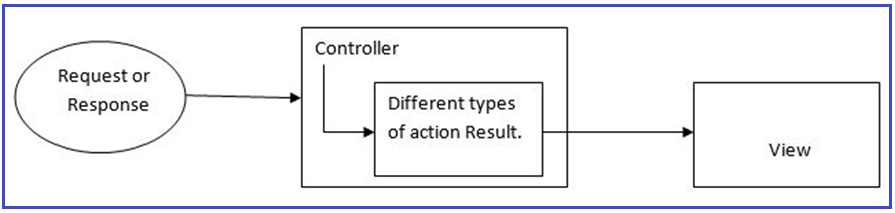
**Action Results in ASP.NET MVC**

**Action Results in ASP.NET MVC**



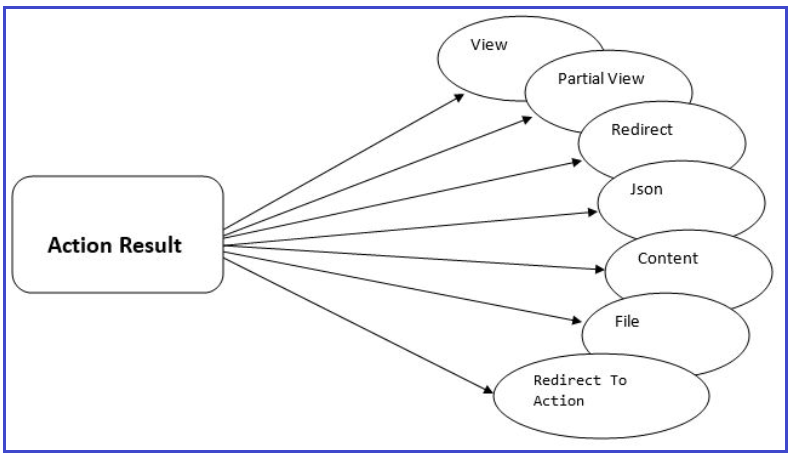
**What is the Action Method in ASP.NET MVC?**

Before going to understand Action Results, first, we need to understand what are action methods in ASP.NET MVC Application. All the public methods inside a Controller which respond to the URL are known as Action Methods. When creating an Action Method, we must follow the below rules.

1. The action method must be **public**.
2. It cannot be **overloaded**
3. It cannot be a **static** method
4. **ActionResult** is the base class of all the result types that an action method returns.

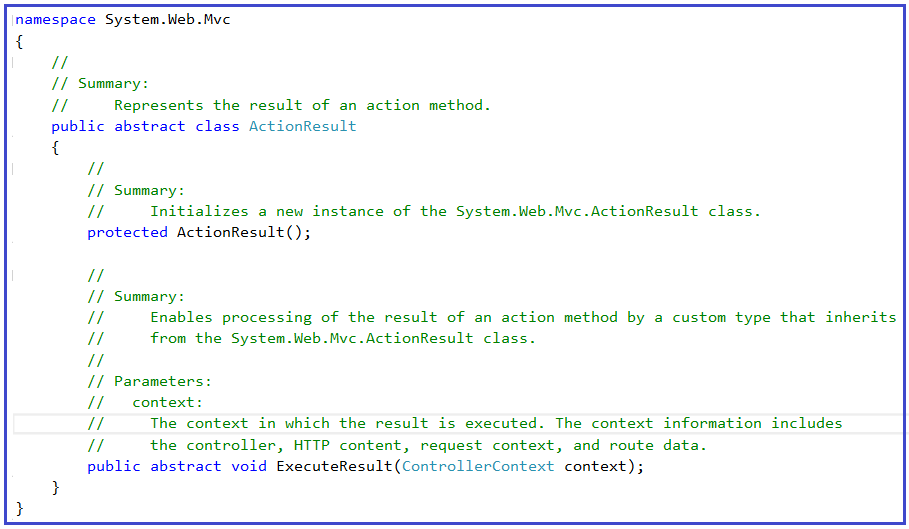
**What is the Action Result in ASP.NET MVC?**

Action Result is the return type of an action method. The action result is an abstract class. It is the base class for all types that an action method returns. As you can see in the below image, View, Partial View, Redirect, Json, Content, File, Redirect To Action, etc. are derived from the abstract Action Result class and these types can also be used as the return type of an action method.



**ActionResult Class:**

The following diagram shows the signature of the ActionResult class. As you can see in the below image, ActionResult is an Abstract class having one constructor and one method. The constructor is basically used to initializes a new instance of the ActionResult class. The ExecuteResult method enables the processing of the result of an action method by a custom type that inherits from the ActionResult class. The ExecuteResult method takes one parameter i.e. context i.e. the context in which the result is executed. The context includes the information of Controller, HTTP Content, request context, and route data.



**Why is ActionResult an abstract class in ASP.NET MVC?**

It’s because different controller action methods can return different types of results as per the business needs and still the ASP.NET MVC Framework handles them properly. If you mention the return type of an action method as ActionResult, then this action method can return any type which is derived from the ActionResult abstract class.

**Types of Action Results**

There are many different types of Action Results that an action method can return in ASP.NET MVC. Each Action Result returns a different type of result format. ActionResult is the base class of all the result types. The following are the Result types that an action method can return in ASP.NET MVC Application.

1. **ViewResult –** Represents HTML and markup.
2. **PartialViewResult –** Represents HTML and markup.
3. **EmptyResult** – Represents no result.
4. **RedirectResult**– Represents a redirection to a new URL.
5. **RedirectToActionResult**– It is returning the result to a specified controller and action method
6. **JsonResult**– Represents a JavaScript Object Notation result that can be used in an AJAX application.
7. **JavaScriptResult**– Represents a JavaScript script.
8. **ContentResult**– Represents a text result.
9. **FileContentResult**– Represents a downloadable file (with the binary content).
10. **FilePathResult**– Represents a downloadable file (with a path).
11. **FileStreamResult**– Represents a downloadable file (with a file stream).

Many of the derived classes we’re going to discuss have associated helpers. These helpers provide shortcuts to the constructor methods of their related Results. That allows us to write **return View()** rather than **return new ViewResult().**

**Example:**

In the below example, the action method return type is ActionResult, and the method returning two types of results. First is, **return View** which is similar to return new ViewResult(). Here, View() is the shortcut for new ViewResult(). Second is, return RedirectToAction(); which is similar to return new RedirectToActionResult(). Here, RedirectToAction() is the shortcut for new RedirectToActionResult().

**<**span style="color: #000000; font-family: arial, helvetica, sans-serif;"**>public** ActionResult ChooseView**()**

**{**

**if** **(**DateTime.Now.Day % 2 == 0**)**

**{**

**return** View**(**"View1"**)**;

**}**

**else**

**{**

**return** RedirectToAction**(**"View2"**)**;

**}**

**}**

**<**/span**>**

**Categorized of Action Results:**

You may be guessed that the above example implementation is done because ActionResult has a lot of derived classes, and you are absolutely right. But what exactly are these different kinds of results? These results are categorized into three sections:

1. **Content-returning**
2. **Redirection**
3. **Status**.

