## Introduction

Let us about Model View Architecture and its advantages than the Web forms .Asp.Net MVC is a frame work for developing asp.net Web applications that Model View Controller Design Pattern .  
  
The Asp.Net MVC contains 3 Core Parts

  1)Model

  2)View

  3)Controller



## Model :

Model is the part of the application that implements the data domain logic .It is responsible for data base connections and implementing business rules Model responds to the controller request .

## View :

View is the graphical data presentation i.e (User Interface ).When ever a controller request's the model , it will fetch the data and display in the appropriate View .

## Controller :

The controller is the heart of the MVC .When ever a request came from the clients ,the controller accepts the request and it will invoke the model and model fetches the data from DB and send's the result to appropriate View .

In the controller the routing engine will play a major role.

MS provides two types of routing engine's

1)ASPX

2)Razor.

I prefer razor because it allows only pure html controls ,which is the main objective of MVC .

They are some third party routing engine's also available like nHaml ,Spark etc.

We can also create a custom routing engine.

I will explain in depth about the routing engine's in the coming articles .

## Advantages of MVC over Webforms :

1)We have clear separation of layers that makes application maintenance very simple

2)No Asp Web server control's so,No event handlers page size will become very light

3)View state is not supported , so performance is increased

4)View Engine's supported

5)Search engine optimization is very high

6)User Interface is lightly coupled

7)Parallel Development is Possible  and easy to do unit testing

## Dis-advantage :

1)Since there was no support for the View state ,Developer has to write more code to maintain data in the page when an exception occurred while submitting the data .

Add main view

Now we will add one main view, from where we will call partial view. Here is the content of main view.

@{  
    Layout = null;  
}  
   
<!DOCTYPE html>  
<html>  
<head>  
    <title>Index</title>  
</head>  
<body>  
    <div>  
        This is normal Page.  
        @{Html.RenderPartial("\_Partial");}  
    </div>  
</body>  
</html>

Html helper method to call partial view

To render partial view from any view we have to use RenderPartial() function of HTML helper class.  There are four overloaded function of RenderPartial() and we have chosen the first one which takes view name as argument. We have supplied \_Partical argument which is nothing but our partial view.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Various return types of action in MVC application

## Various return types of action in MVC application

In this article we will discuss various return types of Action. Generally in MVC application the default return type of action is ActionResult but we can return those type of value whose class is derive from ActionResult class.  Here we will see various return types one by one with example.

##### Return default View

If we do not specify any view name when we want to return view then by default, default view will be returned.  In below example, from Index action we are returning view but we did not specify view name, then by default it will call the view which is attached with this action.

public class TestController : Controller  
    {  
        public ActionResult Index()  
        {  
            return View();  
        }  
   
    }

##### http://www.dotnetfunda.com/UserFiles/ArticlesFiles/Sourav.Kayal_3314_return_view.jpg

##### Return Specific view from Action.

If we don’t want to call default view then we can specify view name. For example in below example we have specify view name. We have to keep in mind that the view should be in the same controller folder. Otherwise we have to keep view in shared folder.

public class TestController : Controller  
    {  
        public ActionResult Index()  
        {             
  return View("Index");  
        }  
    }

##### http://www.dotnetfunda.com/UserFiles/ArticlesFiles/Sourav.Kayal_3314_return_view.jpg

##### Return content

Sometimes it needs to return raw string data rather than calling any view. In this situation we can return content. In below example we are returning Content and it will return simple string in browser.

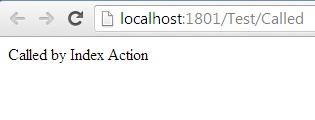
public class TestController : Controller  
    {  
        public ActionResult Index()  
        {            
      return Content("The content is return from Action");  
        }  
}

##### http://www.dotnetfunda.com/UserFiles/ArticlesFiles/Sourav.Kayal_5463_return_content.jpg

##### Return to Different Action

This is common situation where it needs to call other action from some other action. In below example we have created two actions in Test controller. Where from Index action we will call Called action. To do same ,we have to use RedirectToAction() function.

public class TestController : Controller  
    {  
        public ActionResult Index()  
        {  
            return RedirectToAction("Called");  
        }  
        public ActionResult Called()  
        {  
            return Content("Called by Index Action");  
        }  
    }

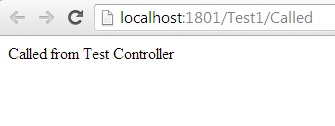


Now, please note that both actions are in same controller. When both actions are in same controller then action name is enough to call specific action.

##### Redirect to Different Action within same or different controller

When we want to call any action which belongs to different controller, we have to specify path of that controller. In below example we want to call Called action which belongs to Test1 controller from Index action which belongs to Test controller.

public class TestController : Controller  
    {  
        public ActionResult Index()  
        {  
            return RedirectToAction("../Test1/Called");  
        }  
          
    }  
    public class Test1Controller : Controller  
    {  
        public ActionResult Called()  
        {  
            return Content("Called from Test Controller");  
        }  
    }  
 



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 3 ways to pass data from controller to view

 ways to pass data from controller to view

In this article we will discuss three different ways to pass data from controller to view in MVC web application. Those three approaches are , using ViewBag ,ViewData and TempData

ViewBag:-

ViewBag is a dynamic approach to pass data.

It’s available only in MVC3 .

It does not require type casting for complex data type. Means there is no type cast needed even if we pass object of a class.

It does not retain value in redirection.

ViewData:-

It store data in Key value pair(Like session)

Type casting is needed for complex data type

ViewData cannot retain it’s value in redirection.

TempData:-

Type cast is needed for complex data type.

Store data using Kay(Like session)

Can able to retain value in redirection.

Create Person class in Model

using System;  
using System.ComponentModel.DataAnnotations;  
   
namespace PersonModel  
{  
    public class Person  
    {  
        public String Name { get; set; }  
        public String Surname { get; set;}  
   
    }  
}

This is our small person class containing two properties called Name and Surname

Create controller class

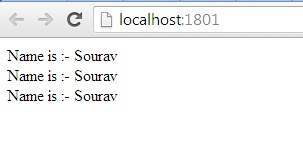
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Mvc;  
   
namespace MVCTest.Controllers  
{  
    public class PersonController : Controller  
    {  
         
        public ActionResult ShowUser()  
        {  
 PersonModel.Person P = new PersonModel.Person();  
 P.Name = "Sourav";  
 P.Surname = "Kayal";  
  ViewBag.Person = P;  
 ViewData["P"] = P;  
 TempData["P"] = P;  
            return View("Show");  
        }  
   
    }  
}

In this Person controller we have implemented ShowUser() action which will assign object of Person class in ViewBag, ViewData and TempData and it will call to Show view.

Create View  
@model PersonModel.Person  
   
@{  
    Layout = null;  
}  
   
<!DOCTYPE html>  
   
<html>  
<head>  
    <title>Show</title>  
</head>  
<body>  
    <div>  
    @{  
        var ViewBagValue = ViewBag.Person;  
        var ViewDataValue =(PersonModel.Person) ViewData["P"];  
        var TempDataValue =(PersonModel.Person) TempData["P"];  
    }  
     Name is :-  @ViewBagValue.Name<br/>  
     Name is:-  @ViewDataValue.Name <br />  
     Name is:-  @TempDataValue.Name <br />  
          
    </div>  
</body>  
</html>

Here we are extracting value from three different variables. We can see there is no casting needed for ViewBag where casting needed for ViewData and TempData

Here is sample output:-



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*4\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

How to pass value from URL in MVC application.

## Pass parameter from URL in ASP.NET MVC application

In this article we will be discussing how to pass parameter in controller from URL in MVC application. First of all create one empty MVC web project and implement below code.

## Create Model Class

At first we will create one model class called Person with two properties. Then within this class we will write simple ADO.NET code to fetch data from SQLServer Database.

using System;  
using System.Data;  
using System.Data.SqlClient;  
using System.Collections.Generic;  
   
namespace PersonModel  
{  
    public class Person  
    {  
        public String Name { get; set; }  
        public String Surname { get; set;

}  
   
   
        SqlConnection con = new SqlConnection("Data Source=SERVERNAME\\SQL\_INSTANCE;Initial Catalog=test;Integrated

Security=True");  
   
        public List<Person> GetPerson(int Id)  
        {  
            List<Person> objP = new List<Person>();  
            //Person P = null;  
            if (con.State == ConnectionState.Broken || con.State == ConnectionState.Closed)  
            {

con.Open();  
            }  
            SqlCommand cmd = new SqlCommand("select \* from person where id=" + Id, con);  
            SqlDataReader rd= cmd.ExecuteReader();  
            while (rd.Read())  
            {

Person p = new

Person();

p.Name = rd.GetString(1);

p.Surname = rd.GetString(2);

objP.Add(p);  
            }  
            return objP;  
        }  
   
    }  
   
}

Within GetPerson() function we are fetching data from database and pushing it onto List in a form of object. Then we are returning entire List to controller class.

## Create controller class

Here we will create controller class. The class name is PersonController  and it is inherited from Controller class.

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Mvc;  
using PersonModel;  
namespace MVC3.Controllers  
{  
    public class PersonController : Controller  
    {  
        //  
        // GET: /Person/  
   
        public ActionResult Index(int Id)  
        {  
            Person obj = new Person();  
            List<Person> Li = obj.GetPerson(Id);  
            return View(Li.ToList());  
        }  
   
    }  
}

Withing Index() action we are consuming output of Model class and returning to view. One more thing is need to observer, Index() action is getting Id as parameter and this value will get pass from URL.

## Create View

It’s the time to create one simple view where we will show output of controller class. Have a look on below code.

@model IList<PersonModel.Person>  
   
@{  
    Layout = null;  
}  
   
<!DOCTYPE html>  
   
<html>  
<head>  
    <title>Index</title>  
</head>  
<body>  
    <div>  
    @{  
        foreach(var O in Model)  
        {  
         <li>Name:- </li>      @O.Name;  
         <li>Surname:- </li>   @O.Surname;  
        }  
    }  
          
    </div>  
</body>  
</html>

Set default routing URL

This is the last part of configuration. We will set default URL for the project Change routing section on Global.aspx page.

public static void RegisterRoutes(RouteCollection routes)  
        {

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");  
 

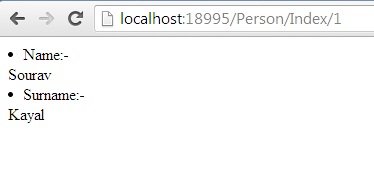
routes.MapRoute(  
 "Default", // Route name  
 "{controller}/{action}/{id}",

// URL with parameters  
            new

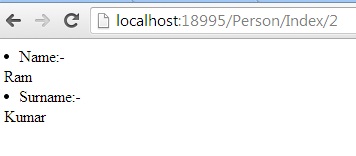
{ controller = "Person", action = "Index", id = UrlParameter.Optional

} // Parameter defaults  
 );  
        }

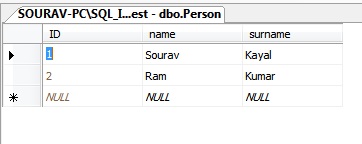
 Here is output when we are passing 1 as parameter.



