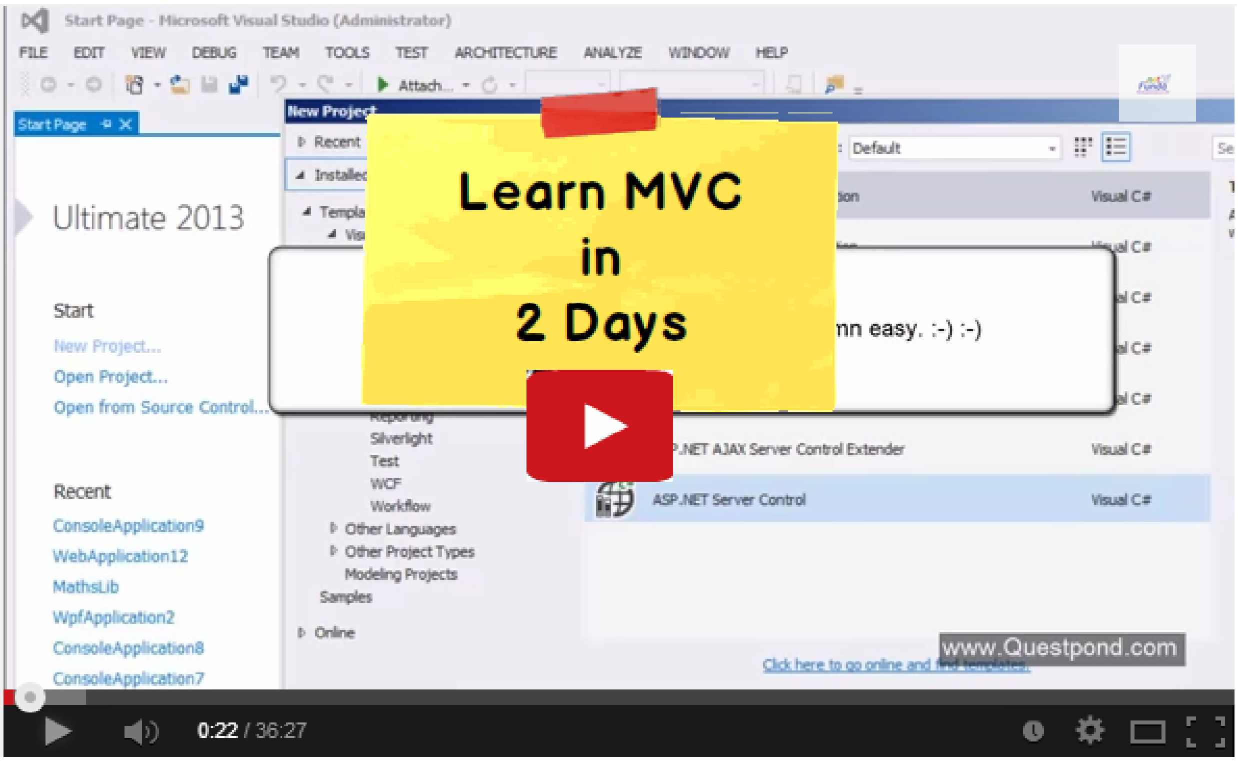
Table of content

Disclaimer

Reading these MVC interview questions does not mean you will go and clear MVC interviews. The purpose of this article is to quickly brush up your MVC knowledge before you go for MVC interviews. This article does not teach MVC, it’s a last minute revision sheet before going for MVC interviews.

If you want to learn MVC from scratch, start by reading Learn MVC (Model view controller) step by step 7 days or you can also start with my step by step MVC (Model View Controller) video series from YouTube.

If you want to learn MVC 5 in a short time i.e. 2 days a.k.a 16 hours below is a video series for the same.



Need help to improve this article

I have tried my level best to cover what questions i have faced in MVC interviews. But i feel the below questions are not enough and in real MVC interview's much more is asked. If you can share your question in the comment below. I would love to incorporate them in this article so that others are benefited.
If your question is great and i like it i will ship you a free copy of my .NET interview question book only in India (sorry i am not so rich for outside countries).

What is MVC (Model View Controller)?

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

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

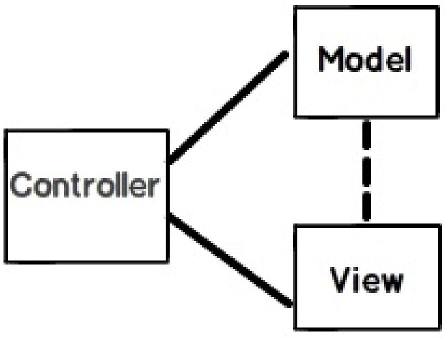


Figure: MVC (Model view controller)

Explain MVC application life cycle?

There are six broader events which occur in MVC application life cycle below diagrams summarize it.

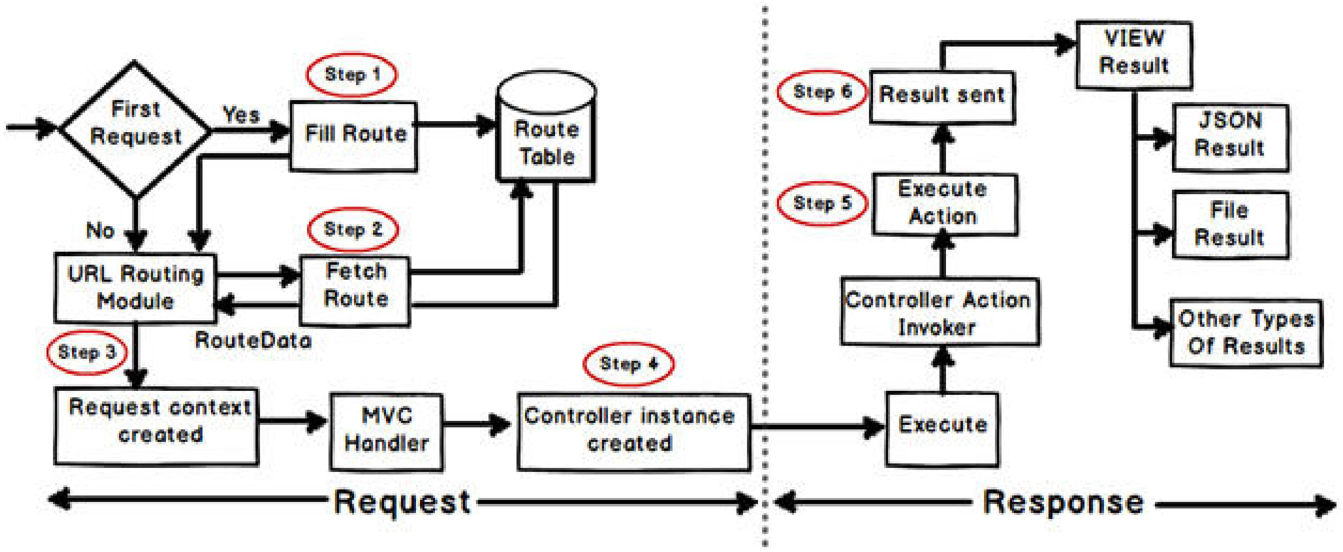


Image Courtesy: - http://www.dotnetinterviewquestions.in/article_explain-mvc-application-life-cycle_210.html

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.

Is MVC suitable for both Windows and Web applications?

The MVC architecture is suited for a web application than Windows. For Window applications, MVP, i.e., “Model View Presenter” is more applicable. If you are using WPF and Silverlight, MVVM is more suitable due to bindings.

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 three layered architecture?