Let’s have a look at these three categorized:

**Content-Returning Action Result in ASP.NET MVC:**

The Content-Returning ActionResults in ASP.NET MVC are responsible for returning content to the browser or calling the script. The examples are as follows:

1. **ViewResult**
2. **PartialViewResult**
3. **FileResult**
4. **ContentResult**
5. **EmptyResult**
6. **JsonResult**
7. **JavaScriptResult**

**Redirection Action Result in ASP.NET MVC:**

The Redirection ActionResults in ASP.NET MVC are responsible for redirecting to other URLs or actions. The examples are as follows:

1. **RedirectResult**
2. **RedirectToRouteResult**
3. **RedirectToActionResult**

**Status Action Result in ASP.NET MVC:**

The Status ActionResults in ASP.NET MVC are responsible for returning status codes to the browser. The examples are as follows:

1. **HttpStatusCodeResult**
2. **HttpUnauthorizedResult**
3. **HttpNotFoundResult**

**What should be the return type of an action method – ActionResult or specific derived type?**

It basically depends on the situation. If your action method returns one type of result, then it is good to use a specific derived type based on the return value. But, if your action method returns different kinds of results based on different conditions, then you should use ActionResult as the return type.  For better understanding, please have a look at the below example. As, the Index method returning two types of Results i.e. View Result and Json Result, so, we are using the return type of the Action method as ActionResult.

**public** ActionResult Index**()**

**{**

**if** **(**Your\_Condition**)**

**return** View**()**; // returns ViewResult object

**else**

**return** Json**(**"Data"**)**; // returns JsonResult object

**}**

In the below example, the action method going to return one type of result i.e. JsonResult, so, it is advisable to use JsonResult as the return type of the Action method.

**public** JsonResult Index**()**

**{**

**return** Json**(**"Data"**)**; // returns JsonResult object

**}**

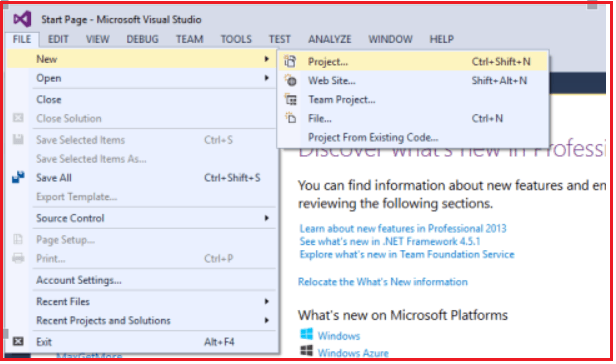
**View Result and Partial View Result in MVC**

**View Result and Partial View Result in ASP.NET MVC**

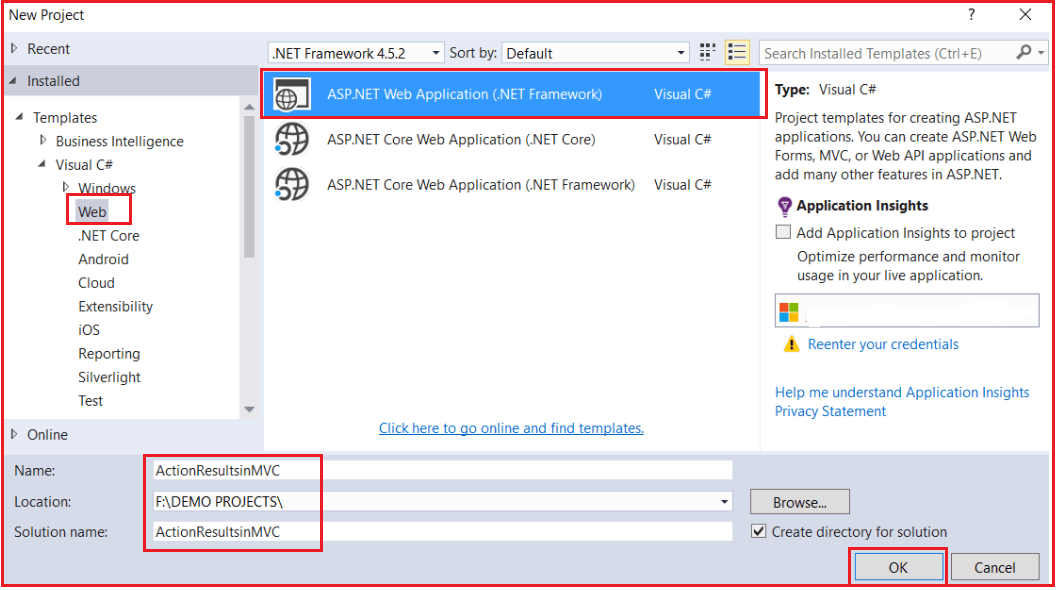
In this article, I am going to discuss the View Result and Partial View Result in the ASP.NET MVC application. Please read our previous article where we discussed the [**Action Results in ASP.NET MVC**](https://dotnettutorials.net/lesson/action-result-overview-mvc/) Application. The ASP.NET MVC Framework provides different types of Action Results. Each action result returns a different format of the output. As a programmer, we need to use different action results to get the expected output.