When we are passing 2 as parameter



This is our table structure



## Conclusion

Here we have seen how to fetch data from database according to value passed from URL. Hope you have enjoyed it.

# Send List from Controller to View in MVC application

How to send list from controller to view.

Send List from Controller to View in MVC application

In this article we will learn how to pass list from controller to View. At first we will create model class and pass list of model class object from controller to view.

## Create model class.

Our model class is very simple. It contains only two properties called Name and Surname. Both are string type.

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
   
namespace MVC3.Models  
{  
    public class Customer  
    {  
        public String Name { get; set; }  
        public String Surname { get; set;}  
    }  
}

## Create controller class

Now, We will create controller class to create list of object. Our Customer controller contains Index() action which will call view and pass list of object.

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Mvc;  
using MVC3.Models;  
   
namespace MVC3.Controllers  
{  
    public class CustomerController : Controller  
    {  
        public ActionResult Index()  
        {  
            List<Customer> ObjCustomer = new List<Customer>();  
            Customer Obj = new Customer();  
 Obj.Name = "Sourav ";  
 Obj.Surname = "Kayal";  
 ObjCustomer.Add(Obj);

Obj = new Customer();  
 Obj.Name = "Ram";  
 Obj.Surname = "Kumar";  
 ObjCustomer.Add(Obj);  
   
            return View(ObjCustomer.ToList());  
        }  
   
    }  
}

## Razor view

Finally we will create view to display list in client part. In below we have implemented view with the help of Razor view engine.

@model IList<MVC3.Models.Customer>  
@{  
   // Layout = null;  
}  
<!DOCTYPE html>  
<html>  
<head>  
    <title>Index</title>  
</head>  
<body>  
    <div>  
      @foreach

(var obj in

Model) {  
          <li>Name:-    @obj.Name</li>  
          <li>Surname:- @obj.Surname</li> <br />  
      }  
   
    </div>  
</body>  
</html>

## ASPX View

If you are interested to use ASPX view then use below code.

<%@ Page

Language="C#"

Inherits="System.Web.Mvc.ViewPage<dynamic>"

%>  
<!DOCTYPE html>  
<html>  
<head runat="server">  
    <title>Index</title>  
</head>  
<body>  
    <div>  
        <% var V = Model; %>  
        <% foreach (var a in V)  
        { %>  
           <b>Name:-   </b>   <%= a.Name %>  
           <b>Surname:-</b>  <%= a.Surname%> <br />  
   
        <%} %>  
   
   
    </div>  
</body>  
</html>

At last we will set Default controller and action for this application. Here we have set Customer as controller and Index as action.

 routes.MapRoute(

"Default", // Route name

"{controller}/{action}/{id}",

// URL with parameters  
   new {

controller = "Customer", action = "Index", id = UrlParameter.Optional

} // Parameter defaults

);

Here is sample output



## Conclusion

Here we have learned how to send List from Controller to view in MVC application

# Create strong type view in MVC application

In this quick article we will learn how to create strong type view in MVC application. There is little advantage in strong type view. We can send data in form of object. Let’s implement strong type view with small example.

##### Create simple model

Open one empty MVC application and add below model into it.

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
   
namespace MVCTest.Models  
{  
    public class Person  
    {  
        public String Name{ get; set; }  
        public String Surname { get; set;

}  
        public String Age

{ get; set; }  
    }  
}

Our model is very simple, only we have implemented Person classes which contain three properties.

##### Create Person Controller

Let’s create one controller to handle model data. As we have defined our model class name as Person. We will define controller name as PersonController. Have a look on below code.

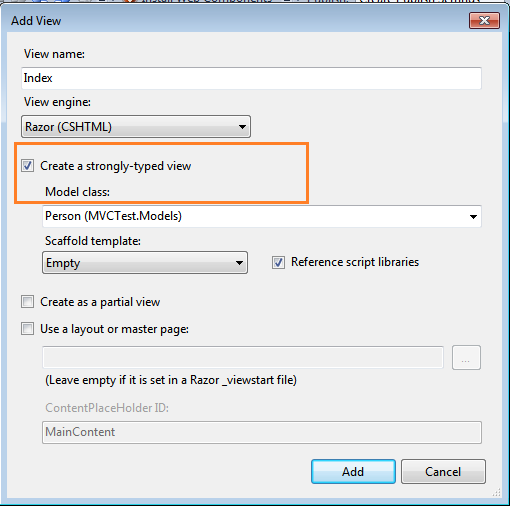
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Mvc;  
using MVCTest.Models;  
   
namespace MVCTest.Controllers  
{  
    public class PersonController : Controller  
    {  
        
        public ActionResult Index()  
        {  
            return View();  
        }  
        public ActionResult GetData(Person objP)  
        {  
            String Name = objP.Name;  
            String Surname = objP.Surname;  
            Int32 Age = Convert.ToInt32(objP.Age);  
            return Content("Name:-" + Name + " Surname:-" + objP.Surname + " Age:-" +

objP.Age);  
        }  
   
    }  
}

The controller contains tow actions. Index action will show HTML form, and GetData() function will call when we sill submit form.

##### Create strong type view

Now we will create strong type view. The advantage of strong type view is we can send and receive data in the form of object. To create strong type view, just right click on Index action, you will see below form.



Now,

 add below code in view

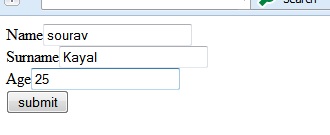
@model MVCTest.Models.Person  
   
@{  
    Layout = null;  
}  
   
<!DOCTYPE html>  
   
<html>  
<head>  
    <title>Index</title>  
</head>  
<body>  
    <div>  
        @using

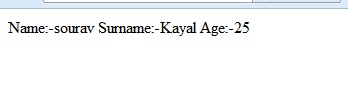
(Html.BeginForm("Person", "GetData", FormMethod.Post))  
        {  
         @Html.LabelFor(m=>m.Name)  
         @Html.TextBoxFor(m=> m.Name)  
   
         @Html.LabelFor(m=>m.Surname)  
         @Html.TextBoxFor(m => m.Surname);

         @Html.LabelFor(m => m.Age)  
         @Html.TextBoxFor(m => m.Age);  
   
         <input type="submit" name="Submit" value="submit" />  
        }  
   
    </div>  
</body>  
</html>

Have a look on first line of View. We are pointing Person class as model. And as it is strong type view, we can use lamda expression to point each property of model class.

##### Output:-





# Looping and branching in Razor view

In this article we will learn looping and branching concept in Razor view engine in MVC application.

This article assumes that the reader has basic understanding of MVC design pattern. Let’s start with simple example each one of them.

##### For Loop

Let’s have a look how to implement for loop in Razor view engine. It’s very similar with normal for loop in C#. Only we have to specify @ symbol in front of loop.

@{  
    Layout = null;  
}  
<!DOCTYPE html>  
<html>  
<head>  
    <title>Index</title>  
</head>  
<body>  
    <div>  
           @for(int i = 0; i < 5; i++)  
           {  
 <p>Value of i is= @i  </p>  
           }  
             
    </div>  
</body>  
</html>

##### http://www.dotnetfunda.com/UserFiles/ArticlesFiles/Sourav.Kayal_1417_for_Loop.jpg

##### Foreach loop

Have a look on below code to learn implementation of for each loop in Razor view.

@{  
    Layout = null;  
}  
<!DOCTYPE html>  
<html>  
<head>  
    <title>Index</title>  
</head>  
<body>  
    <div>  
        <p>For each Loop</p>  
          @{int []Val = {1,2,3,4,5};}  
          @foreach (var i in Val)  
          {  
 <p>Value of i is = @i </p>  
          }  
    </div>  
</body>  
</html>

##### http://www.dotnetfunda.com/UserFiles/ArticlesFiles/Sourav.Kayal_9269_foreach_loop.jpg

##### While Loop

In below view, we will see how to implement while loop in Razor view.  At first we have declare A(Loop control variable) and within loop we are incrementing value of A.

@{  
    Layout = null;  
}  
<!DOCTYPE html>  
<html>  
<head>  
    <title>Index</title>  
</head>  
<body>  
    <div>  
        <p>Using While Loop</p>  
           @{int A =1;}  
           @while(A<=5)  
           {  
 <p>Value of A is= @A </p>  
           }  
             
    </div>  
</body>  
</html>

##### http://www.dotnetfunda.com/UserFiles/ArticlesFiles/Sourav.Kayal_6203_While_Loop.jpg

##### Do while loop

Again, do while loop is very similar with do while in C#. Look at the code below

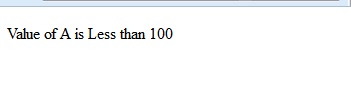
@{  
    Layout = null;  
}  
<!DOCTYPE html>  
<html>  
<head>  
    <title>Index</title>  
</head>  
<body>  
    <div>  
        <p>Using Do-While Loop</p>  
           @{int A =1;}  
           @do  
           {  
 <p>Value of A is= @A  </p>  
 A++;  
           }while(A<=5);  
    </div>  
</body>  
</html>

##### http://www.dotnetfunda.com/UserFiles/ArticlesFiles/Sourav.Kayal_4621_do_while_book.jpg

##### If Else condition

Let’s see how to implement If Else condition in Razor syntax. Have a look on below code.