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

Functionality	**Three 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

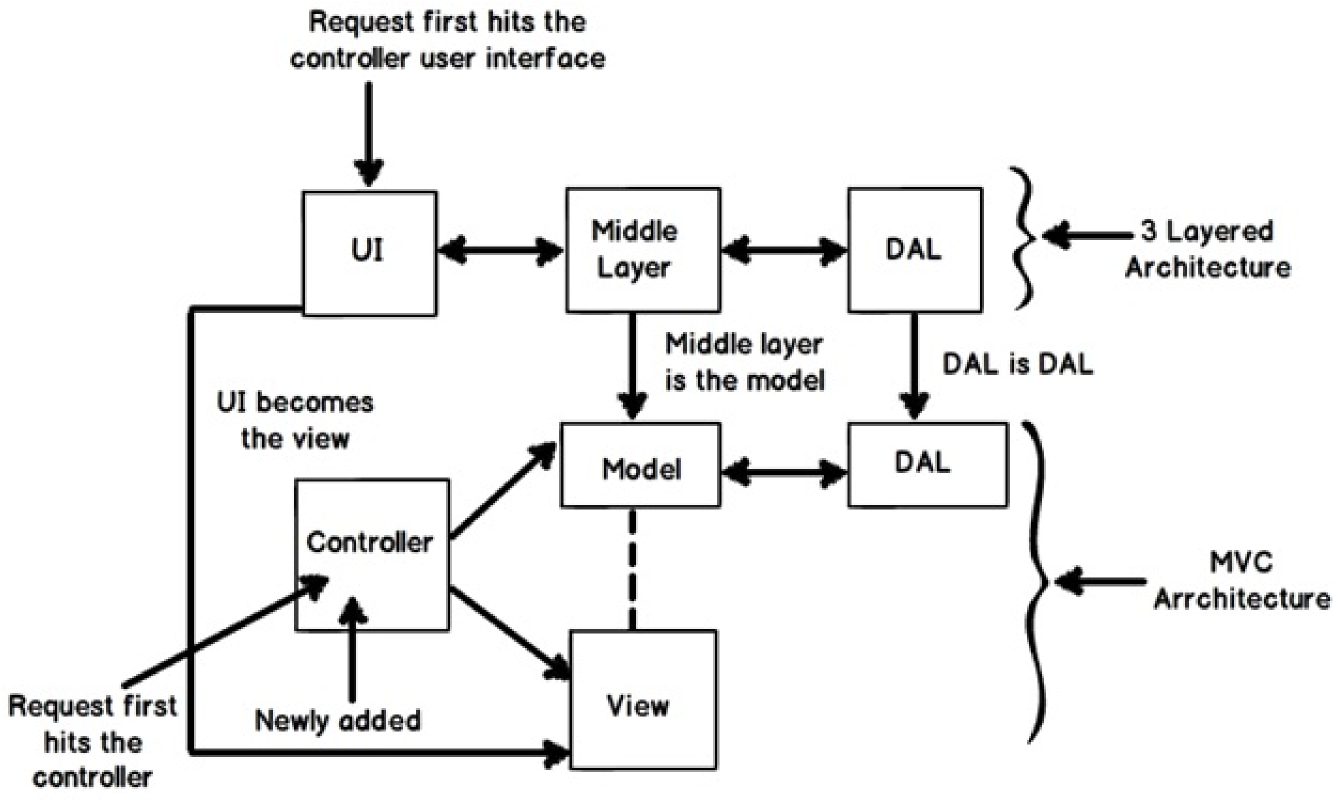


Figure: Three layered architecture

What is the latest version of MVC?

MVC 6 is the latest version which is also termed as ASP VNEXT.

What is the difference between each version of MVC 2, 3 , 4, 5 and 6?

MVC 6

ASP.NET MVC and Web API has been merged in to one.

Dependency injection is inbuilt and part of MVC.

Side by side - deploy the runtime and framework with your application

Everything packaged with NuGet, Including the .NET runtime itself.

New JSON based project structure.

No need to recompile for every change. Just hit save and refresh the browser.

Compilation done with the new Roslyn real-time compiler.

vNext is Open Source via the .NET Foundation and is taking public contributions.

vNext (and Rosyln) also runs on Mono, on both Mac and Linux today.

MVC 5

One ASP.NET

Attribute based routing

Asp.Net Identity

Bootstrap in the MVC template

Authentication Filters

Filter overrides

MVC 4

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

MVC 3

Razor

Readymade project templates

HTML 5 enabled templates

Support for Multiple View Engines

JavaScript and Ajax

Model Validation Improvements

MVC 2

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

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”.

What is routing in MVC?

Routing helps you to define a URL structure and map the URL with the controller.

For instance let’s say we want that when a user types “*http://localhost/View/ViewCustomer/*”, it goes to the “Customer” Controller and invokes the `DisplayCustomer` action. This is defined by adding an entry in to the `routes` collection using the `maproute` function. Below is the underlined 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 "RouteConfig.cs" file and registered using "global.asax" application start event.

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.

Explain attribute based routing in MVC?

This is a feature introduced in MVC 5. By using the "Route" attribute we can define the URL structure. For example in the below code we have decorated the "GotoAbout" action with the route attribute. The route attribute says that the "GotoAbout" can be invoked using the URL structure "Users/about".

```
public class HomeController : Controller
{
       [Route("Users/about")]
       public ActionResult GotoAbout()
       {
           return View();
       }
}
```

What is the advantage of defining route structures in the code?

Most of the time developers code in the action methods. Developers can see the URL structure right upfront rather than going to the “routeconfig.cs” and see the lengthy codes. For instance in the below code the developer can see right upfront that the “GotoAbout” action can be invoked by four different URL structure.

This is much user friendly as compared to scrolling through the “routeconfig.cs” file and going through the length line of code to figure out which URL structure is mapped to which action.

```
public class HomeController : Controller
{
       [Route("Users/about")]
       [Route("Users/WhoareWe")]
       [Route("Users/OurTeam")]
       [Route("Users/aboutCompany")]
       public ActionResult GotoAbout()
       {
           return View();
       }
}
```

How can we navigate from one view to another using a hyperlink?

By using the `ActionLink` method as shown in the below code. The below code will create a simple URL which helps 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 with the `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 sessions in MVC?

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

What is the difference between tempdata, viewdata, and viewbag?

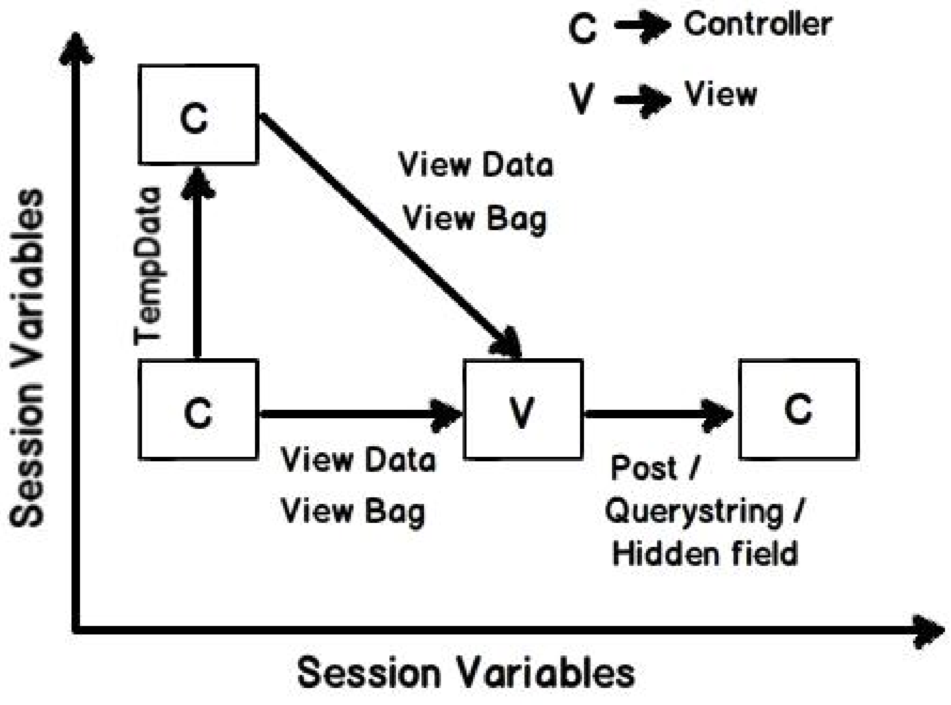


Figure: Difference between tempdata, viewdata, and viewbag

- **Temp data** - Helps to maintain data when you move from one controller to another controller or from one action to another 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.

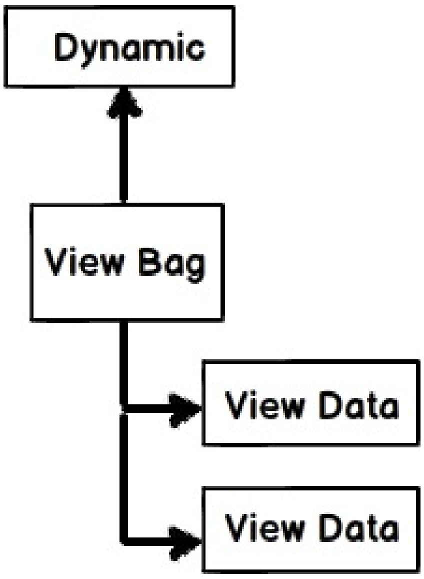


Figure: dynamic keyword

- **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 the different mechanisms for 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 difference between TempData and ViewData ?

“TempData” maintains data for the complete request while “ViewData” maintains data only from Controller to the view.

Does “TempData” preserve data in the next request also?

“TempData” is available through out 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[&ldquo;MyData&rdquo;];
TempData.Keep(&ldquo;MyData&rdquo;);
```

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();
```

If you want to read more in detail you can read from this detailed blog on MVC Peek and Keep.

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.