**Create a New ASP.NET MVC Application**

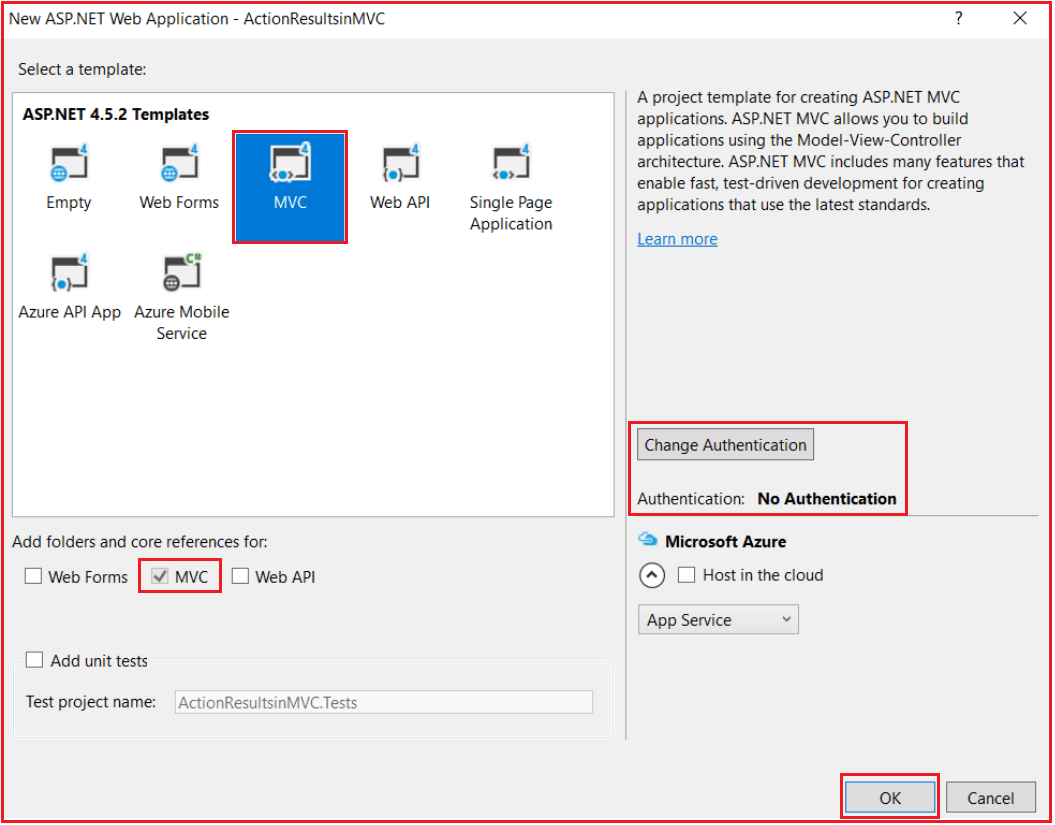
Open Visual Studio and create a new project. To do so, select **File => New => Project**as shown in the below image.



After clicking on the “**Project**“ link a new dialog will pop up. In that we are going to select web templates from the left pane after selecting the web template, select “**ASP.NET Web Application**” and next we are going to name the project “**ActionResultsinMVC**” and clicking on the **OK** button as shown in the below image.



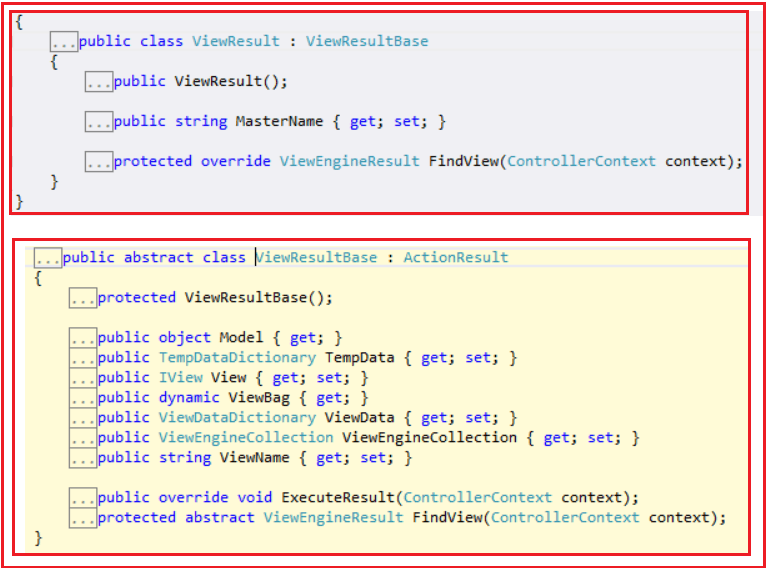
Once you click on the **OK** button a new dialog will pop up with the Name “**New ASP.NET Project**” for selecting project Templates. In this dialog, we are going to choose the MVC project template and then we are going to choose the Authentication type for doing that just click on the Change Authentication button, a new dialog will pop up with the name “**Change Authentication**” here we are going to choose No Authentication click on **OK** Button.



Once you click on the OK button, it will take some time to create a project for us. Let us first understand the need and use of ViewResult and then we will understand the need and use of PartialViewResult in ASP.NET MVC Application.

**View Result in ASP.NET MVC:**

The View Result in MVC is returning the result to a View Page. The View Result can return data to the View Page through the model class. The view page is a simple HTML page. Here view page has a “.cshtml” extension. The ViewResult is a class and is derived from the “ViewResultBase” class. The “ViewResultBase” is derived from ActionResult class as shown in the below image. So, ViewResult is indirectly derived from the ActionResult abstract class. And we already know that ActionResult is the base class of different action results.



View Result class is inherited from the Action Result class via the View Result Base class. The above diagram shown describes the inheritance of Action Results.

**Example: Let’s see the HomeController Index Action method to understand View Result**

**public** **class** HomeController : Controller

**{**

**public** ViewResult Index**()**

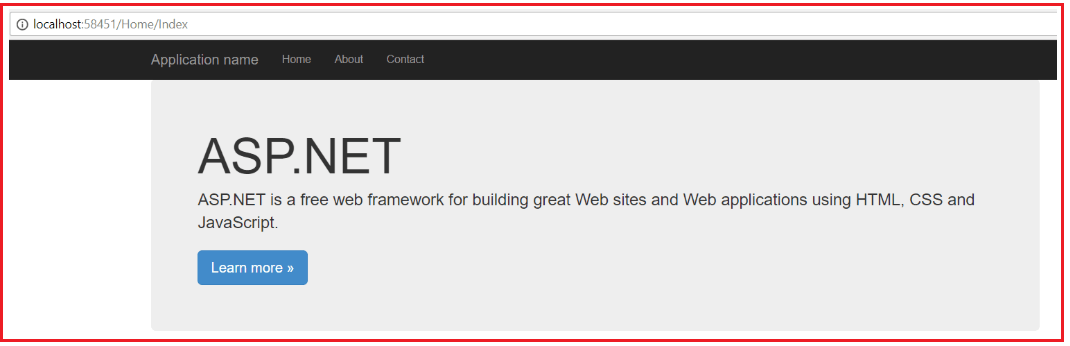
**{**

**return** View**()**;

**}**

**}**

Since ASP.NET MVC follows the convention-over-configuration approach, MVC will look for a View named “**Index**” in the **Views/Home** subfolder, and then look in the **Views/Shared** subfolder and if it doesn’t find it then it will throw an **InvalidOperationException**. So when we navigate to **Home/index** it will display the following page.