@{  
    Layout = null;  
}  
<!DOCTYPE html>  
<html>  
<head>  
    <title>Index</title>  
</head>  
<body>  
    <div>  
            @{int A =10}  
            @if(A<100)  
 {    
 <p>Value of A is Less than 100</p>  
            }  
            else  
            {  
 <p>Value of A is greater than 100</p>  
            }  
    </div>  
</body>  
</html>



Interview Questions

### MVC Interview Questions/Answers

|  |
| --- |
| **What are the 3 main components of an ASP.NET MVC application?** 1. M - Model 2. V - View 3. C - Controller  **In which assembly is the MVC framework defined?** System.Web.Mvc   * System.Web.Helpers.dll * System.Web.Mvc.dll * System.Web.Razor.dll * System.Web.WebPages.Deployment.dll * System.Web.WebPages.dll * System.Web.WebPages.Razor.dll   **Is it possible to combine ASP.NET webforms and ASP.MVC and develop a single web**  **application?** Yes, it is possible to combine ASP.NET webforms and ASP.MVC and develop a single  web application.  **What does Model, View and Controller represent in an MVC application?** Model: Model represents the application data domain. In short the applications business logic is  contained with in the model.  View: Views represent the user interface, with which the end users interact. In short the all the  user interface logic is contained with in the UI.  Controller: Controller is the component that responds to user actions.  Based on the user actions, the respective controller, work with the model,  and selects a view to render that displays the user interface.  The user input logic is contained with in the controller.  **What is the greatest advantage of using asp.net mvc over asp.net webforms?** It is difficult to unit test UI with webforms, where views in mvc can be very easily unit tested.  **Which approach provides better support for test driven development - ASP.NET MVC or ASP.NET Webforms?** ASP.NET MVC  **What are the advantages of ASP.NET MVC?** 1. Extensive support for TDD. With asp.net MVC, views can also be very easily unit tested. 2. Complex applications can be easily managed 3. Seperation of concerns. Different aspects of the application can be divided into Model,  View and Controller. 4. ASP.NET MVC views are light weight, as they donot use viewstate.  **Is it possible to unit test an MVC application without running the controllers in an ASP.NET process?** Yes, all the features in an asp.net MVC application are interface based and hence mocking is much easier.  So, we don't have to run the controllers in an ASP.NET process for unit testing.  **Is it possible to share a view across multiple controllers?** Yes, put the view into the shared folder.  This will automatically make the view available across multiple controllers.  **What is the role of a controller in an MVC application?** The controller responds to user interactions, with the application,  by selecting the action method to execute and alse selecting the view to render.  **Where are the routing rules defined in an asp.net MVC application?** In Application\_Start event in Global.asax  **Name a few different return types of a controller action method?** The following are just a few return types of a controller action method. In general an action method can return an instance of a any class that derives from ActionResult class. 1. ViewResult 2. JavaScriptResult 3. RedirectResult 4. ContentResult 5. JsonResult  **What is the significance of NonActionAttribute?** In general, all public methods of a controller class are treated as action methods.  If you want prevent this default behaviour, just decorate the public method with  NonActionAttribute.  **What is the significance of ASP.NET routing?** ASP.NET MVC uses ASP.NET routing, to map incoming browser requests to controller action  methods. ASP.NET Routing makes use of route table. Route table is created when your web application first starts. The route table is present in the  Global.asax file.  **What are the 3 segments of the default route, that is present in an ASP.NET MVC application?** 1st Segment - Controller Name 2nd Segment - Action Method Name 3rd Segment - Parameter that is passed to the action method  **Example:** [http://pragimtech.com/Customer/Details/5](http://www.dotnetobject.com/expage.php?exurl=http://pragimtech.com/Customer/Details/5) Controller Name = Customer Action Method Name = Details Parameter Id = 5  **ASP.NET MVC application, makes use of settings at 2 places for routing to work|  correctly. What are these 2 places?** 1. Web.Config File : ASP.NET routing has to be enabled here. 2. Global.asax File : The Route table is created in the application Start event handler,   of the Global.asax file.  **What is the adavantage of using ASP.NET routing?** In an ASP.NET web application that does not make use of routing, an incoming browser request  should map to a physical file. If the file does not exist, we get page not found error.  An ASP.NET web application that does make use of routing, makes use of URLs that do not have  to map to specific files in a Web site. Because the URL does not have to map to a file,  you can use URLs that are descriptive of the user's action and therefore are more easily  understood by users.  **What are the 3 things that are needed to specify a route?** **1. URL Pattern** - You can include placeholders in a URL pattern so that variable data can be  passed to the request handler without requiring a query string. **2. Handler** - The handler can be a physical file such as an .aspx file or a controller class. **3. Name for the Route** - Name is optional.  **Is the following route definition a valid route definition?** {controller}{action}/{id} No, the above definition is not a valid route definition, because there is no literal value or  delimiter between the placeholders. Therefore, routing cannot determine where to separate the  value for the controller placeholder from the value for the action placeholder.  **What is the use of the following default route?** {resource}.axd/{\*pathInfo} This route definition, prevent requests for the Web resource files such as WebResource.axd or  ScriptResource.axd from being passed to a controller.  **What is the difference between adding routes, to a webforms application and to an  mvc application?** To add routes to a webforms application,  we use MapPageRoute() method of the RouteCollection  class, where as to add routes to an MVC application we use MapRoute() method.  **How do you handle variable number of segments in a route definition?** Use a route with a catch-all parameter. An example is shown below. \* is referred to as catch-all parameter. controller/{action}/{\*parametervalues}  **What are the 2 ways of adding constraints to a route?** **1.** Use regular expressions **2.** Use an object that implements IRouteConstraint interface  **Give 2 examples for scenarios when routing is not applied?** **1. A Physical File is Found that Matches the URL Pattern** - This default behaviour can be  overriden by setting the RouteExistingFiles property of the RouteCollection object to true. **2. Routing Is Explicitly Disabled for a URL Pattern** - Use the RouteCollection.Ignore()  method to prevent routing from handling certain requests.  **What is the use of action filters in an MVC application?** Action Filters allow us to add pre-action and post-action behavior to controller action methods.  **If I have multiple filters impleted, what is the order in which these filters get  executed?** **1.** Authorization filters **2.** Action filters **3.** Response filters **4.** Exception filters  **NonActionAttribute**  **What are the different types of filters, in an asp.net mvc application?** **1.** Authorization filters **2.** Action filters **3.** Result filters **4.** Exception filters  **Give an example for Authorization filters in an asp.net mvc application?** **1.** RequireHttpsAttribute **2.** AuthorizeAttribute  **Which filter executes first in an asp.net mvc application?** Authorization filter   **What are the levels at which filters can be applied in an asp.net mvc application?** **1.** Action Method **2.** Controller **3.** Application **[b]Is it possible to create a custom filter?**[/b] Yes  **What filters are executed in the end?** Exception Filters  **Is it possible to cancel filter execution?** Yes  **What type of filter does OutputCacheAttribute class represents?** Result Filter  **What are the 2 popular asp.net mvc view engines?** **1.** Razor **2.** .aspx  **RenderBody method exists in the Layout page to render child page/view.**   1. **@RenderBody()**  RenderPage **RenderPage method also exists in the Layout page to render other page exists  in your application. A layout page can have multiple RenderPage method.**   1. @RenderPage("~/Views/Shared/\_Header.cshtml")   **What symbol would you use to denote, the start of a code block in razor views?** @  **What symbol would you use to denote, the start of a code block in aspx views?** <%= %>  **In razor syntax, what is the escape sequence character for @ symbol?** The escape sequence character for @ symbol, is another @ symbol  **When using razor views, do you have to take any special steps to proctect your asp.net mvc application from cross site scripting (XSS) attacks?** No, by default content emitted using a @ block is automatically HTML encoded to protect from cross site scripting (XSS) attacks.  **When using aspx view engine, to have a consistent look and feel, across all pages of the application, we can make use of asp.net master pages. What is asp.net master pages equivalent, when using razor views?** To have a consistent look and feel when using razor views, we can make use of layout pages.  Layout pages, reside in the shared folder, and are named as \_Layout.cshtml  **What are sections?** Layout pages, can define sections, which can then be overriden by specific views making use  of the layout. Defining and overriding sections is optional.  **What are the file extensions for razor views?** **1.** .cshtml - If the programming lanugaue is C# **2.** .vbhtml - If the programming lanugaue is VB  **How do you specify comments using razor syntax?** Razor syntax makes use of @\* to indicate the begining of a comment and \*@ to indicate the end. An example is shown below. @\* This is a Comment \*@ |

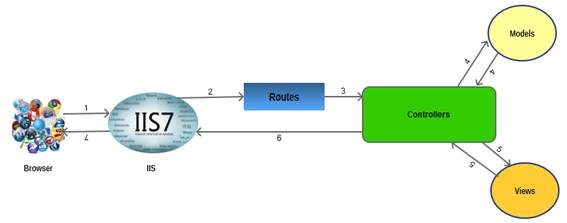
### What is MVC?

MVC is a framework pattern that splits an application’s implementation logic into three component roles: models, views, and controllers.

* **Model**: The business entity on which the overall application operates. Many applications use a persistent storage mechanism (such as a database) to store data. MVC does not specifically mention the data access layer because it is understood to be encapsulated by the Model.
* **View**: The user interface that renders the Model into a form of interaction.
* **Controller**: Handles a request from a View and updates the Model that results in a change of the Model's state.

To implement MVC in .NET, we need mainly three classes (View, Controller and the Model).

### Explain MVC Architecture?



The architecture is self explanatory. The browser (as usual) sends a request to IIS, IIS searches for the route defined in MVC application and passes the request to the controller as per route, the controller communicates with the model and passes the populated model (entity) to View (front end), Views are populated with model properties, and are rendered on the browser, passing the response to browse through IIS via controllers which invoked the particular View.

### What are the new features of MVC3?

ASP.NET MVC 3 shipped just 10 months after MVC 2 in Jan 2011. Some of the top features in MVC 3 included:

* The Razor view engine
* Support for .NET 4 Data Annotations
* Improved model validation
* Greater control and flexibility with support for dependency resolution and global action filters
* Better JavaScript support with unobtrusive JavaScript, jQuery Validation, and JSON binding
* Use of NuGet to deliver software and manage dependencies throughout the platform

### What are the new features of MVC4?

Following are the top features of MVC4:

* ASP.NET Web API
* Enhancements to default project templates
* Mobile project template using jQuery Mobile
* Display Modes
* Task support for Asynchronous Controllers
* Bundling and minification

### Explain “page lifecycle” of an ASP.NET MVC?

### Following processes are performed by ASP.NET MVC page:

1. App initialization
2. Routing
3. Instantiate and execute controller
4. Locate and invoke controller action
5. Instantiate and render view

### Advantages of MVC Framework?

1. Provides a clean separation of concerns between UI (Presentation layer), model (Transfer objects/Domain Objects/Entities) and Business Logic (Controller)
2. Easy to UNIT Test
3. Improved reusability of views/model. One can have multiple views which can point to the same model and vice versa
4. Improved structuring of the code

### Where do we see Separation of Concerns in MVC?

Between the data-processing (Model) and the rest of the application.

When we talk about Views and Controllers, their ownership itself explains separation. The views are just the presentation form of an application, it does not have to know specifically about the requests coming from controller. The Model is independent of View and Controllers, it only holds business entities that can be passed to any View by the controller as required for exposing them to the end user. The controller is independent of Views and Models, its sole purpose is to handle requests and pass it on as per the routes defined and as per the need of rendering views. Thus our business entities (model), business logic (controllers) and presentation logic (views) lie in logical/physical layers independent of each other.