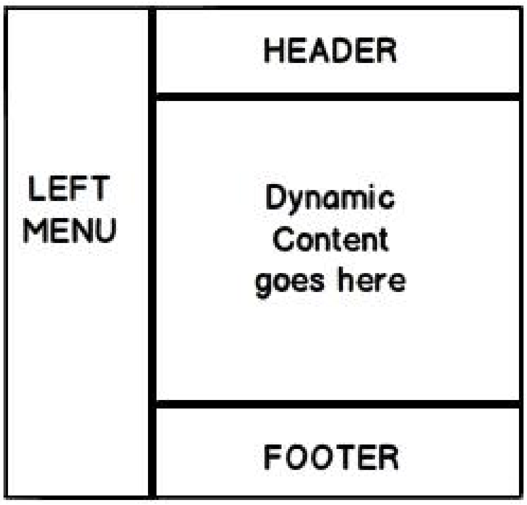
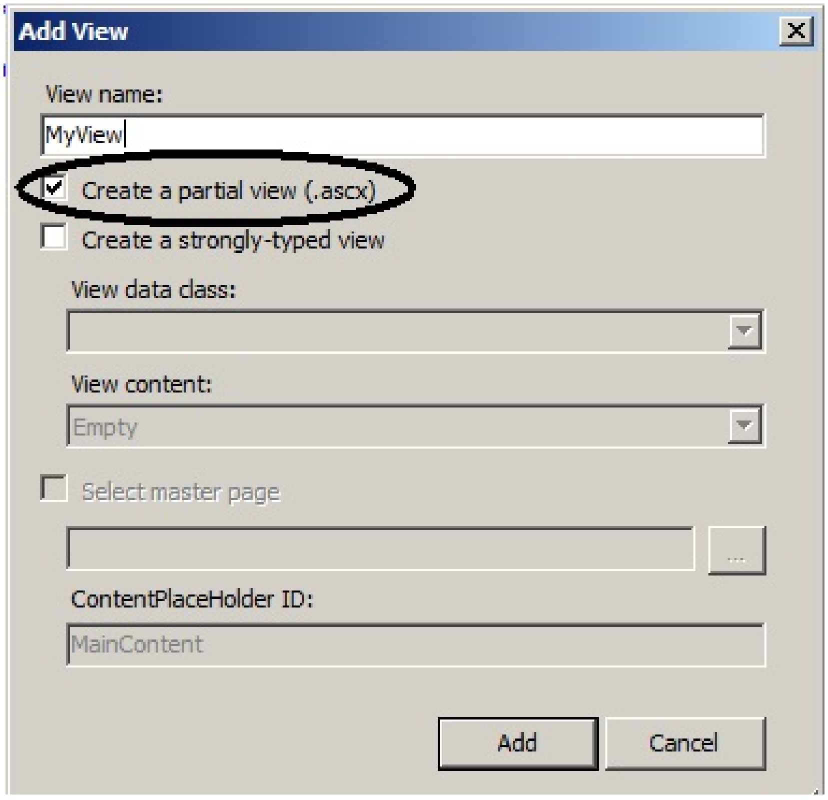


Figure: Partial views in MVC

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 a partial view and consume it?

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


Add View

View name: MyView

[✓] Create a partial view (.ascx)

[ ] Create a strongly-typed view

View data class:

View content: Empty

[ ] Select master page

ContentPlaceHolder ID: MainContent

Figure: Create partial view

Once the partial view is created you can then call the partial view in the main view using the `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 easiest ways of doing validation in MVC is by using data annotations. Data annotations are nothing but attributes which can be applied on 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 it.

```
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 the `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 the `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.

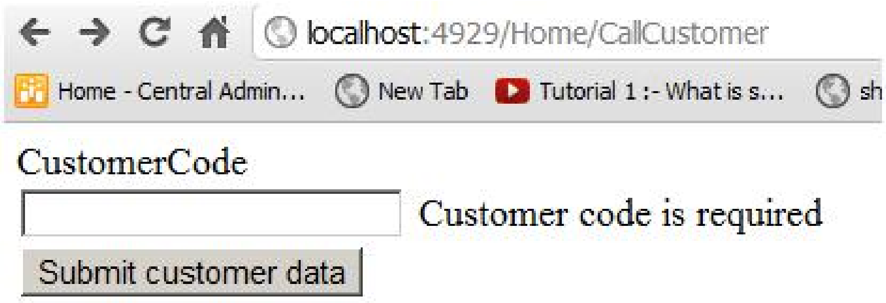

localhost:4929/Home/CallCustomer
Home - Central Admin...
New Tab
Tutorial 1 :- What is s...
sh
CustomerCode
Customer code is required
Submit customer data

Figure: Validations in MVC

Can we display all errors in one go?

Yes, we can; use the `ValidationSummary` method from the `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 a regular expression, you can use the `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; }
```

Sometimes you would like to compare the value of one field with another 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 the `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>
```

The second step is to call the `EnableClientValidation` method.

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

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 have 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?

As per Microsoft, Razor is more preferred because it's light weight and has simple syntaxes.

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 modify the *web.config* file and set the 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 that. Now only the users specified in the controller and action can access it.

```
[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 in ASP.NET. The first step is to set the authentication mode equal to `Forms`. The `loginUrl` points to a controller here rather than a page.

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

We also need to create a controller where we will check if 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 the `Authorize` attribute so that any unauthorized user making a call to these controllers will be redirected to the controller (in this case the controller is "Login") which will do the 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:

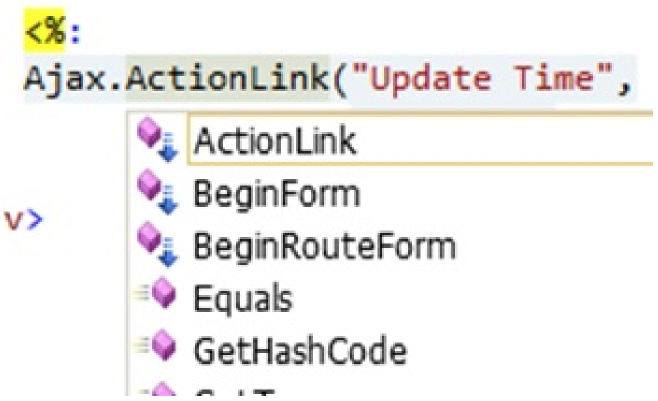
- AJAX libraries
- jQuery

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

```
<script language="javascript">
```

```
function OnSuccess(data1)
{
// Do something here
}
</script>
```

In case you want to make AJAX calls on hyperlink clicks, you can use the `Ajax.ActionLink` function as shown in the below code.



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

Figure: Implement AJAX in MVC

So if you want to create an AJAX asynchronous hyperlink by name `GetDate` which calls the `GetDate` function in the controller, below is the code for that. Once the controller responds, this data is displayed in the HTML `DIV` tag named `DateDiv`.

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

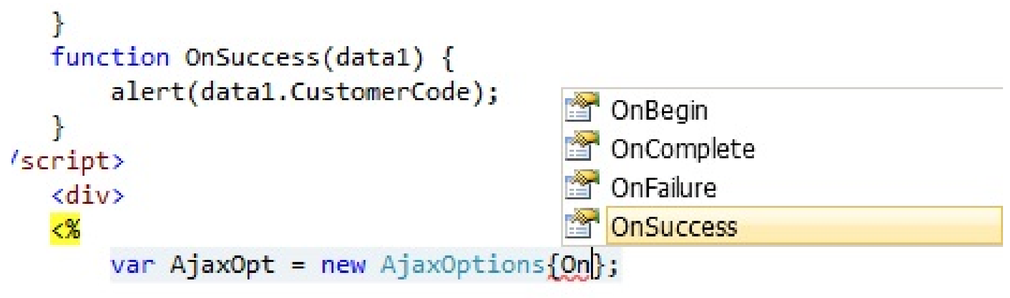
Below is the controller code. You can see how the `GetDate` function has a pause of 10 seconds.

```
public class Default1Controller : Controller
{
   public string GetDate()
   {
       Thread.Sleep(10000);
       return DateTime.Now.ToString();
   }
}
```

The second way of making an 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 into a function called `GetData` and you can make a call to the `GetData` function on a button or a hyperlink 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: Tracked in AJAX

What is the difference between ActionResult and ViewResult?

- `ActionResult` is an abstract class while `ViewResult` derives from the `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 views dynamically, `ActionResult` is the best thing. For example in the below code snippet, you can see we have a simple action called `DynamicView`. Depending on the flag (`IsHtmlView`) it will either return a `ViewResult` or `JsonResult`.

```
public ActionResult DynamicView()
{
   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 the `ActionResult` class which is a base class that can have 11 subtypes as 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 `ViewData` 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 help you to perform logic while an MVC action is executing or after an MVC action has executed.

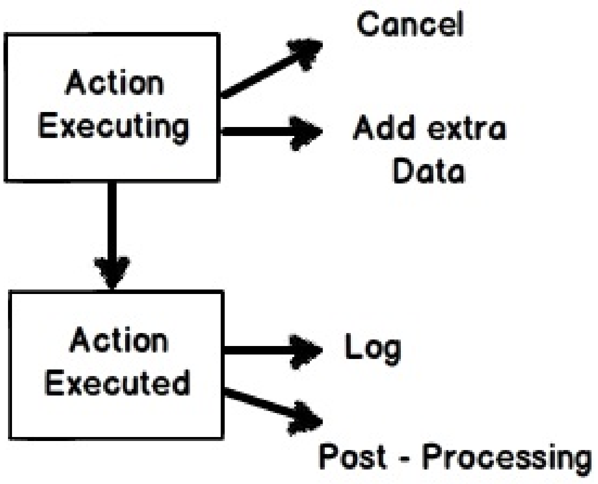


Figure: ActionFilters in MVC

Action filters are useful in the following scenarios:

1. Implement post-processing logic before the 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 an inline action attribute we need to implement the `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 the 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 the `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 the `MyActionAttribute` class which was created in the previous code.

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

What are the different types of action filters?

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

If we have multiple filters, what's the sequence for execution?

1. Authorization filters
2. Action filters
3. Response filters
4. Exception filters

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 three 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 the `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 that.