**What if I wanted to return a view other than the one that matches the action name? Then we need to explicitly specify the View name as shown below.**

**public** **class** HomeController : Controller

**{**

**public** ViewResult Index**()**

**{**

**return** View**(**"About"**)**;

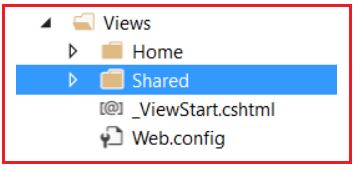
**}**

**}**

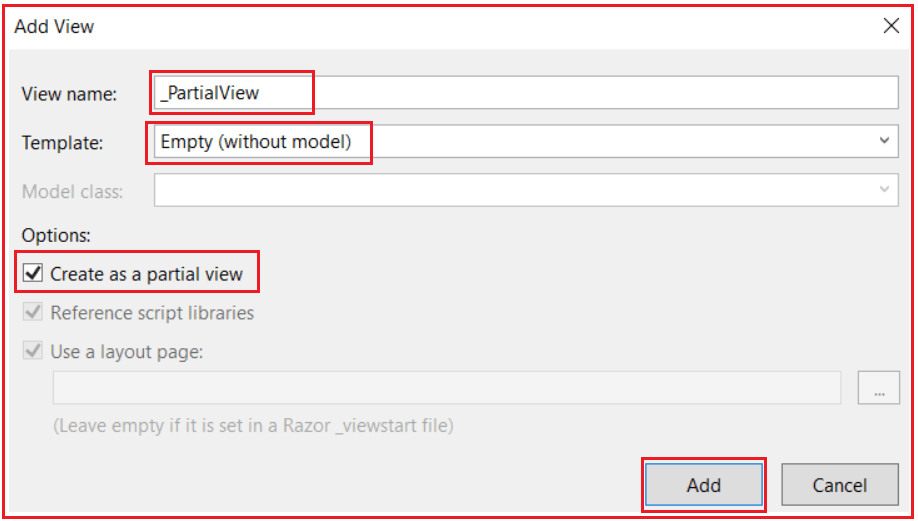
Now it will attempt to find a view with the name “About” in the Views/Home folder and if it is not found there, then it will search the Views/Shared subfolder.

**Partial View Result in ASP.NET MVC Application**

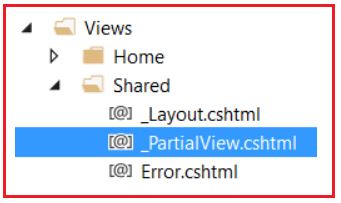
It is also possible to return a Partial View instead of View from an Action Method in the ASP.NET MVC Application. The Partial View Result in MVC is returning the result to a Partial View Page. A partial view is one of the views that we can call inside a Normal view page. First, let’s add a Partial View inside the Shared Folder.



Right-Click on the Shared Folder which is inside the Views folder and then selects **Add => View** option from the context menu which will open the following Add view window.



**Note:** Provide the View name as “**\_PartialView**”, select the Template as **Empty** and then check the Create as a partial view checkbox and click on **Add** button which will add the Partial view in the shared folder as shown in the below image.



Now open **\_PartialView.cshtml** file and then Copy and paste the below code into it.

**<h3>Its a Partial View</h3>**

**Let’s modify the index action method to Return a Partial View as shown below.**

**public** **class** HomeController : Controller

**{**

**public** PartialViewResult Index**()**

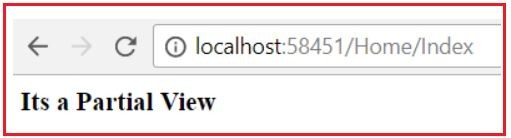
**{**

**return** PartialView**(**"\_PartialView"**)**;

**}**

**}**

Run the application and navigate to Home/Index, it will display the page as shown below



Now it is displaying the content of that partial view but without the layout page. This isn’t very useful by itself, so a more useful application might be to call this action in an AJAX scenario and display the returned view. We will discuss this in a later article.

**Note:** We need to create a Partial view inside the shared folder although it is not mandatory. Here, in this article, I just show you how to use the Partial View Result in ASP.NET MVC Application. In a later article, we will discuss the real-time examples of using [**Partial Views in the ASP.NET MVC**](https://dotnettutorials.net/lesson/partial-views-in-mvc/) Application.

**JSON Result in ASP.NET MVC**

**JSON Result in ASP.NET MVC**

In this article, I am going to discuss the **JSON Result in the ASP.NET MVC** application. Please read our previous article as we are going to work with the same example that we started in [**View Result and Partial View Result in ASP.NET MVC**](https://dotnettutorials.net/lesson/view-result-and-partial-view-result-mvc/) article, ASP.NET MVC has different types of Action Results. Each action result returns a different format of the output. As a programmer, we need to use different action results to get the expected output. Action Results return the result to the view page for the given request.

**JSON Result in ASP.NET MVC**

The JSON result is one of the most important Action results in the ASP.NET MVC application. This action result returns the data in JSON Format i.e. in the form of key-value pairs. And moreover, we need to call this method using Ajax from a view. So, in my point of view, JSON Result is one of the coolest ActionResults. JsonResult is used to represent JSON-encoded data, which is most commonly used to return structured data to a calling script, especially in AJAX scenarios.

But that’s not why I like it so much. In ASP.NET MVC, you can JSONify anything**.**  For example, Let’s modify the Home Controller as shown below

**public** **class** HomeController : Controller

**{**

**[**HttpGet**]**

**public** JsonResult Index**()**

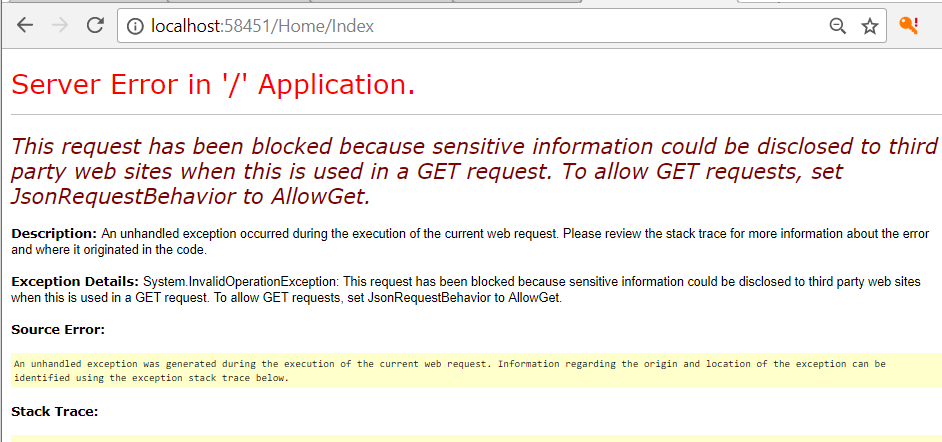
**{**

**return** Json**(**new **{** Name = "John Smith", ID = 4, DateOfBirth = new DateTime**(**1999, 12, 31**)** **})**;

**}**

**}**

Now, run the application and navigates to **Home/Index** in the URL and you will get the following error.



Here, as you can see, MVC Framework throwing an exception saying that “**This request has been blocked because sensitive information could be disclosed to third party websites when this is used in a GET request.**“

MVC is trying to protect you here; it doesn’t want you to share JSON information over a GET request because it could potentially contain sensitive information. This is attempting to protect you from an exploit known as JSON Hijacking. You can turn off this error by modifying the Index method as shown below.