### What is JSON Binding?

MVC 3 included JavaScript Object Notation (JSON) binding support via the new JsonValueProviderFactory, enabling the action methods to accept and model-bind data in JSON format. This is especially useful in advanced Ajax scenarios like client templates and data

binding that need to post data back to the server

### What is JSON Binding?

MVC 3 included JavaScript Object Notation (JSON) binding support via the   
new JsonValueProviderFactory, enabling the action methods to accept and model-bind data in JSON format. This is especially useful in advanced Ajax scenarios like client templates and data binding that need to post data back to

the server

#### 1. Explain MVC (Model-View-Controller) in general?

MVC (Model-View-Controller) is an architectural software pattern that basically decouples various components of a web application. By using MVC pattern, we can develop applications that are more flexible to changes without affecting the other components of our application.

* "Model" is basically domain data.
* "View" is user interface to render domain data.
* "Controller" translates user actions into appropriate operations performed on model.

#### 2. What is ASP.NET MVC?

#### ASP.NET MVC is a web development framework from Microsoft that is based on MVC (Model-View-Controller) architectural design pattern. Microsoft has streamlined the development of MVC based applications using ASP.NET MVC framework.

#### 3. Difference between ASP.NET MVC and ASP.NET WebForms?

ASP.NET Web Forms uses Page controller pattern approach for rendering layout, whereas ASP.NET MVC uses Front controller approach. In case of Page controller approach, every page has its own controller, i.e., code-behind file that processes the request. On the other hand, in ASP.NET MVC, a common controller for all pages processes the requests.  
Follow the link for the [difference between the ASP.NET MVC and ASP.NET WebForms](http://www.devcurry.com/2009/09/difference-between-aspnet-webforms-and.html#.UjRvRNLtaBI).

#### 4. What are the Core features of ASP.NET MVC?

Core features of ASP.NET MVC framework are:

* Clear separation of application concerns (Presentation and Business Logic)
* An extensible and pluggable framework
* Extensive support for ASP.NET Routing
* Support for existing ASP.NET features

Follow for detailed understanding of the above mentioned [core features](http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview).

#### 5. Can you please explain the request flow in ASP.NET MVC framework?

Request flow for ASP.NET MVC framework is as follows:

Request hits the controller coming from client. Controller plays its role and decides which model to use in order to serve the request further passing that model to view which then transforms the model and generates an appropriate response that is rendered to the client.

#### 6. What is Routing in ASP.NET MVC?

In case of a typical ASP.NET application, incoming requests are mapped to physical files such as .aspx file. ASP.NET MVC framework uses friendly URLs that more easily describe user’s action but are not mapped to physical files.

ASP.NET MVC framework uses a routing engine, that maps URLs to controller classes. We can define routing rules for the engine, so that it can map incoming request URLs to appropriate controller.

Practically, when a user types a URL in a browser window for an ASP.NET MVC application and presses “go” button, routing engine uses routing rules that are defined in Global.asax file in order to parse the URL and find out the path of corresponding controller.

#### 7. What is the difference between ViewData, ViewBag and TempData?

In order to pass data from controller to view and in next subsequent request, ASP.NET MVC framework provides different options i.e., ViewData, ViewBag and TempData.

Both ViewBag and ViewData are used to communicate between controller and corresponding view. But this communication is only for server call, it becomes null if redirect occurs. So, in short, it's a mechanism to maintain state between controller and corresponding view.

ViewData is a dictionary object while ViewBag is a dynamic property (a new C# 4.0 feature). ViewData being dictionary object is accessible using strings as keys and also requires typecasting for complex types. On the other hand, ViewBag doesn't have typecasting and null checks.

TempData is also a dictionary object that stays for the time of an HTTP Request. So, Tempdata can be used to maintain data between redirects, i.e., from one controller to the other controller.

#### 8. What are Action Methods in ASP.NET MVC?

I already explained about request flow in ASP.NET MVC framework that request coming from client hits controller first. Actually MVC application determines the corresponding controller by using routing rules defined in Global.asax. And controllers have specific methods for each user actions. Each request coming to controller is for a specific ActionMethod. The following code example, “ShowBooks” is an example of an Action method.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/653746/Top-ASP-NET-MVC-Interview-Questions)

public ViewResult ShowBooks(int id)

{

var computerBook = db.Books.Where(p => P.BookID == id).First();

return View(computerBook);

}

#### 9. Explain the role of Model in ASP.NET MVC?

One of the core features of ASP.NET MVC is that it separates the input and UI logic from business logic. Role of Model in ASP.NET MVC is to contain all application logic including validation, business and data access logic except view, i.e., input and controller, i.e., UI logic.

Model is normally responsible for accessing data from some persistent medium like database and manipulate it.

#### 10. What are Action Filters in ASP.NET MVC?

If we need to apply some specific logic before or after action methods, we use action filters. We can apply these action filters to a controller or a specific controller action. Action filters are basically custom classes that provide a means for adding pre-action or post-action behavior to controller actions.

For example:

* Authorize filter can be used to restrict access to a specific user or a role.
* OutputCache filter can cache the output of a controller action for a specific duration.

## Related Web Development Tutorials

* [Top 10 HTML5 Interview Questions](http://www.webdevelopmenthelp.net/2013/04/HTML5-Interview-Questions.html)
* [7 jQuery code snippets every web developer must have](http://www.webdevelopmenthelp.net/2013/09/7-jQuery-Code-Snippet.html)
* [WCF Vs ASMX Web Services](http://www.topwcftutorials.net/2012/06/wcf-vs-asmx-web-services.html)
* [Top 10 WCF Interview Questions](http://www.topwcftutorials.net/2012/06/wcf-top-10-interview-questions.html)
* [4 simple steps to enable tracing in WCF](http://www.topwcftutorials.net/2012/06/simple-steps-to-enable-tracing-in-wcf.html)

# ASP.NET MVC 4 Dependency Injection

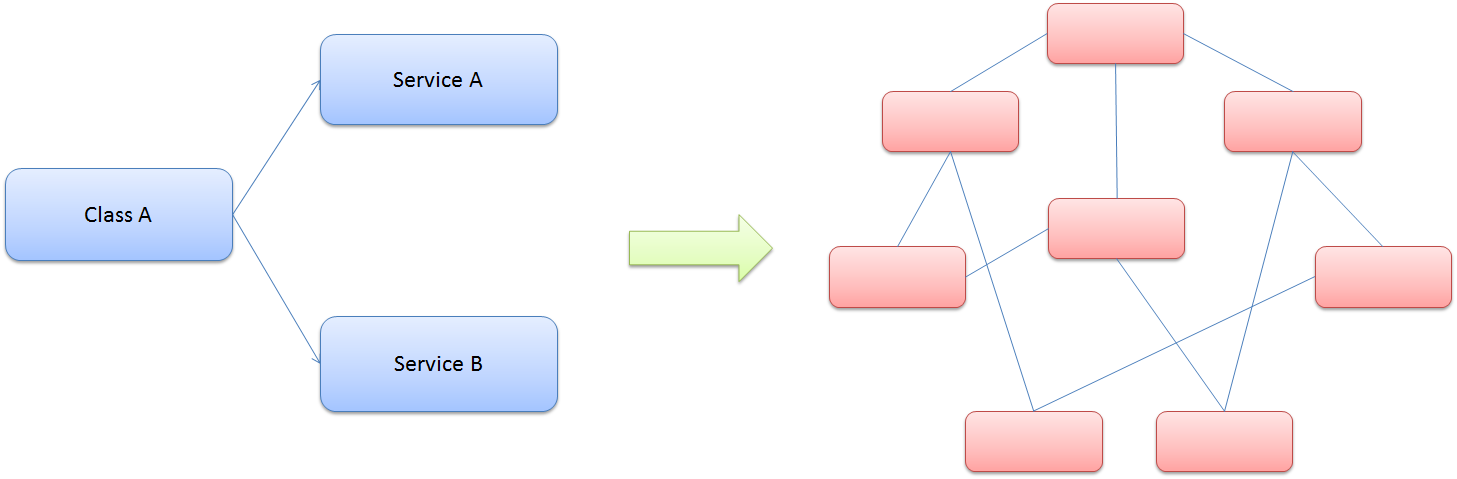
By [Web Camps Team](http://www.asp.net/mvc/tutorials/hands-on-labs/aspnet-mvc-4-dependency-injection#author-45262)|February 18, 2013

[Print](javascript:window.print();)

**Note:** This Hands-on Lab assumes you have basic knowledge of **ASP.NET MVC** and **ASP.NET MVC 4 filters**. If you have not used **ASP.NET MVC 4 filters** before, we recommend you to go over **ASP.NET MVC Custom Action Filters** Hands-on Lab.

All sample code and snippets are included in the Web Camps Training Kit, available at<http://go.microsoft.com/fwlink/?LinkID=248297&clcid=0x409>.

In **Object Oriented Programming** paradigm, objects work together in a collaboration model where there are contributors and consumers. Naturally, this communication model generates dependencies between objects and components, becoming difficult to manage when complexity increases.



Class dependencies and model complexity

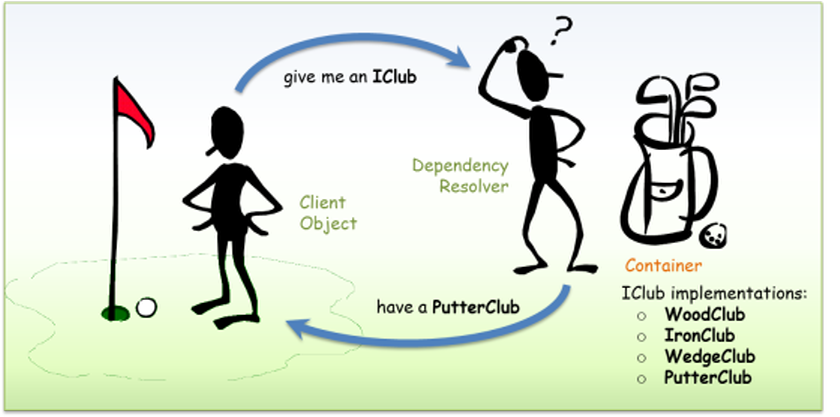
You have probably heard about the **Factory Pattern** and the separation between the interface and the implementation using services, where the client objects are often responsible for service location.

The Dependency Injection pattern is a particular implementation of Inversion of Control. **Inversion of Control (IoC)** means that objects do not create other objects on which they rely to do their work. Instead, they get the objects that they need from an outside source (for example, an xml configuration file).