```
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);
    }
}
```

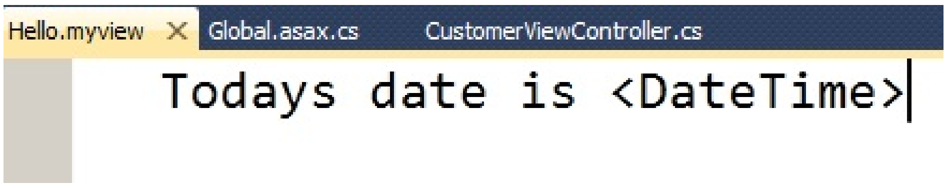
Step 2: We need to create a class which inherits from `VirtualPathProviderViewEngine` and in this class we need to provide the folder path and the extension of the view name. For instance, for Razor the extension is “cshtml”; for aspx, the view extension is “.aspx”, so in the same way for our custom 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 that.

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

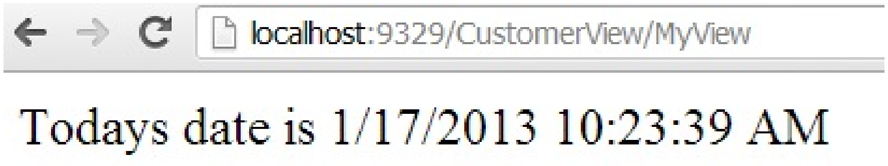
Below is a simple output of the custom view written using the commands defined at the top.



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

Figure: Custom view engine using MVC

If you invoke this view, you should see the following output:


localhost:9329/CustomerView/MyView

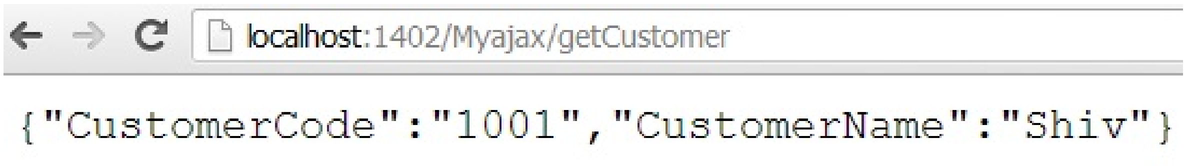
Todays date is 1/17/2013 10:23:39 AM

How to send result back in JSON format in MVC

In MVC, we have the `JsonResult` class by which we can return back data in JSON format. Below is a simple sample code which returns back a `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.


localhost:1402/Myajax/getCustomer

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

What is WebAPI?

HTTP is the most used protocol. For the past many years, browser was the most preferred client by which we consumed 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, JavaScript, Windows applications, etc.

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

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

But WCF SOAP also does the same thing, so how does WebAPI differ?

	SOAP	**WEB API**
Size	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 messages, JSON, simple XML format, etc. So writing parsing logic for that is very easy.
Principles	SOAP follows WS-* specification.	WebAPI follows REST principles. (Please refer to REST in WCF chapter.)

With WCF you can implement REST, so why WebAPI?

WCF was brought into implement SOA, the intention was never 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:

Step 1: Create the project using the WebAPI template.

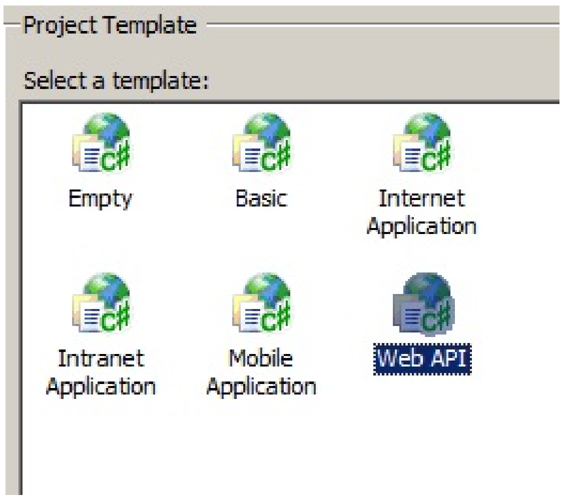
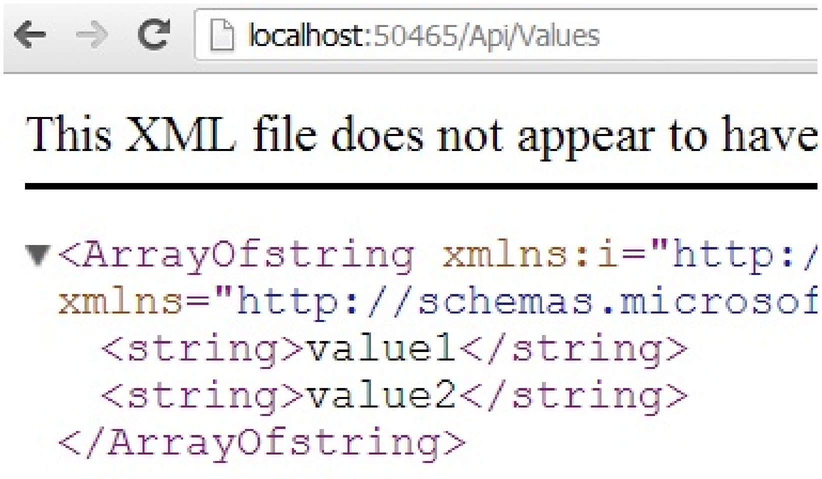


Figure: Implement WebAPI in MVC

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 the 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 an 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: HTTP

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

To detect if the call on the controller is a POST action or a GET action we can use the `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 improve request load times of a page thus increasing performance.

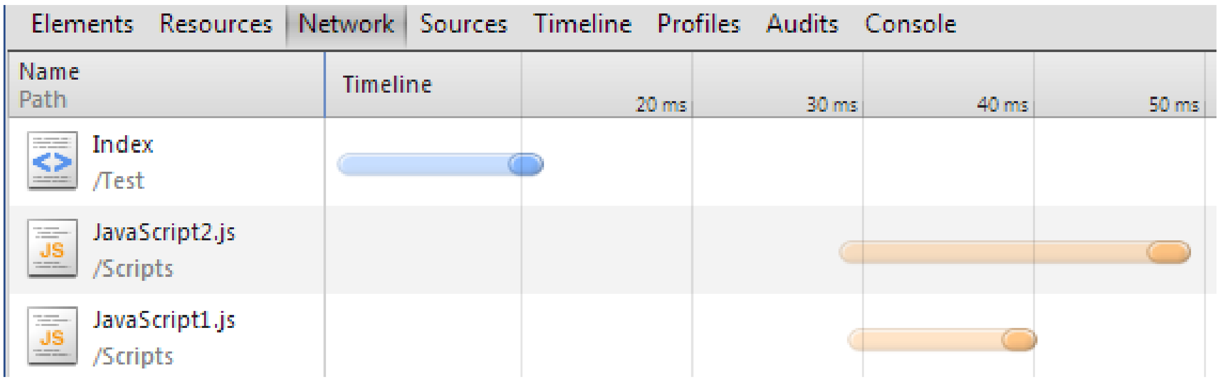
How does bundling increase performance?

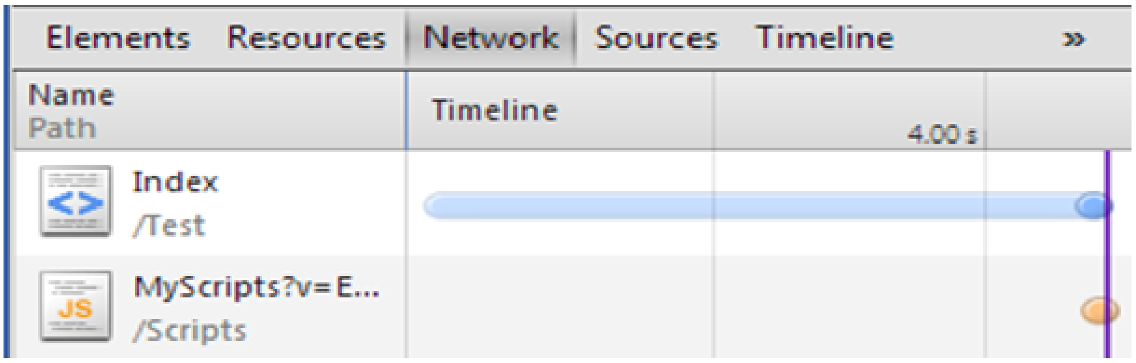
Web projects always need CSS and script files. Bundling helps us combine multiple JavaScript and CSS files in to a single entity thus minimizing multiple requests in to a single request.

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*.

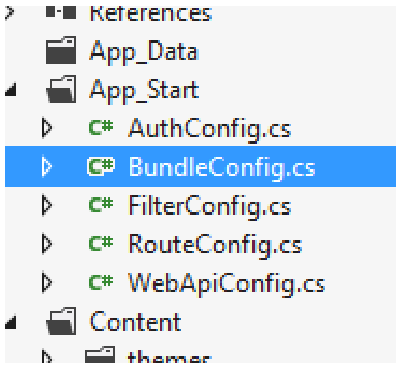
The below scenario can become worse if we have a lot of JavaScript files resulting in multiple requests, thus decreasing performance. If we can somehow combine all the JS files into a single bundle and request them as a single unit that would result in increased performance (see the next figure which has a single request).





So how do we implement bundling in MVC?

Open *BundleConfig.cs* from the *App_Start* folder.



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

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

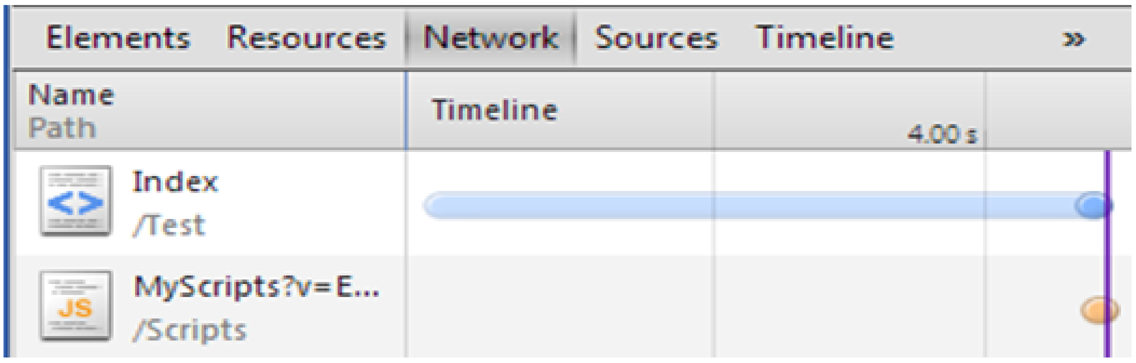
Below is how your *BundleConfig.cs* file will look like:

```
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 into one single unit we then to include all the JS files into the view using the below code. The below code needs to be 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 into 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 it

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 like below. You can see how whitespaces and comments are removed to minimize file size, thus increasing performance.