**public** **class** HomeController : Controller

**{**

**[**HttpGet**]**

**public** JsonResult Index**()**

**{**

**return** Json**(**new **{** Name = "John Smith", ID = 4, DateOfBirth = new DateTime**(**1999, 12, 31**)** **}**,

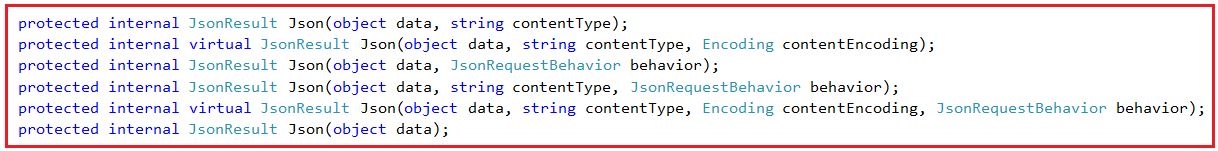
JsonRequestBehavior.AllowGet**)**;

**}**

**}**

**OUTPUT: {“Name”:”John Smith”,”ID”:4,”DateOfBirth”:”\/Date(946578600000)\/”}**

**JsonRequestBehavior.AllowGet** does exactly what it sounds like; it allows the browsers to access JSON information in a GET request. I would only recommend turning this on if you know what you are doing since it could potentially expose you to JSON Hijacking. There are six overloaded versions available for this JSON as shown in the below image.



**Parameters:**

1. **data**: The JavaScript object graph to serialize.
2. **contentType**: The content type (MIME type).
3. **contentEncoding**: The content-encoding.
4. **behavior**: The JSON request behavior.

**JSON with Complex Type in ASP.NET MVC Application:**

First Create a Model in the Models folder with the name **Person.cs** and then copy and paste the following code into it.

**public** **class** Person

**{**

**public** **int** Id **{** **get**; **set**; **}**

**public** string FirstName **{** **get**; **set**; **}**

**public** string LastName **{** **get**; **set**; **}**

**}**

**Modify Home Controller:**

Copy and paste the following code in Home Controller

**public** **class** HomeController : Controller

**{**

**[**HttpGet**]**

**public** ActionResult Index**()**

**{**

var persons = new List**<**Person**>**

**{**

new Person**{**Id=1, FirstName="Harry", LastName="Potter"**}**,

new Person**{**Id=2, FirstName="James", LastName="Raj"**}**

**}**;

**return** Json**(**persons, "application/json", System.Text.Encoding.UTF8 ,JsonRequestBehavior.AllowGet**)**;

**}**

**}**

**Output: [{“Id”:1,”FirstName”:”Harry”,”LastName”:”Potter”},{“Id”:2,”FirstName”:”James”,”LastName”:”Raj”}]**

While returning more data in JSON format, there is a need to mention the maximum length. Assign the maximum length of data Using “MaxJsonLength” property as shown in the below code.

**public** **class** HomeController : Controller

**{**

**[**HttpGet**]**

**public** ActionResult Index**()**

**{**

var persons = new List**<**Person**>**

**{**

new Person**{**Id=1, FirstName="Harry", LastName="Potter"**}**,

new Person**{**Id=2, FirstName="James", LastName="Raj"**}**

**}**;

var jsonResult = Json**(**persons, "application/json", System.Text.Encoding.UTF8 ,JsonRequestBehavior.AllowGet**)**;

jsonResult.MaxJsonLength = **int**.MaxValue;

**return** jsonResult;

**}**

**}**

**Redirect, RedirectToRoute and RedirectToAction in ASP.NET MVC**

**Redirect, RedirectToRoute and RedirectToAction in ASP.NET MVC**

In this article, I am going to discuss **Redirect, RedirectToRoute, and RedirectToAction** in the ASP.NET MVC Application. The ASP.NET MVC has different types of Action Results. Each action result returns a different format of the output. As a programmer, we need to use different action results to get the expected output. Action Results return the result to view the page for the given request.

Note: We are going to work with the same example that we started in [**View Result and Partial View Result in ASP.NET MVC**](https://dotnettutorials.net/lesson/view-result-and-partial-view-result-mvc/) article and continue in [**File Result, Content Result, Empty Result, JavaScript Result, and JSON Result in MVC**](https://dotnettutorials.net/lesson/file-result-content-result-empty-result-javascript-result-json-result-mvc/) article of this MVC article series.

**Redirect Result in ASP.NET MVC**

Suppose, you want to redirect to a specific URL, then you need to use the Redirect method and this method takes the URL to recirect. For example, suppose, we want to redirect to the **URL: https://dotnettutorials.net**, then we need to use the Redirect method as shown in the below code.

**public** **class** HomeController : Controller

**{**

**public** RedirectResult Index**()**

**{**

**return** Redirect**(**"https://dotnettutorials.net"**)**;

**}**

**}**

This works great for redirecting to outside sites from the current application, but not for redirecting to other pages within the same application. For that, we can use RedirectToRouteResult. Redirect result is returning the result to a specific URL. It is rendered to the page by URL. If we give the wrong URL, it will show 404-page errors.

**RedirectToRoute Result in ASP.NET MVC**

The RedirectToRouteResult is used whenever we need to go from one action method to another action method within the same or different controller in ASP.NET MVC Application. For example, in the below code, we are redirecting to Home Controller, About action method from the Index action method of Home Controller.

**public** **class** HomeController : Controller

**{**

**public** RedirectToRouteResult Index**()**

**{**

**return** RedirectToRoute**(**new **{** controller = "Home", action = "About" **})**;

**}**

**}**

That’s not very friendly though. There’s a better way, an overload of this helper called RedirectToAction:

**RedirectToAction Result in ASP.NET MVC**

The RedirectToAction Result in ASP.NET MVC is returning the result to a specified controller and action method. Controller name is optional in RedirectToAction method. If not mentioned, the Controller name redirects to a mentioned action method in the current Controller. Suppose the action name is not available but mentioned in the current controller, then it will show a 404 error page.

**public** **class** HomeController : Controller

**{**

**public** ActionResult Index**()**

**{**

**return** RedirectToAction**(**"Login", "Account"**)**;

**}**

**}**

# Status Results in ASP.NET MVC

## ****Status Results in ASP.NET MVC Application****

In this article, I am going to discuss **Status Results in ASP.NET MVC** Application. Please read our previous article, where we discussed [**Redirect, RedirectToRoute, and RedirectToAction in the ASP.NET MVC**](https://dotnettutorials.net/lesson/redirect-redirecttoaction-mvc/) Application. ASP.NET MVC has different types of Action Results. Each action result returns a different format of the output. As a programmer, we need to use different action results to get the expected output. Action Results return the result to view the page for the given request. Please read the below articles before proceeding to this article.

We are going to work with the same example that we worked on in our previous 4 articles. The Status Action Results will return status codes to the browser for it to use. As part of the Status Result, we ate going to discuss the following three things.