**Dependency Injection (DI)** means that this is done without the object intervention, usually by a framework component that passes constructor parameters and set properties.

### The Dependency Injection (DI) Design Pattern

At a high level, the goal of Dependency Injection is that a client class (e.g. the golfer) needs something that satisfies an interface (e.g. IClub). It doesn't care what the concrete type is (e.g. WoodClub, IronClub, WedgeClub orPutterClub), it wants someone else to handle that (e.g. a good caddy). The Dependency Resolver in ASP.NET MVC can allow you to register your dependency logic somewhere else (e.g. a container or a bag of clubs).



Dependency Injection - Golf analogy

The advantages of using Dependency Injection pattern and Inversion of Control are the following:

* Reduces class coupling
* Increases code reusing
* Improves code maintainability
* Improves application testing

**Note:** Dependency Injection is sometimes compared with Abstract Factory Design Pattern, but there is a slight difference between both approaches. DI has a Framework working behind to solve dependencies by calling the factories and the registered services.

Now that you understand the Dependency Injection Pattern, you will learn throughout this lab how to apply it in ASP.NET MVC 4. You will start using Dependency Injection in the **Controllers** to include a database access service. Next, you will apply Dependency Injection to the **Views** to consume a service and show information. Finally, you will extend the DI to ASP.NET MVC 4 Filters, injecting a custom action filter in the solution.

In this Hands-on Lab, you will learn how to:

* Integrate ASP.NET MVC 4 with Unity for Dependency Injection using NuGet Packages
* Use Dependency Injection inside an ASP.NET MVC Controller
* Use Dependency Injection inside an ASP.NET MVC View
* Use Dependency Injection inside an ASP.NET MVC Action Filter

**Note:** This Lab is using Unity.Mvc3 NuGet Package for dependency resolution, but it is possible to adapt any Dependency Injection Framework to work with ASP.NET MVC 4.

### Prerequisites

You must have the following items to complete this lab:

* [Microsoft Visual Studio Express 2012 for Web](http://www.microsoft.com/visualstudio/eng/products/visual-studio-express-for-web) or superior (read [Appendix A](http://www.asp.net/mvc/tutorials/hands-on-labs/aspnet-mvc-4-dependency-injection#AppendixA) for instructions on how to install it).

### Setup

**Installing Code Snippets**

For convenience, much of the code you will be managing along this lab is available as Visual Studio code snippets. To install the code snippets run **.\Source\Setup\CodeSnippets.vsi** file.

If you are not familiar with the Visual Studio Code Snippets, and want to learn how to use them, you can refer to the appendix from this document "[Appendix C: Using Code Snippets](http://www.asp.net/mvc/tutorials/hands-on-labs/aspnet-mvc-4-dependency-injection#AppendixC)".

## Exercises

This Hands-On Lab is comprised by the following exercises:

1. [Exercise 1: Injecting a Controller](http://www.asp.net/mvc/tutorials/hands-on-labs/aspnet-mvc-4-dependency-injection#Exercise1)
2. [Exercise 2: Injecting a View](http://www.asp.net/mvc/tutorials/hands-on-labs/aspnet-mvc-4-dependency-injection#Exercise2)
3. [Exercise 3: Injecting Filters](http://www.asp.net/mvc/tutorials/hands-on-labs/aspnet-mvc-4-dependency-injection#Exercise3)

**Note:** Each exercise is accompanied by an **End** folder containing the resulting solution you should obtain after completing the exercises. You can use this solution as a guide if you need additional help working through the exercises.

Estimated time to complete this lab: **30 minutes**.

## http://www.codeproject.com/Articles/195434/Actions-in-ASP-NET-MVC

## Action Selectors

ASP.NET MVC 3 defines a set of Action selectors which determine the selection of an Action. One of them isActionName, used for defining an alias for an Action. When we define an alias for an Action, the Action will be invoked using only the alias; not with the Action name.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/195434/Actions-in-ASP-NET-MVC)

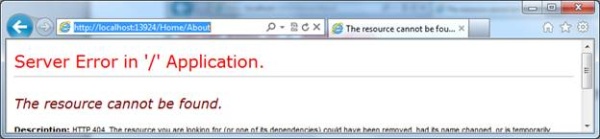
[ActionName("NewAbout")]

public ActionResult About()

{

return Content("Hello from New About");

}





ASP.NET has more Action selectors like HTTPPost and HTTPGet, which we will discuss later.

s

\*\*\*\*\*\*\*\*\*\*\*\*HTML Helpers in ASP.NET MVC\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

HTML Helper is just a method that returns a HTML string. The string can represent any type of content that you want. For example, you can use HTML Helpers to render standard HTML tags like HTML <input>, <button> and <img> tag

Inline Html Helpers

Built-In Html Helpers divided into 2 types

Standard Html Helpers

Strongly Typed HTML Helpers

Templated HTML Helpers

Custom Html Helpers

\*\*\*\*\*\*\*\*\*\*\*\*Persisting Data with TempData\*\*\*\*\*\*\*\*\*\*\*\*

TempData is used to pass data from current request to subsequent request (means redirecting from one page to another). It’s life is very short and lies only till the target view is fully loaded. But you can persist data in TempData by calling Keep() method.

TempData with Keep method

void Keep()

void Keep(string key)

\*\*\*\*\*\*\*\*\*\*\*AJAX Helpers in ASP.NET MVC\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AJAX Helpers are used to create AJAX enabled elements like as Ajax enabled forms and links which performs request asynchronously. AJAX Helpers are extension methods of AJAXHelper class which exist in System.Web.Mvc.Ajax namespace.

Property Description

Url Specify the URL that will be requested from the server.

Confirm Specify a message that will be displayed in a confirm dialog to the end user. When user clicks on OK button in the confirmation dialog, the Ajax call performs.

OnBegin Specify a JavaScript function name which is called at the beginning of the Ajax request.

OnComplete Specify a JavaScript function name which is called at the end of the Ajax request.

OnSuccess Specify a JavaScript function name which is called when the Ajax request is successful.

OnFailure Specify a JavaScript function name which is called if the Ajax request fails.

LoadingElementId Specify progress message container’s Id to display a progress message or animation to the end user while an Ajax request is being made.

LoadingElementDuration Specify a time duration in milliseconds that controls the duration of the progress message or animation.

UpdateTargetId Specify the target container’s Id that will be populated with the HTML returned by the action method.

InsertionMode Specify the way of populating the target container. The possible values are InsertAfter, InsertBefore and Replace (which is the default).

\*\*\*\*\*\*\*\*\*\*\*Understanding ASP.NET MVC Scaffolding\*\*\*\*\*\*\*\*\*\*\*\*

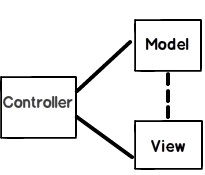
\*\*\*\*\*\*\*\*\*\*\*\* strong type view in MVC application\*\*\*\*\*\*\*\*\*\*\*\*

We can send data in form of object

## 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.

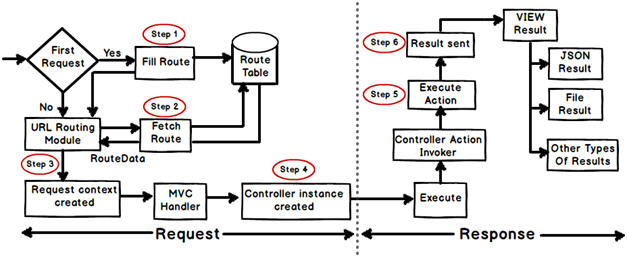
* The View is responsible for the look and feel.
* Model represents the real world object and provides data to the View.
* The Controller is responsible for taking the end user request and loading the appropriate Model and View.



***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?

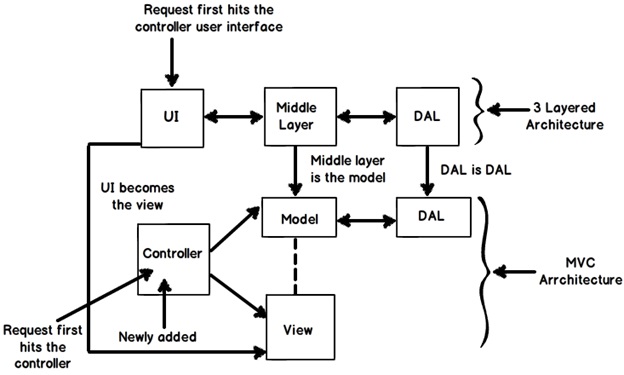
There are two big benefits of MVC:

* 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?

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

## What is the difference between each version of MVC?

Below is a detailed table of differences. But during an interview it’s difficult to talk about all of them due to time limitation. So I have highlighted the important differences that 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 * Model-Validator Providers * New RequireHttpsAttributeAction Filter * Templated Helpers * Display Model-Level Errors | * **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 |

## 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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<%= 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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<%= 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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

Html.TextBox("CustomerCode")

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

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

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 routescollection using the maproute function. Below is the underlined code which shows how the URL structure and mapping with controller and action is defined.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

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 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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<%= 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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

[HttpGet]

public ViewResult DisplayCustomer(int id)

{

Customer objCustomer = Customers[id];

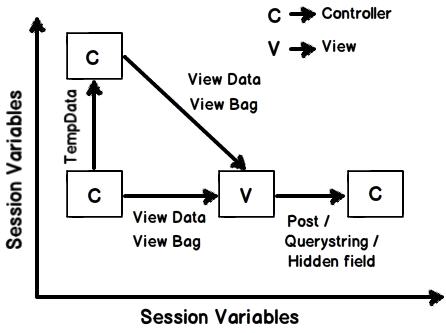
return View("DisplayCustomer",objCustomer);

}

## How can we maintain sessions in MVC?

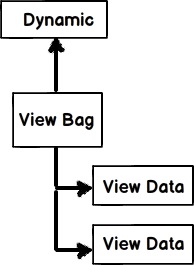
Sessions can be maintained in MVC by three ways: tempdata, viewdata, and viewbag.

## 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***

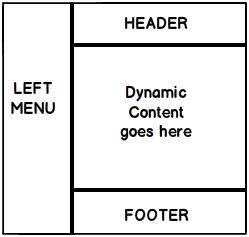
* **Session variables -** By using session variables we can maintain data from any entity to any entity.
* **Hidden fields and HTML controls -** Helps to maintain data from UI to controller only. So you can send data from HTML controls or hidden fields to the controller using POST or GET HTTP methods.

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 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.