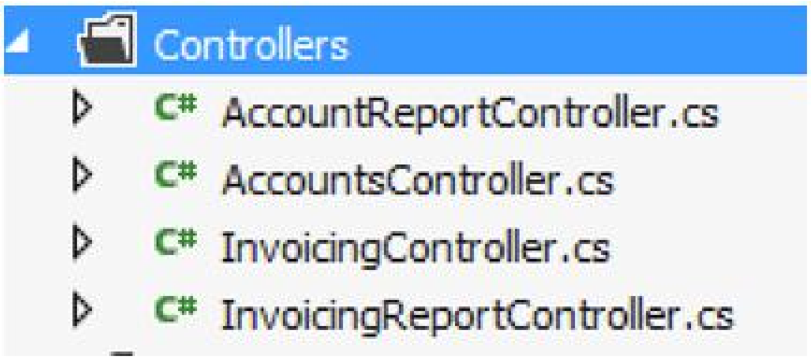
```
var x=0;x=x+1;x=x*2;
```

How do we implement minification?

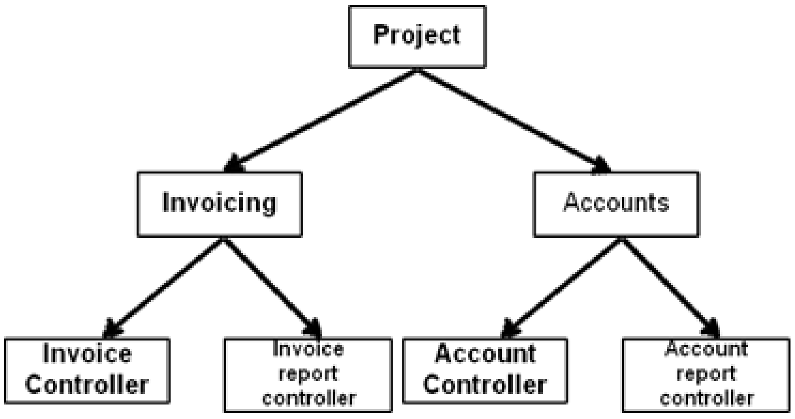
When you implement bundling, minification is implemented by itself. In other words the steps to implement bundling and minification are the same.

Explain Areas in MVC?

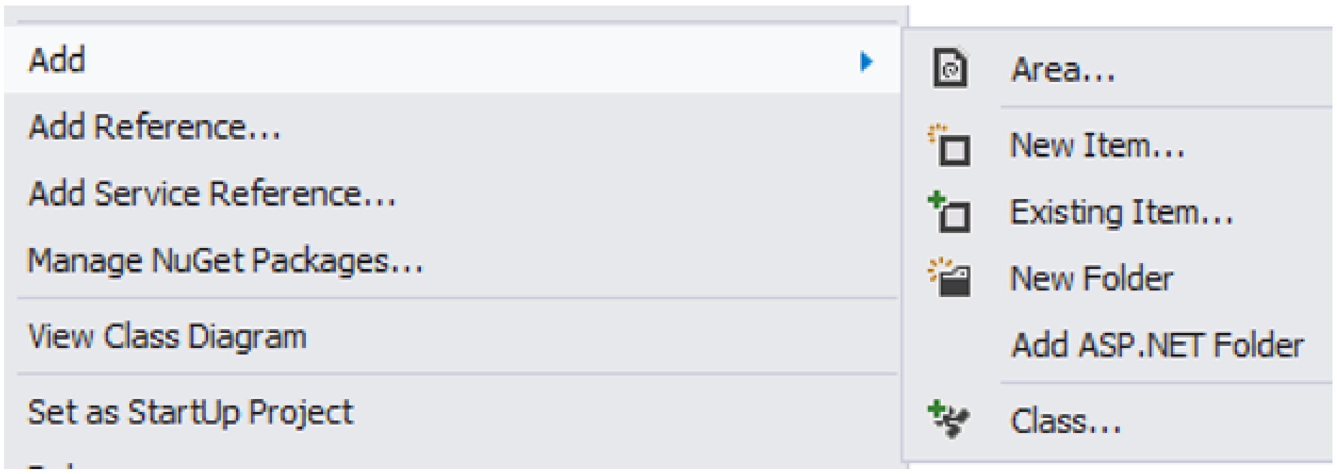
Areas help you to group functionalities in to independent modules thus making your project more organized. For example in the below MVC project we have four controller classes and as time passes by if more controller classes are added it will be difficult to manage. In bigger projects you will end up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



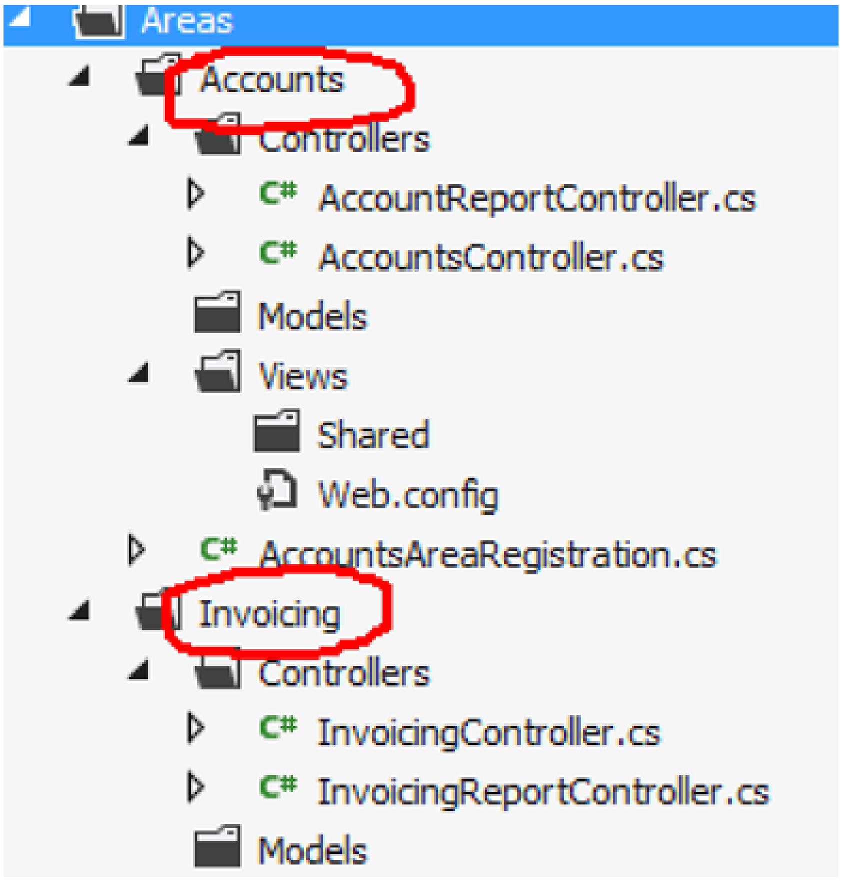
If we can group controller classes in to logical section like "Invoicing" and "Accounting" that would make life easier and that's what "Area" are meant to.



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.



Explain the concept of View Model in MVC?

A view model is a simple class which represents data to be displayed on the view.