1. **HttpStatusCodeResult**
2. **HttpUnauthorizedResult**
3. **HttpNotFoundResult**

##### ****HttpStatusCodeResult in ASP.NET MVC:****

HttpStatusCodeResult in ASP.NET MVC Framework returns an HTTP status code to the browser, along with a custom message to be displayed. Let’s modify the Home Controller to understand this concept in ASP.NET MVC.

**using** *System.Net;*

**using** *System.Web.Mvc;*

**public** **class** HomeController : Controller

**{**

**public** HttpStatusCodeResult UnauthorizedStatusCode**()**

**{**

**return** new HttpStatusCodeResult**(**HttpStatusCode.Unauthorized, "You are not authorized to access this controller action."**)**;

**}**

**public** HttpStatusCodeResult BadGateway**()**

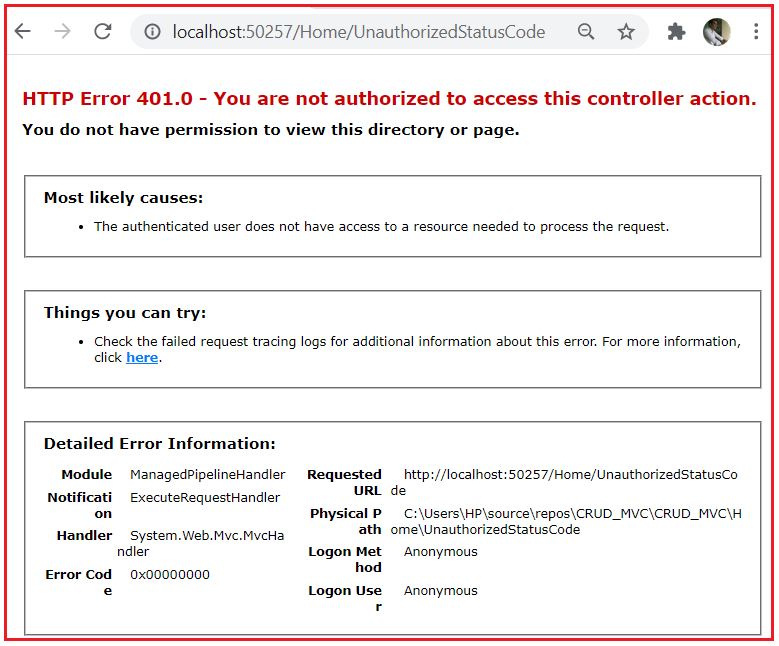
**{**

**return** new HttpStatusCodeResult**(**HttpStatusCode.BadGateway, "I have no idea what this error means."**)**;

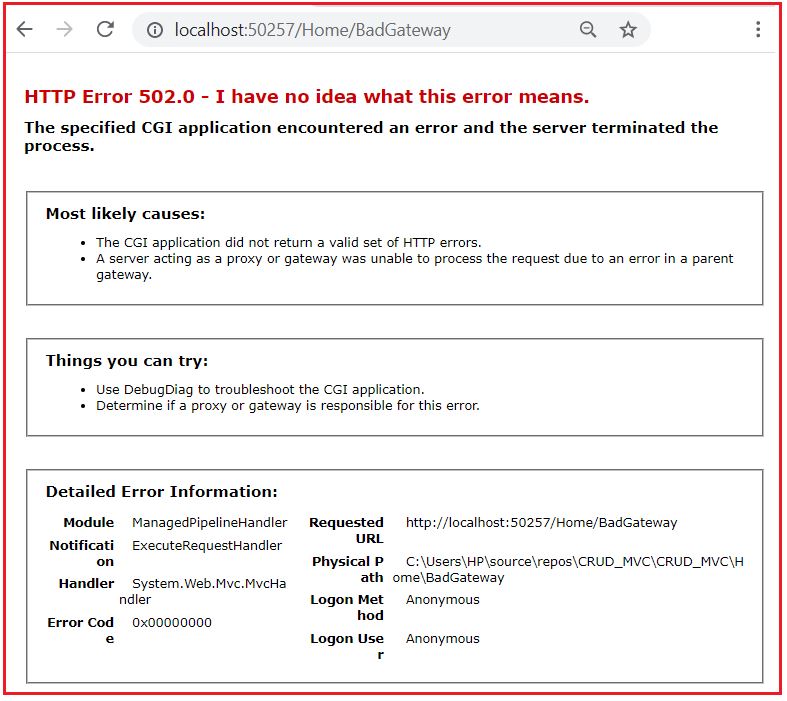
**}**

**}**

There is no helper method for this ActionResult. The HttpStatusCode enumeration contains all HTTP status codes (so that you don’t have to remember what 402 or 307 mean). These are useful in exception-driven scenarios where you have custom error pages defined. So when we navigate to **Home/UnauthorizedStatusCode**, it will display the following error page.



Similarly, when we navigate to **Home/BadGateway**, it will display the following error page.



##### ****HttpUnauthorizedResult in ASP.NET MVC****

Returning an HttpUnauthorizedResult is the same as returning HttpStatusCodeResult with HttpStatusCode.Unauthorized, it’s just more readable:

**using** *System.Web.Mvc;*

**public** **class** HomeController : Controller

**{**

**public** HttpStatusCodeResult UnauthorizedResult**()**

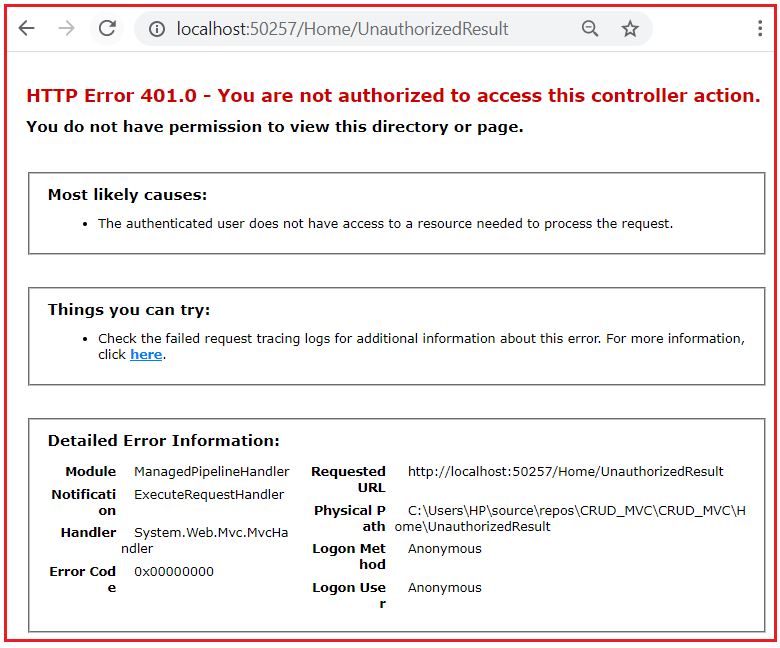
**{**

**return** new HttpUnauthorizedResult**(**"You are not authorized to access this controller action."**)**;

**}**

**}**

When we navigate to **Home/UnauthorizedResult**, then it will display the following error page