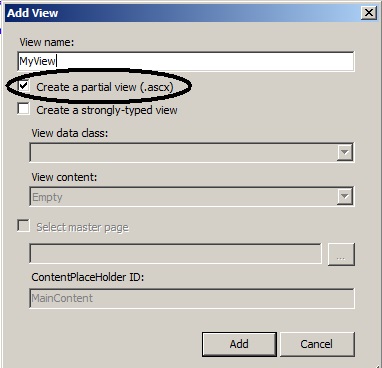


***Figure: Partial views in MVC***

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.



***Figure: Create partial view***

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

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<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 simpleCustomer 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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<% 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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

public ActionResult PostCustomer(Customer obj)

{

if (ModelState.IsValid)

{

obj.Save();

return View("Thanks");

}

else

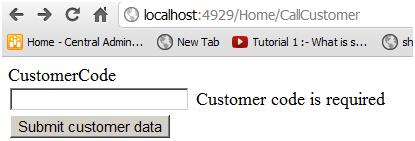
{

return View("Customer");

}

}

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



***Figure: Validations in MVC***

## Can we display all errors in one go?

Yes, we can; use the ValidationSummary method from the Html helper class.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<%= Html.ValidationSummary() %>

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

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

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

[StringLength(160)]

public string FirstName { get; set; }

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

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

[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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

[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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

TryUpdateModel(NewCustomer);

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

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<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>

The second step is to call the EnableClientValidation method.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<% 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:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<%=DateTime.Now%>

In Razor, it’s just one line of code:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

@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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

[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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

[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 calledgetCustomer. So now the submit action click will be an asynchronous AJAX call.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<script language="javascript">

function OnSuccess(data1)

{

*// Do something here*

}

</script>

<div>

<%

var AjaxOpt = new AjaxOptions{OnSuccess="OnSuccess"};

%>

<% using (Ajax.BeginForm("getCustomer","MyAjax",AjaxOpt)) { %>

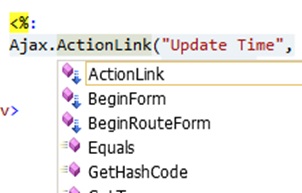
<input id="txtCustomerCode" type="text" /><br />

<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 the Ajax.ActionLink function as shown in the below code.



***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 namedDateDiv.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

<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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

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.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

function GetData()

{

var url = "/MyAjax/getCustomer";

$.post(url, function (data)

{

$("#txtCustomerCode").val(data.CustomerCode);

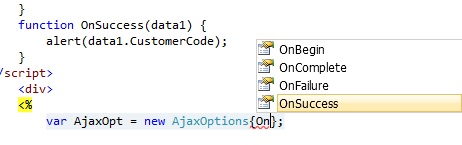
$("#txtCustomerName").val(data.CustomerName);

}

)

}

## What kind of events can be tracked in AJAX?



***Figure: Tracked in AJAX***

## What is the difference between ActionResult and ViewResult?

* ActionResult is an abstract class while ViewResult derives from the ActionResult class. ActionResulthas 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 aViewResult or JsonResult.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/556995/MVC-interview-questions-with-answers)

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?

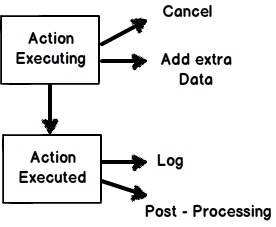
**Note**: It’s difficult to remember all the 12 types. But some important ones you can remember for the interview areActionResult, ViewResult, and JsonResult. Below is a detailed list for your interest:

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 IActionFilterinterface has two methods: OnActionExecuted and OnActionExecuting. We can implement pre-processing logic or cancellation logic in these methods.

Here we discuss some of the major functionalities of MVC 3, that were game changing.  
  
These functionalities are:

* jQuery Validation
* Unobtrusive JavaSctipt
* JSON Binding
* Dependency resolution
* Global action filters
* Razor view engine

Routing:

. The ASP.NET Routing module is responsible for mapping incoming browser requests to a particular **MVC** controller actions.

Routing uses URL patterns to identify proper controllers as well as actions. The URL pattern for routes in **MVC**applications typically is domain/{controller}/{action}/{id}.

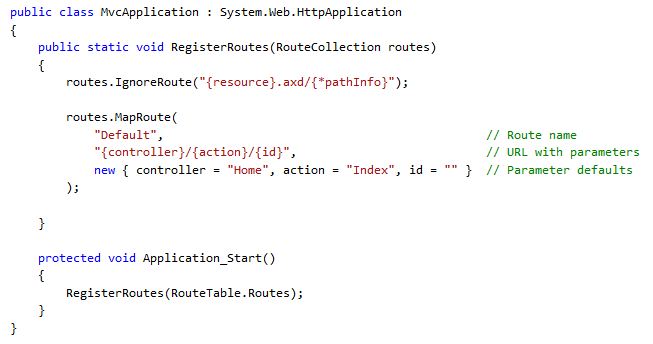
## How to Define Controllers

Physically a controller is just another class that must be post-fixed with the "Controller" word that can expose controller actions. When a user enters a URL into the browser, the **MVC** application uses routing rules to parse the URL and to determine the path of the controller. Any method that is defined as public is considered an action method. There are some additional requirements that must be satisfied by a controller action.

1. A method used as a controller action cannot be overloaded.
2. Controller action cannot be a static method.

## How to Define Custom Routing

Any class derived from the ControllerBase class and having a name that ends with "Controller" does not require any custom routing declaration. But you can define your own route as well. A custom route constraint enables you to prevent a route from being matched unless some custom condition is matched. If you want to add custom routes in an **MVC** application, you use the MapRoute(RouteCollection, String, String) method in Global.ascx.



##### When you define a route, you can assign a default routing value for a parameter. The default is utilized if the value for that parameter is not present in the URL.

##### 3. Bundling and Minification

Bundle and minification framework will reduce the number of hits internally made by HTTP request for a web page. Also, this will reduce the size of overall request with minification.

**Bundling**: Bundling feature starts with ASP.NET 4.5. Bundle feature will combine the multiple files into a single file. We can create JavaScript, CSS and other bundles. Bundling will improve the performance of the web request as this will have less number of files which means less HTTP requests.

**Minification**: Minification will just do the code optimization to the scripts and CSS files by removing the unnecessary white space, comments and shortening variable names to one characters.

**Controlling Bundling and Minification**: Bundling and minification is enabled or disabled by setting the value in*Web.config* file compilation debug attribute.

## MVC 4 Framework

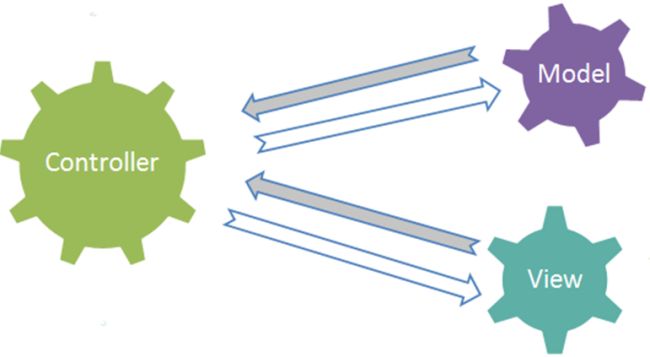
##### 1. Add Controller to other project folder 2. Task Support for Asynchronous Controllers 3. Bundling and Minification

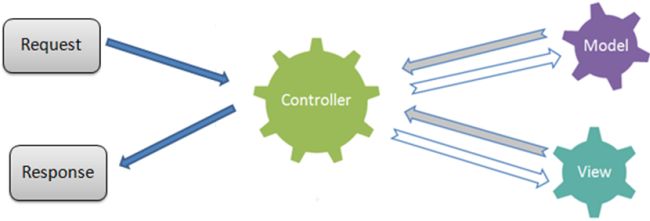
##### 4. Enabling Logins from Facebook and Other Sites Using OAuth and OpenID MVC 4 has a default template “Internet Project” that supports for OAuth and OpenID login using theDotNetOpenAuth library. 5. App\_Start folder and separate classes

**7. Azure SDK**

**8. Database Migrations**

MCV 4 included the entity framework 5. Entity framework 5 supports the data migration, this is one of the great features available.

The Model View Controller (MVC) pattern is an architectural design principal that separates components of web applications. This separation gives us more control over individual parts of an application. It lets us easily develop modify and test applications.  
  
The ASP.NET MVC Framework is a lightweight highly testable and presentation framework that integrates with existing ASP.NET features, smarter pages and membership based authentication.  
  
The MVC framework is defined in the System.Web.MVC assemblies.  
  


  
  
In this whenever the user requests a page the first hit goes to the controller.  
  
Then the request goes to the view and a View is returned.  
  
**Controller / View**  
**Views**

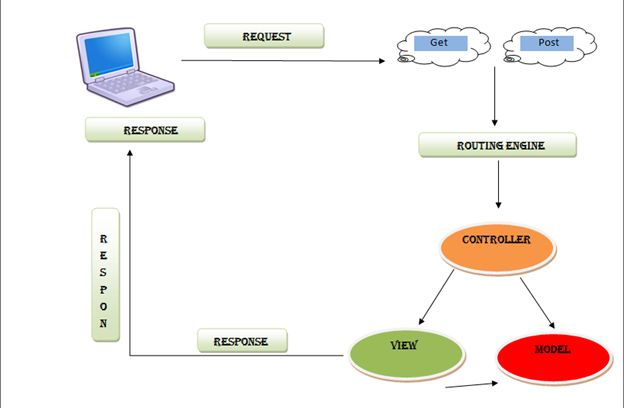
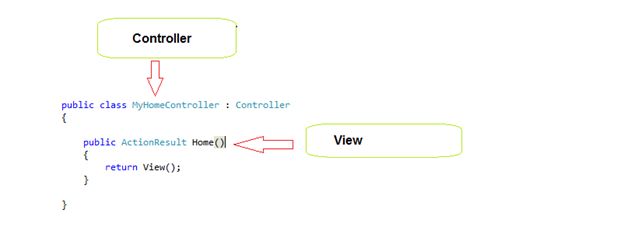
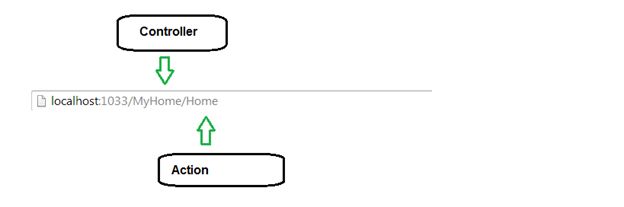
1. Present data to the user.
2. Read only view as well as forms.
3. Minimal display only logic
4. Most often the views are created from the model data.