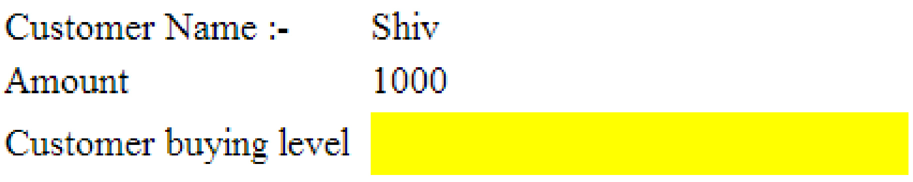
For example below is a simple customermodel object with “CustomerName” and “Amount” property.

```
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 indicationwhich 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

What kind of logic view model class will have?

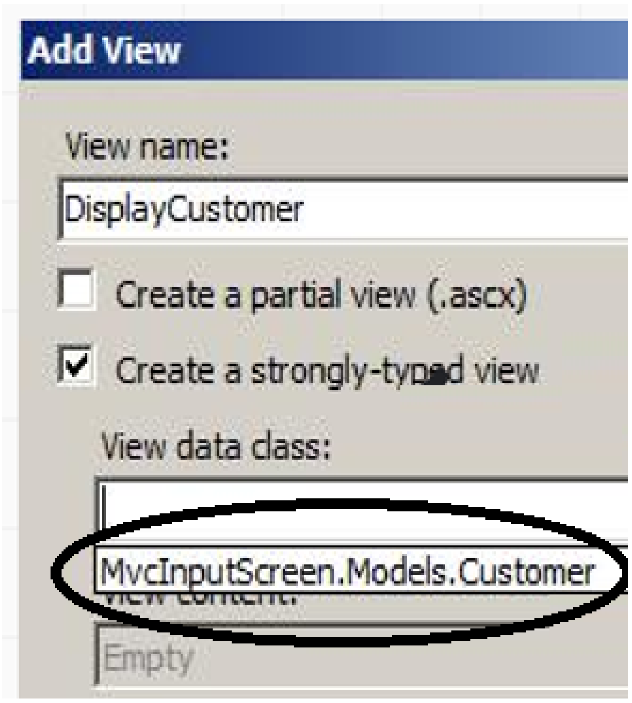
As the name says view model this class has the gel code or connection code which connects the view and the model.

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 differentCustomer 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.

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();
}
```

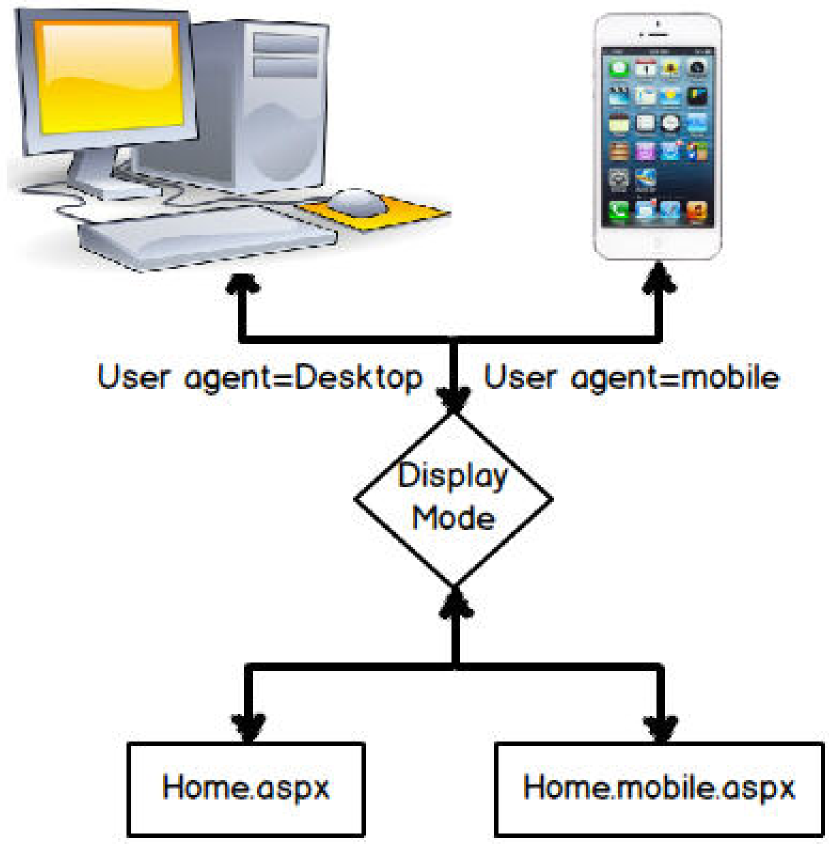
In the view we can refer both the model using the view model as shown in the below code.

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

Explain the need of display mode in MVC?

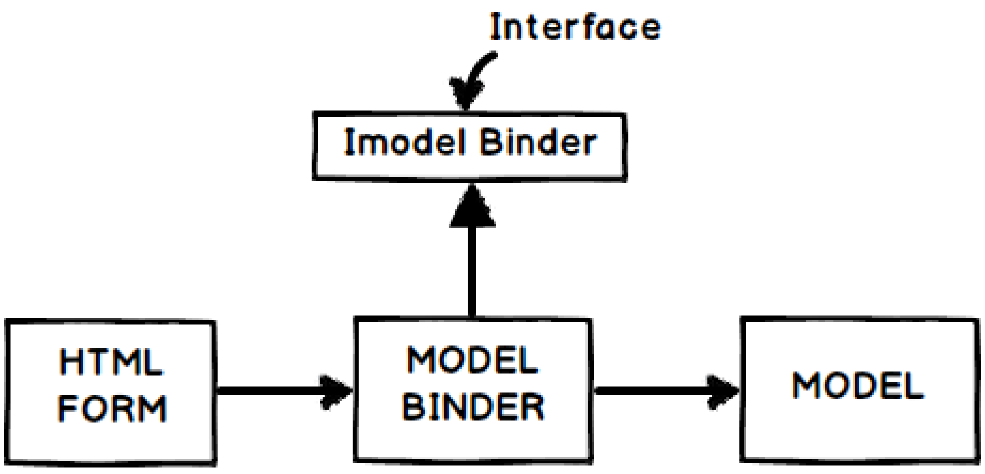
Display mode displays views depending on the device the user has logged in with. So we can create different views for different devices anddisplay mode will handle the rest.

For example we can create a view “Home.aspx” which will render for the desktop computers and Home.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.



Explain MVC model binders?

Model binder maps HTML form elements to the model. It acts like a bridge between HTML UI and MVC model. Many times HTML UI names are different than the model property names. So in the binder we can write the mapping logic between the UI and the model.

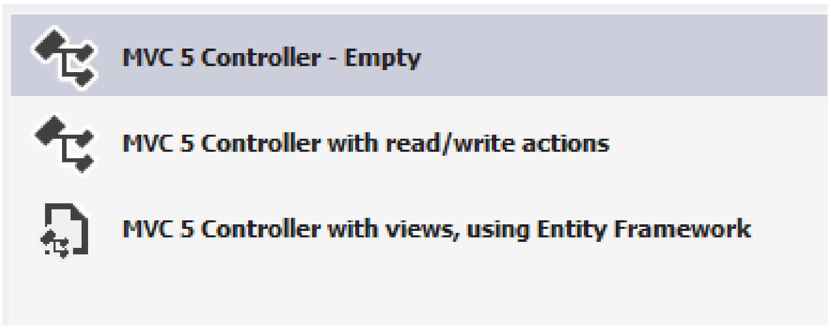


Explain the concept of MVC Scaffolding?

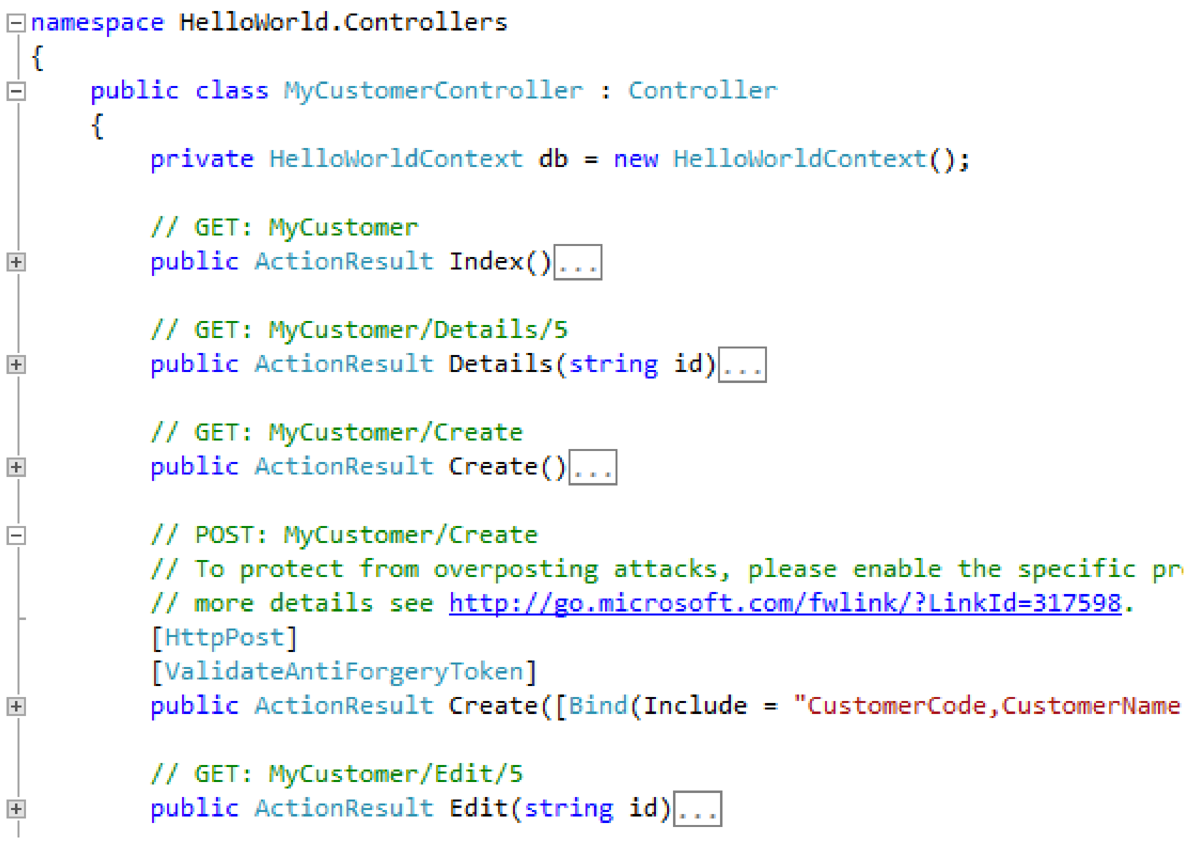
```
Note :- Do not get scared with the word. Its actually a very simple thing.
```

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).