#### ****HttpNotFoundResult in ASP.NET MVC****

This is also an overload of HttpStatusCodeResult, but unlike HttpUnauthorizedResult, it actually does have a helper method:

**using** *System.Web.Mvc;*

**public** **class** HomeController : Controller

**{**

**public** HttpNotFoundResult NotFound**()**

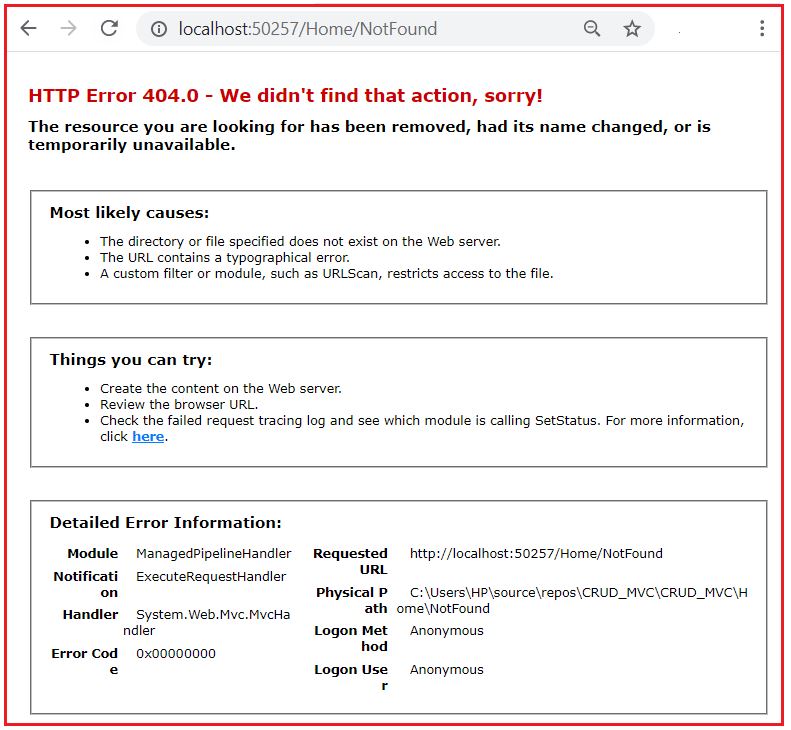
**{**

**return** HttpNotFound**(**"We didn't find that action, sorry!"**)**;

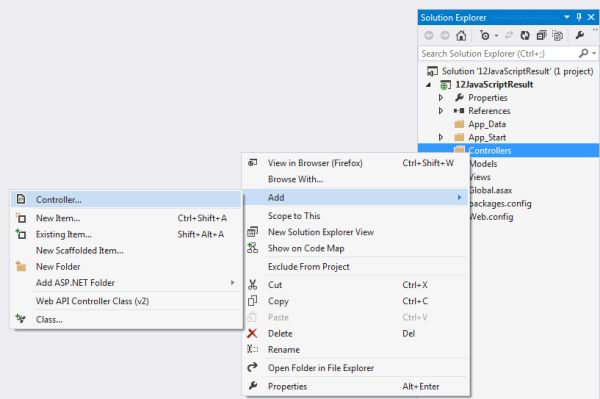
**}**

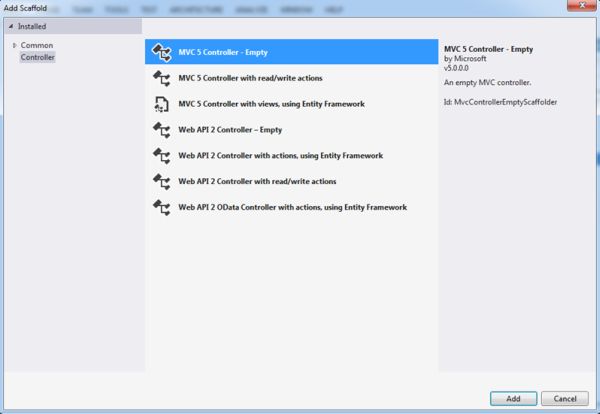
**}**

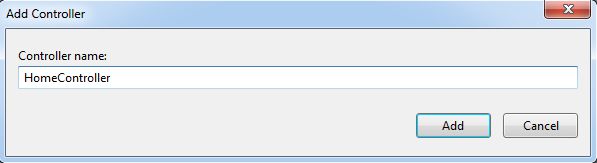
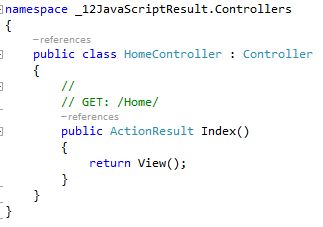
When we navigate to **Home/NotFound**, then it will display the following error page.

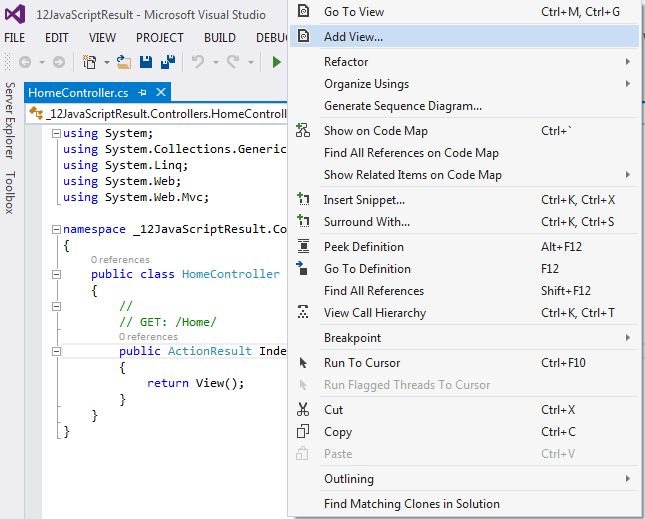


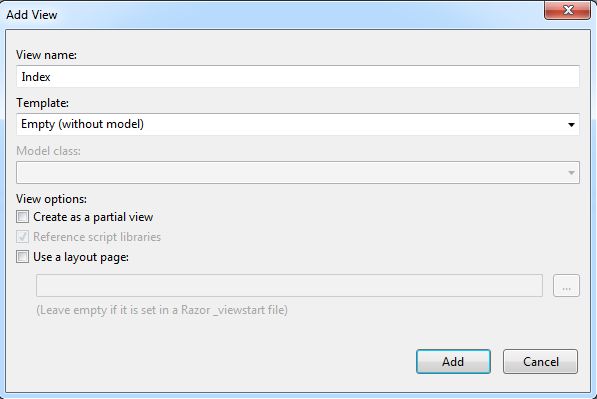
**JavaScript result in controller in MVC.**  
**Step 1**

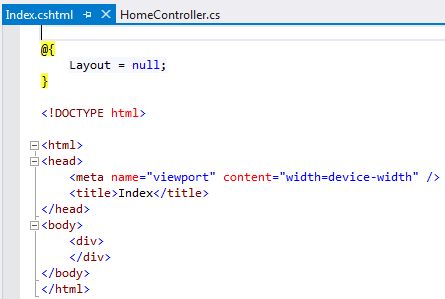
Create a MVC project from the "Empty" template.  
  