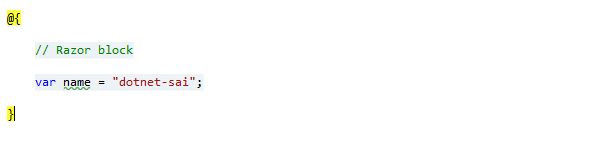
**Controller**

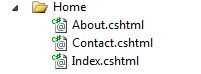
1. Responds to the request.
2. Connects the model to the view.
3. Invokes the model code as appropriate.
4. Typically controllers read data from a view, control user input and send input data to the model.

**Model**

1. Domain Specific representation of data
2. Business logic
3. Often the model objects retrieves data from (and stores data to) a database.

**How MVC works   
  
**  
**Flow of M - V - C**  
Here in the diagram when the user requests a page it either does a GET () or a POST ().  
  
The request goes to the Routing engine.  
  
Then the Controller (Front Controller) is hit.  
  
Then it will see request and return the requested view.   
  
**MVC Convention (How to call a Page in MVC)  
  
  
  
What is Razor?**

1. Razor is markup syntax for adding server based code to a webpage.
2. Razor is server-side markup syntax much as PHP.
3. Razor supports C# and VB Programming Languages.
4. A Razor code block is enclosed in:  
     
   @{ }  
     
   
5. The code statements end with a semicolon ( ; )
6. Variables are declared by Var
7. Strings are enclosed with a Quotation mark.
8. C# code is case sensitive.   
   e.g  
     
   

**C# file have Extensions with .cshtml  
  
  
  
What id Data Annotation?**In MVC you can validate data in the UI and before sending to the database auto-magically.  
  
If you add Data Annaotation Validators to your model classes then something.  
  
Data annotations are not the only way to specify validation rules, but they're easy to implement.  
  
Add "using System.ComponentModel.DataAnnotations;" to your model.  
  
The following are Data Annotations we can use in a Model:

1. **DisplayName:** Provides a general-purpose attribute that lets you specify localizable strings to display.
2. **Required:** A value is required
3. **DataType:** The data type annotation can be used to specify the data type for validation.
4. **StringLength:** Max. Length of array or string data allowed
5. **DisplayFormat:** Specify the display format for a property like various formats for a Date property.
6. **ReqularExpression:** validate the value of a property by a specified Regular Expression pattern.
7. **Range:** Numeric range constraints for the data field value
8. **MaxLength:** Specify max length for a string property.
9. **Bind:** Specify fields to include or exclude when adding a parameter or form values to model properties.
10. **Compare:** The Compares property compares two properties
11. **Key:** Denotes one or more properties that uniquely identify an entity.

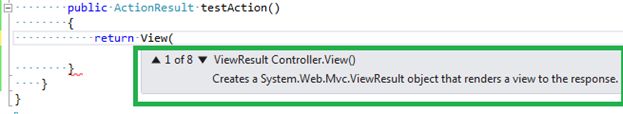
After this start with the first tutorial.

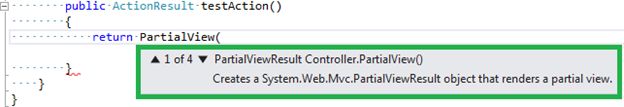
This is one hot question in .NET job interviews, I have heard from many of my friends. Developers with limited hands-on experience with MVC should be able to provide an answer to the question, because the scenario is very common and every now and then it needs to return something from the controller to the presentation environment.  
  
We are very familiar with the “ActionResult” class that is the base class of many classes and we can return an object of those classes. The class hierarchy is as in the following:

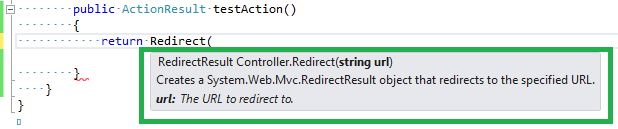
System.Object   
System.Web.Mvc.ActionResult

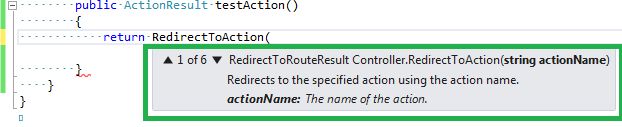
* System.Web.Mvc.ContentResult
* System.Web.Mvc.EmptyResult
* System.Web.Mvc.FileResult
* System.Web.Mvc.HttpStatusCodeResult
* System.Web.Mvc.JavaScriptResult
* System.Web.Mvc.JsonResult
* System.Web.Mvc.RedirectResult
* System.Web.Mvc.RedirectToRouteResult
* System.Web.Mvc.ViewResultBase

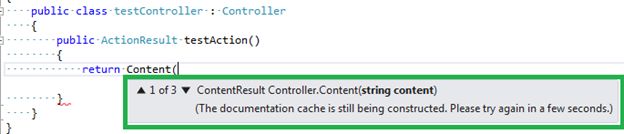
In this example we will see all of the derived classes that is inherited from the “ActionResult” base class. So, let's start one by one.  
  
**Return View**  
This is a most common and very frequently used type. We see that we can pass eight parameters when we return the view. We can specify the view name explicitly or may not.

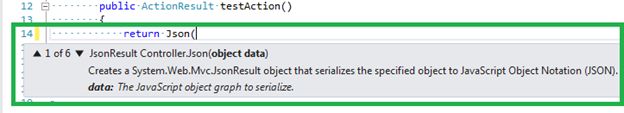
  
  
**Return partial View**  
The concept of a partial view is very similar to the master page concept in Web Form applications. The partial view is nothing but  pagelet, that we can return from the controller and that merges with the main view and generates one concrete HTML page.

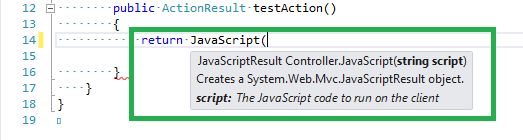
  
  
It may take 4 parameters to render in the partial view.  
  
**Redirect**  
This is equivalent to Response.redirect() or Server.Transfer() functions. It takes the URL path to be redirect , though we can use Response.Redirect() or Server.Transfer() in MVC too.

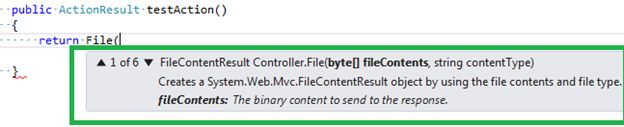
  
  
**Redirect To Action**  
Sometimes it is necessary to call another action after completion of one action, this is very similar to a function call in traditional function oriented programming or Object Oriented Programming. It may take 6 parameters. The first parameter is very simple, only action name.

  
  
**Return content**  
This is useful when we want to return a small amount of strings from a controller/action. It takes three parameters. The first one is a simple string and the remaining two are strings with little information.

  
  
**Return JSON**  
This is very useful when we don't want an entire HTML page but only want a value. Generally in AJAX-based single-page applications we do not load an entire page again and again but load fresh data from the DB using AJAX. In this scenario we can return only a JSON object and in the success function of jQuery ajax (let's assume we are using the jQuery library to implement AJAX) we can just manipulate data.

  
  
**Return JavaScript**  
When we wanted to return a JavaScript string , we may use this function. It takes only one parameter, the string only.

  
  
**Return File**  
We are allowed to return a binary file if needed from a controller. It takes 6 parameters maximum.

  
  
**Conclusion**  
Those are all of the return types in the action in a MVC controller, but in reality we do not get a chance to use all of them, in my limited experience people usually like to return  View() from the action. What do you say? Happy learning.

**s**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Dotnetfunda\*\*\*\*\*\*\*\*\*\*\*\*

**2. What is Web API ‘s in Asp.Net MVC 4 ?**  
Ans.

* Web API is a new framework for consuming & building HTTP Services.
* Web API supports wide range of clients including different browsers and mobile devices.
* It is very good platform for developing RESTful services since it talk’s about HTTP.

**. What is the use of web API ? Why Web API needed, If you have already RESTful services using WCF ?**

Ans. Yes, we can still develop the RESTful services with WCF, but there are two main reasons that prompt users to use Web API instead of RESTful services.

* ASP.NET Web API is included in ASP.NET MVC which obviously increases TDD (Test Data Driven) approach in the development of RESTful services.
* For developing RESTful services in WCF you still needs lot of config settings, URI templates, contract’s & endpoints which developing RESTful services using web API is simple

**6. What is the use of Display Modes?  
Ans.**

* **View** can be changed automatically based on browser(For mobile and desktop browser’s)

Display Modes is newly added feature in ASP.NET MVC 4. Views selected automatically by application depending on the browser. Example: If a desktop browser requests login page of an application it will return Views\Account\Login.cshtml view & if a mobile browser requests home page it will return Views\Account\Login.mobile.cshtml view.

### ViewBag

It is most popular data carrier in [MVC](http://www.thingswelearned.com/CatAct/Articles/f0f57977-7ac0-47bb-970a-39643b2b9b72/) because ViewBag is dynamic property and doesn't requires typecasting.

### ViewData

It is dictionary object derived from ViewDataDictionary class, requires typecasting for complex data type and need check for null values for to avoid errors.

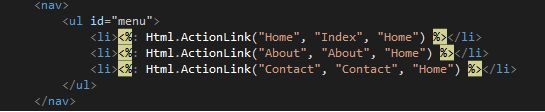
### TempData

It is dictionary object derived from TempDataDictionary class, data is stored in short lives session.

# Differences Between Razor and ASPX View Engine in MVC

1. **Performance**  
   * Razor Engine is a little slow compared to Aspx Engine.
   * Aspx Engine is faster compared to Razor Engine.

1. **Syntax**  
   * ‘@’ symbol uses in Razor Engine to write the code. @Html.ActionLink("Login", "LoginView")
   * ‘<%:’ delimiters use as starting point and ‘ %>’ use as ending point. You can write the code between them in ASPX Engine.

<%: Html.ActionLink("Login ", " LoginView ") %>    
  


1. **Cross-Site Scripting Attacks**  
   * Razor Engine prevents Cross-Site Scripting Attacks, in other words it encodes the script or HTML tags like <,> before rendering to view.
   * ASPX Engine does not prevent Cross-Site Scripting Attacks, in other words any script saved in the database will be fired while rendering the page.