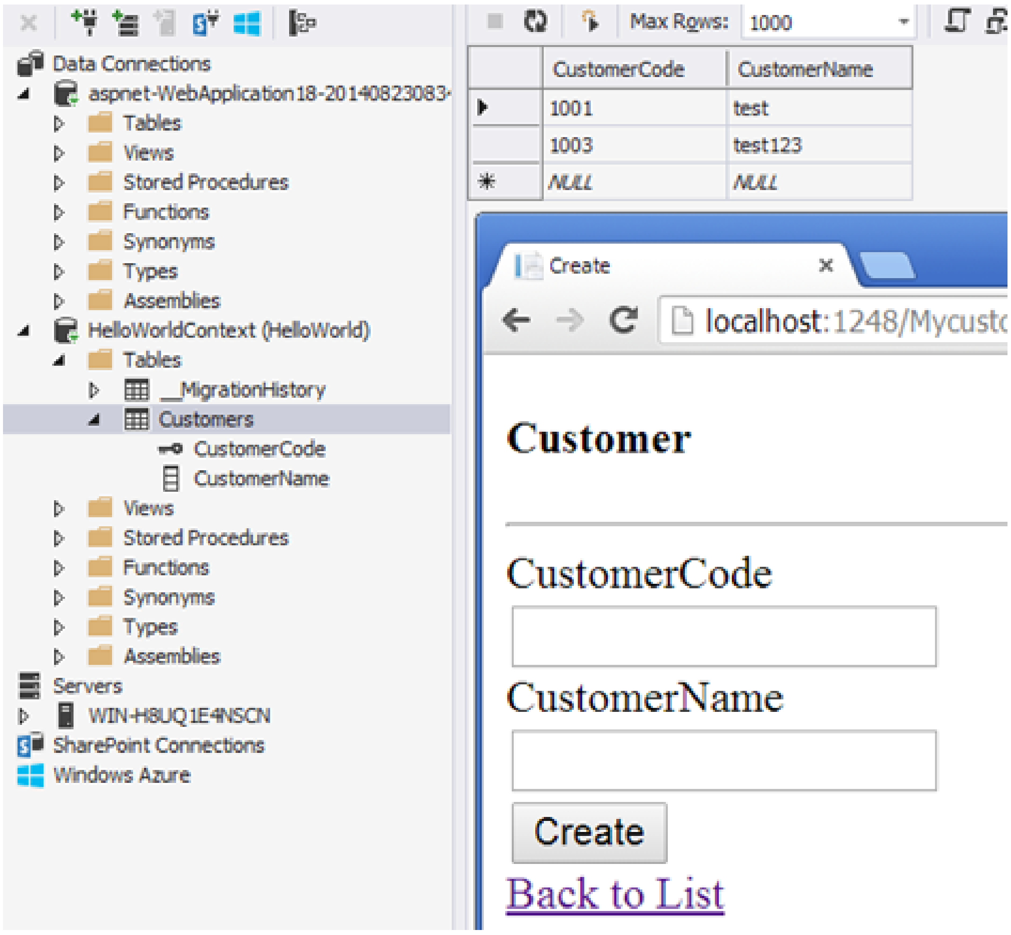

MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework

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

What does scaffolding use internally to connect to database?

It uses Entity framework internally.

How can we do exception handling in MVC?

In the controller you can override the “OnException” event and set the “Result” to the view name which you want to invoke when error occurs. In the below code you can see we have set the “Result” to a view named as “Error”.

We have also set the exception so that it can be displayed inside the view.

```
public class HomeController : Controller
 {
        protected override void OnException(ExceptionContext filterContext)
        {
            Exception ex = filterContext.Exception;
            filterContext.ExceptionHandled = true;

     var model = new HandleErrorInfo(filterContext.Exception, "Controller","Action");

     filterContext.Result = new ViewResult()
{
                ViewName = "Error",
                ViewData = new ViewDataDictionary(model)
     };

        }
}
```

To display the above error in view we can use the below code

```
@Model.Exception;
```

How can you handle multiple Submit buttons pointing to multiple actions in a single MVC view?

Let us elaborate on what the interviewer wants to ask because the above question is just a single liner and is not clear about what the interviewer wants.

Take a scenario where you have a view with two submit buttons as shown in the below code.

```
<form action="Action1" method=post>
<input type=&rdquo;submit&rdquo; name=&rdquo;Submit1&rdquo;/>
<input type=&rdquo;submit&rdquo; name=&rdquo;Submit2&rdquo;>
</form>
```

In the above code when the end user clicks on any of the submit buttons it will make a HTTP POST to “Action1”.

The question from the interviewer is:-

“What if we have want that on “Submit1” button click it should invoke “Action1” and on the “Submit2” button click it should invoke “Action2”.”

Now that we have understood the question let us answer the question in a detailed manner. There are two approaches to solve the above problem one is the normal HTML way and the other is the “Ajax” way.

In the HTML way we need to create two forms and place the “Submit” button inside each of the forms. And every form’s action will point to different / respective actions. You can see the below code the first form is posting to “Action1” and the second form will post to “Action2” depending on which “Submit” button is clicked.

```
<form action="Action1" method=post>
<input type=&rdquo;submit&rdquo; name=&rdquo;Submit1&rdquo;/>
</form>
```

In case the interviewer complains that the above approach is not AJAX this is where the second approach comes in. In the Ajax way we can create two different functions “Fun1” and “Fun1” , see the below code. These function will make Ajax calls by using JQUERY or any other framework. Each of these functions are binded with the “Submit” button’s “OnClick” events.

```
<Script language="javascript">
function Fun1()
{
$.post(&ldquo;/Action1&rdquo;,null,CallBack1);
}
function Fun2()
{
$.post(&ldquo;/Action2&rdquo;,null,CallBack2);
}
</Script>
```

What is CSRF attack and how can we prevent the same in MVC?

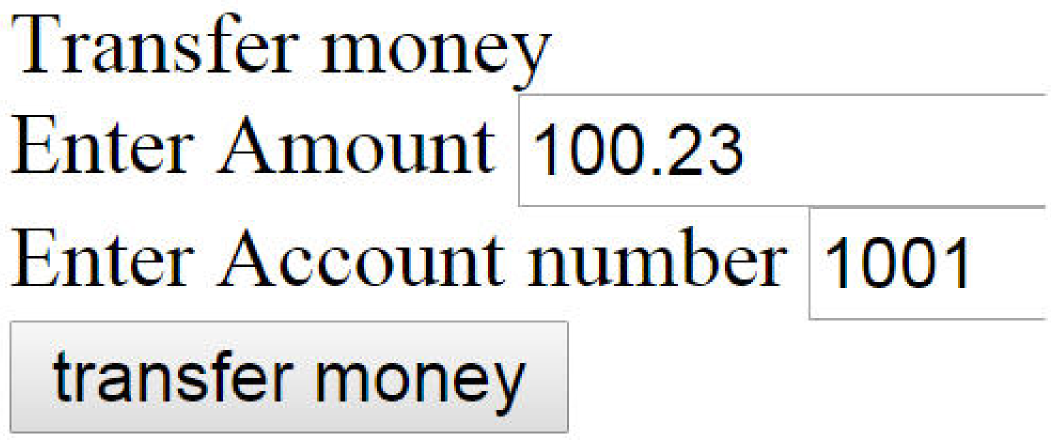
CSRF stands for Cross site request forgery. So if you see the dictonary meaning of forgery: -

"It's an act of copying or imitating things like signature on a cheque, official documents to deceive the authority source for financial gains."

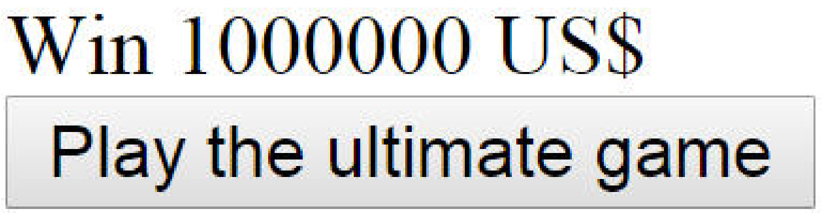
So when it comes to website this forgery is termed as CSRF (Cross Site Request Forgery).

CSRF is a method of attacking a website where the attacker imitates a.k.a forges as a trusted source and sends data to the site. Genuine site processes the information innocently thinking that data is coming from a trusted source.

For example conside the below screen of a online bank. End user's uses this screen to transfer money.