* [Animated Image Slide Show in C#](http://www.c-sharpcorner.com/UploadFile/asmabegam/animated-image-slide-show/)
* [CRUD Operations in MVC](http://www.c-sharpcorner.com/UploadFile/2072a9/crud-operations-in-mvc/)
* [Create the Configuration Database F ...](http://www.c-sharpcorner.com/UploadFile/Ashush/create-the-configuration-database-for-standalone-installatio/)

READER LEVEL:

ARTICLE

[](http://www.c-sharpcorner.com/authors/ff2f08/jignesh-trivedi.aspx)

[20](http://www.c-sharpcorner.com/Members/rank/) [5,701](http://www.c-sharpcorner.com/Authors/AuthorRanking.aspx?AuthorId=ff2f08)

# ASPX View Engine VS Razor View Engine

Posted by [Jignesh Trivedi](http://www.c-sharpcorner.com/authors/ff2f08/jignesh-trivedi.aspx) in [Articles](http://www.c-sharpcorner.com/Articles/) | [ASP.NET MVC](http://www.c-sharpcorner.com/1/226/Asp-Net-mvc.aspx) on September 17, 2013

View Engine renders the HTML to the browser. The view engine templates have a different syntax than the implementation.

* [Tweet](http://twitter.com/share)
* [inShare](javascript:void(0);)
* 0
* [2](http://www.c-sharpcorner.com/UploadFile/ff2f08/aspx-view-engine-vs-razor-view-engine/#ReadAndPostComment)
* 14400

**Introduction**

View Engine renders the HTML to the browser. The view engine templates have a different syntax than the implementation. By default ASP.Net MVC supports ASPX and the Razor View Engine. There are many more third-party view engines, like Spark, Nhaml and so on also available for MVC. We can also write our own view engine.  
  
**ASPX View Engine**  
  
The syntax used for writing a view with the ASPX View Engine is the same as the syntax used in ASP.Net web forms. The file extensions are also the same as for ASP.NET web forms (like .aspx, .ascx, .master). This view engine is the default view engine for MVC 1.0 and MVC 2.0. Implementing the unit testing framework with the ASPX View Engine is very difficult. ASPX uses "<%= %>" or "<%: %>" to render server-side content. We can choose any language with the CodeDom provider. There are on demand or precompiled views supported by an ASPX View Engine. The ASPX View Engine is also known as the Web Form View Engine.  
  
Loop and condition example with ASPX View Engine:

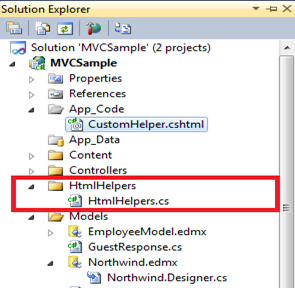
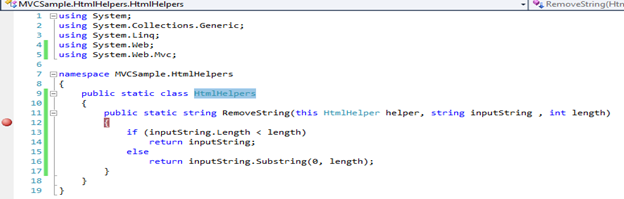
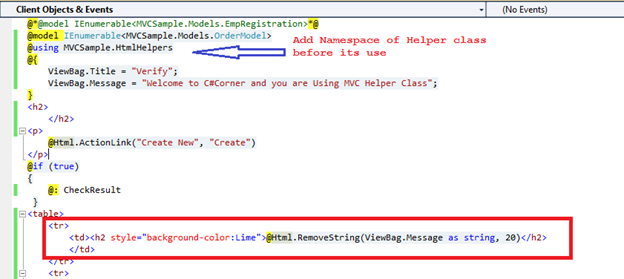
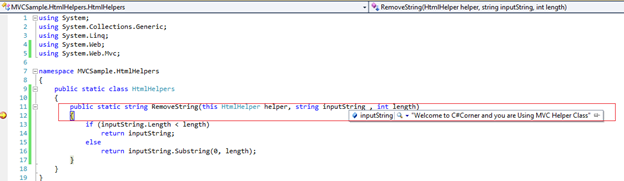
<ul>  
    <%foreach (var item in Products)  
        {  %>  
             <% if (item.IsInStock)  
                  { %>  
                         <p><%=item.ProductName%> is in stock</p>  
                   <% }  
                 else  
                  { %>  
                         <p><%=item.ProductName%> is not in stock</p>  
                  <% } %>  
    <%} %>  
</ul>  
  
**Razor View Engine**  
  
The Razor View Engine is an advanced view engine, available with MVC 3.0 and later versions. Razor uses the "@" character instead of "<% %>" as used by the ASPX View Engine. Razor does not require the code block to be closed, the Razor View Engine parsed itself and it is able to decide during run time that it is a presentation element (content) and that it is a code element. The Razor View Engine is compatible with a unit testing framework. The Razor template does not require the controller or webserver to host it, so views written in Razor are fully testable. The file extension of a Razor view is cshtml (for C#) and vbhtml (for VB.NET). By default all text from the @ expression is HTML encoded. Razor is not a new language. It is easy to learn. The main advantage of Razor, is that there is less transition between HTML and code because Razor provides an optimized syntax to generate HTML using a code focused templating approach.  
  
Loop and condition example with Razor View Engine:  
  
<ul>  
    @foreach (var item in Products)  
    {  
               @if(item.IsinStock)  
               {     
                   @item.ProductName is in stock  
               } else {  
                   @item.ProductName is in stock  
               }   
    }  
</ul>  
  
One of the disadvantages of Razor is, it is not supported by visual editors like Dream Viewer.  
  
**Advantages of Razor View Engine**

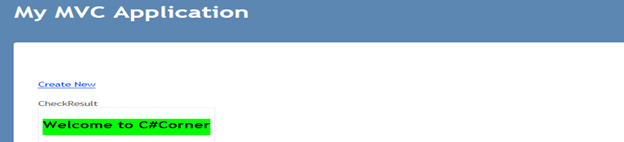
* Easy to Learn: Razor is easy to learn. We can also use our existing HTML skills.
* It is Compact, Expressive, and Fluid. Razor helps us to minimize the coding and provide us a fast and fluid coding work flow.
* The parser (available with Razor) is smart enough. It is also able to decide at run time what is a code element and what is a content element.   
    
  For example, in the following code the @ character is also part of an email address, but Razor is smart enough to identify which is code and which is static content.  
    
  <p>  
          please contact to abc@gmail.com to more information  
         Current Date time : @DateTime.Now  
  </p>
* Razor is not a new language but it is markup so that we can also use Razor with any language like C# and VB.
* Razor also supports the concept of layout pages (the same as Master Pages in ASPX View Engine), that allows us to define a common site template, in other words a common look and feel across all the pages within a web site/application.
* Razor does not require any special tool to write markup. We can also write our markup code with any old plain text editor like Notepad.
* The Razor View Engine is designed such that it also supports unit test views without requiring a controller and web server. This can be hosted in any unit project. There is no special application domain required.
* ASP.NET MVC has HTML helpers that are methods that can be invoked within a code block. All existing HTML extension methods can be used with a Razor View Engine without any code changes.
* The code looks clean.
* Powerful built-in validation of markup that helps us to avoid unwanted runtime exceptions due to errors in the view.
* The Razor View Engine has the section concept that is equivalent to content placeholders in the ASPX View Engine and that can be optional.
* The @model directive provides a cleaner and more concise way to define a strongly typed model.  
    
  Example:  
  @model List<MyMVCapplication.EmployeeMaster>  
  or  
  @model MyMVCapplication.EmployeeDetails

**Razor View Engine VS ASPX View Engine**

|  |  |
| --- | --- |
| **Razor View Engine** | **ASPX View Engine (Web form view engine)** |
| The namespace used by the Razor View Engine is System.Web.Razor | The namespace used by the ASPX View Engine is System.Web.Mvc.WebFormViewEngine |
| The file extensions used by the Razor View Engine are different from a web form view engine. It uses cshtml with C# and vbhtml with vb for views, partial view, templates and layout pages. | The file extensions used by the Web Form View Engines are like ASP.Net web forms. It uses the ASPX extension to view the aspc extension for partial views or User Controls or templates and master extensions for layout/master pages. |
| The Razor View Engine is an advanced view engine that was introduced with MVC 3.0. This is not a new language but it is markup. | A web form view engine is the default view engine and available from the beginning of MVC |
| Razor has a syntax that is very compact and helps us to reduce typing. | The web form view engine has syntax that is the same as an ASP.Net forms application. |
| The Razor View Engine uses @ to render server-side content. | The ASPX/web form view engine uses "<%= %>" or "<%: %>" to render server-side content. |
| By default all text from an @ expression is HTML encoded. | There is a different syntax ("<%: %>") to make text HTML encoded. |
| Razor does not require the code block to be closed, the Razor View Engine parses itself and it is able to decide at runtime which is a content element and which is a code element. | A web form view engine requires the code block to be closed properly otherwise it throws a runtime exception. |
| The Razor View Engine prevents Cross Site Scripting (XSS) attacks by encoding the script or HTML tags before rendering to the view. | A web form View engine does not prevent Cross Site Scripting (XSS) attack. |
| The Razor Engine supports Test Driven Development (TDD). | Web Form view engine does not support Test Driven Development (TDD) because it depends on the System.Web.UI.Page class to make the testing complex. |
| Razor uses "@\* â€¦ \*@" for multiline comments. | The ASPX View Engine uses "<!--...-->" for markup and "/\* â€¦ \*/" for C# code. |
| There is only three transition characters with the Razor View Engine. | There are only three transition characters with the Razor View Engine. |
| The Razor View Engine is a bit slower than the ASPX View Engine. | |

# Extension Helpers Method in MVC

There are many ways to create your own helpers using Razor syntax. I am using one of them. The Helper Method is an optimum way to do it.  
   
So let's start and create a Helper Method.  
  
First we'll start with Extension Method syntax. I have created a sample application and it has HtmlHelpers.cs in the HtmlHelpers folder.  
  
  
   
Open the HtmlHelpers.cs file and you will find the following code snippet, which is an Extension Method that takes two parameters and removes a string using the substring method.  
  
  
   
So far we have declared an Extension Method, now to use it in our View page (.cshtml). I'll use it in the Verify.cshtml page. Here is the code snippet of that page.  
  
  
   
As you can see we are using the RemoveString() Helper Method in the Verify.cshtml page, that caters the raw string (raw string as declared in the ViewBag.Message property) using two parameters being passed.  
   
Now let's run this and see the execution:  
  
  
   
Notice that it took the complete string that we've passed, like @Html.RemoveString(ViewBag.Message as string, 20) and performs the code down the level.

Again Press F5 and see the result.  
  
  
  
   
Notice that it only gives you "Welcome to C#Corner" out of the complete string.  
   
So this is how to create a Helper Method on demand in Razor view Engine. The sample application is attached as a reference.

# Removing or Customizing View Engines in MVC