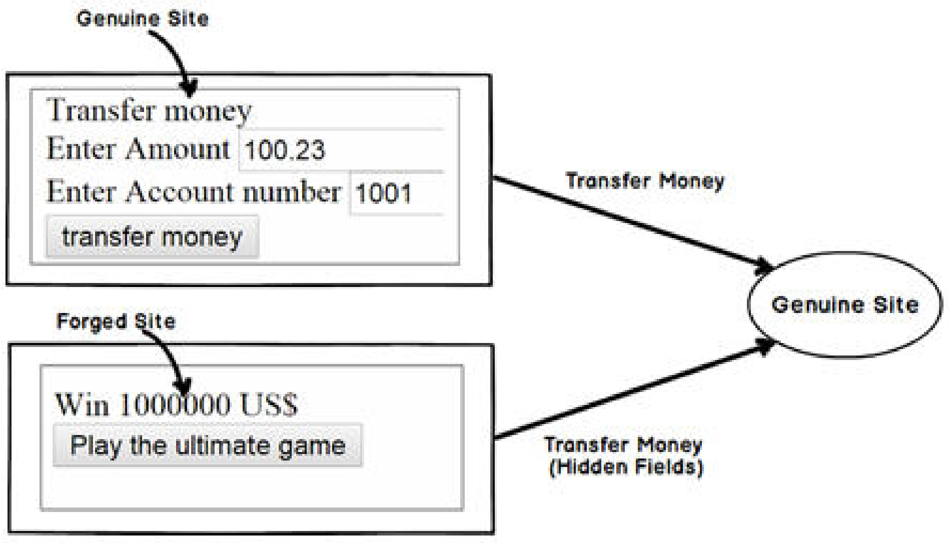
Below is a forged site created by an attacker which looks a game site from outside, but internally it hits the bank site for money transfer.



The internal HTML of the forged site has those hidden fields which have the account number and amount to do money transfer.

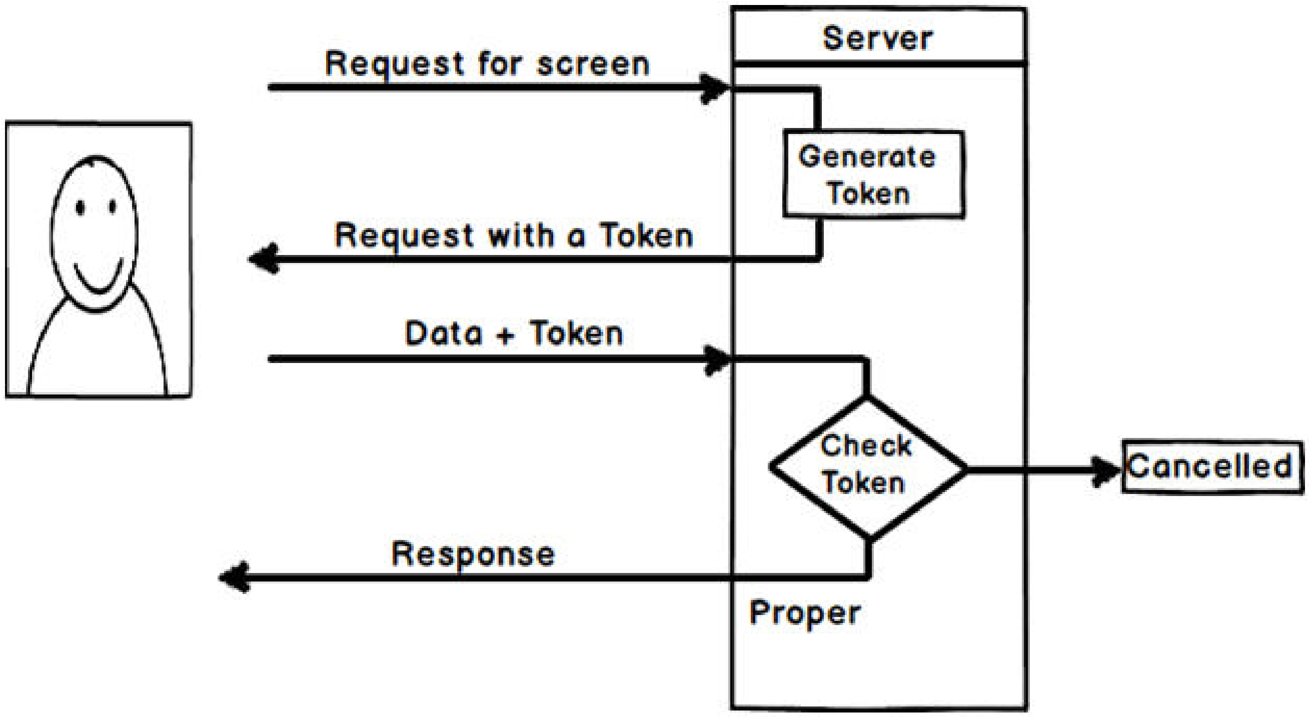
```
<div>
        Win 1000000 US$
<form action="http://localhost:23936/Genuine/Transfer" method=post>
<input type=hidden name="amount" value="10000" />
<input type=hidden name="account" value="3002" />
<input type=submit value="Play the ultimate game" />
</form>
</div>
```

Now let's say the user has logged in to the genuine bank site and the attacker sent this forged game link to his email. The end user thinking that it's a game site clicks on the "Play the Ultimate Game" button and internally the malicious code does the money transfer process.



So a proper solution to this issue can be solved by using tokens: -

- End user browses to the screen of the money transfer. Before the screen is served server injects a secret token inside the HTML screen in form a hidden field.
- Now hence forth when the end user sends request back he has to always send the secret token. This token is validated on the server.



Implementing token is a two-step process in MVC: -

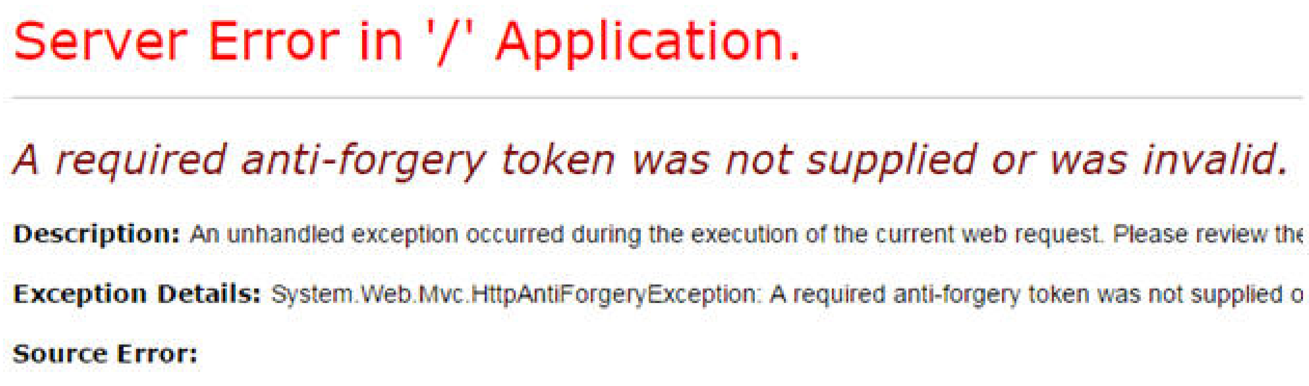
First apply "ValidateAntiForgeryToken" attribute on the action.

```
[ValidateAntiForgeryToken]
public ActionResult Transfer()
{
            // password sending logic will be here
            return Content(Request.Form["amount"] +
                " has been transferred to account "
                + Request.Form["account"]);
}
```

Second in the HTML UI screen call "@Html.AntiForgeryToken()" to generate the token.

```
<div>
        Transfer money
<form action="Transfer" method=post>
        Enter Amount
<input type="text" name="amount" value="" />

        Enter Account number


         @Html.AntiForgeryToken()
<input type=submit value="transfer money" />
</form>
</div>
```

So now henceforth when any untrusted source send a request to the server it would give the below forgery error.



Server Error in '/' Application.

A required anti-forgery token was not supplied or was invalid.

Description: An unhandled exception occurred during the execution of the current web request. Please review the

Exception Details: System.Web.Mvc.HttpAntiForgeryException: A required anti-forgery token was not supplied o

Source Error:

If you do a view source of the HTML you would find the below verification token hidden field with the secret key.

```
<input name="__RequestVerificationToken" type="hidden"
value="7iUdhsDNpEwiZFTYrH5kp/q7jL0sZz+CSBh8mb2ebwvxMJ3eYmUZXp+uofko6eiPD0fmC7Q0o4SXeGgRpxFp
0i+Hx3fgVlVybgCYpyhFw5IRyYhNqi9KyH0se0hBPRu/9kYwEXXnVGB9ggdXCVPcIud/gUzjWVCvU1QxGA9dKPA="
/>
```

Please do read this blog which has detailed steps of how model binders can be created using "IModelBinder" interface: - Explain MVC model Binders?

Download an e-learning copy of MVC interview Q&A from the top of this article for your preparation.

For technical training related to various topics including ASP.NET, Design Patterns, WCF, MVC, BI, WPF contact SukeshMarla@gmail.com or visit www.sukesh-marla.com

Finally do not forget to visit my video site which covers lots of C# interview questions and answers: www.questpond.com.

License

Share

About the Authors