Right-click on "Controllers" and select "Add" >> "Controller...".  
  
  
  
**Step 2**

Select "MVC 5 Controller - Empty" to add an empty controller.  
  
Click on the "Add" button.  
  
  
  
**Step 3**

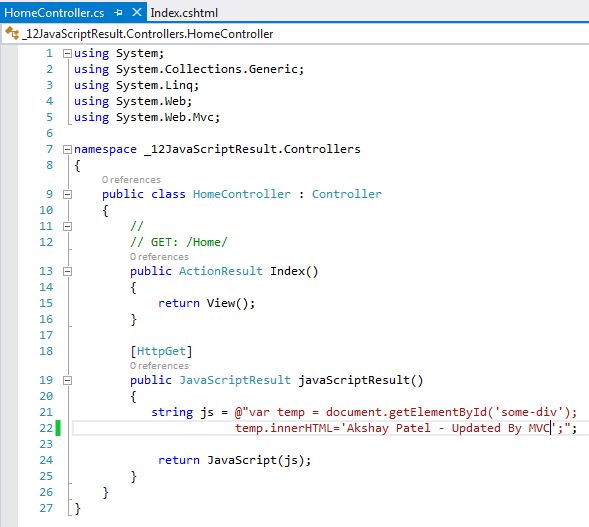
Name the controller as in the following:  
  
  
  
And it will add an action result method to the controller.  
  
  
  
**Step 4**

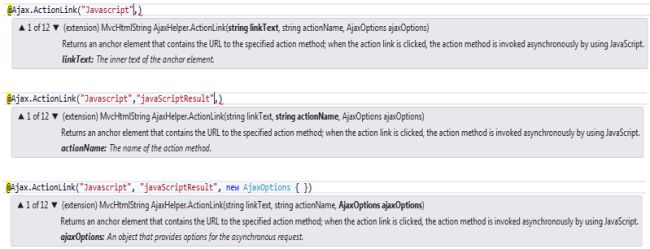
Now we need to create a view.  
  
Right-click on "Index" and select "Add View...".  
  
  
  
**Step 5**

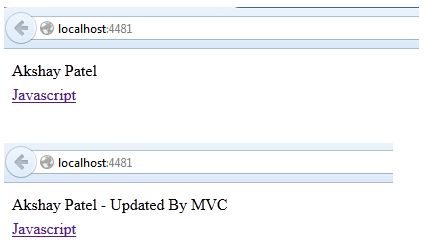
Name the view and select "Empty (without model)" as the template.  
  
Click on the "Add" button.  
  
  
  
**Step 6**

Add a title to the index page.  
  


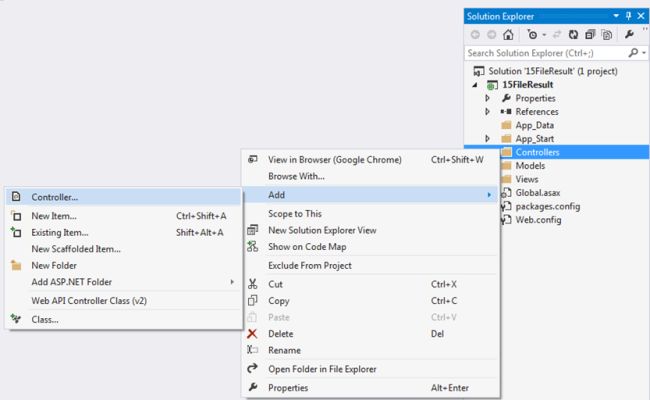
**Step 7**

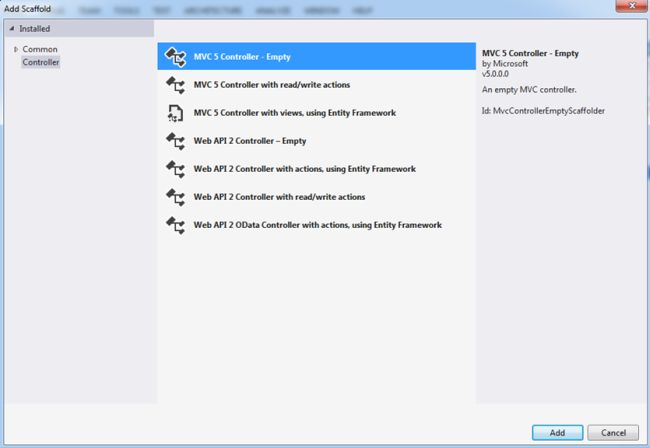
JavaScript result sends JavaScript content to the response. Here we create one div element in the index.cshtml page. We write some text inside the div element. In the JavaScript result method we get a div element and update its content using JavaScript.  
  
  
  
**Step 8**

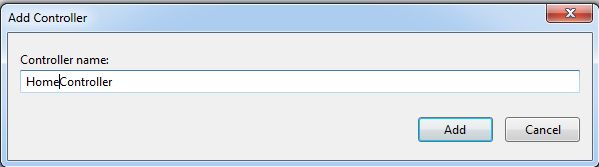
Now let's create a link to call the javaScriptResult method of the Home controller. Here we use ajaxhelper to create a link. Ajaxhelper methods are called using "Ajax" property of the view. Using Ajaxhelper we can "submit a form" and "invoke an action method". So here we use Ajaxhelper to invoke the javascriptresult method using Ajax.ActionLink.  
  
  
  
We use "GET" for httpmethod in ajaxoptions. Our updatetargetid is "some-div" since we want to access the content of that div. And we set the insertion mode to "replace" since we need to replace the content inside the div.  
  
  
  
**Step 9**

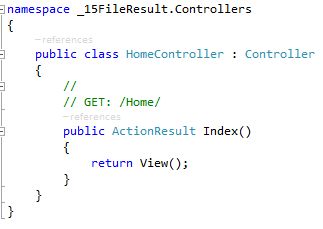
Run the project. The "Akshay Patel" string is rendered on the browser. Now click on the JavaScript link and it will update the rendered string to "Akshay Patel - Updated By MVC".  
  


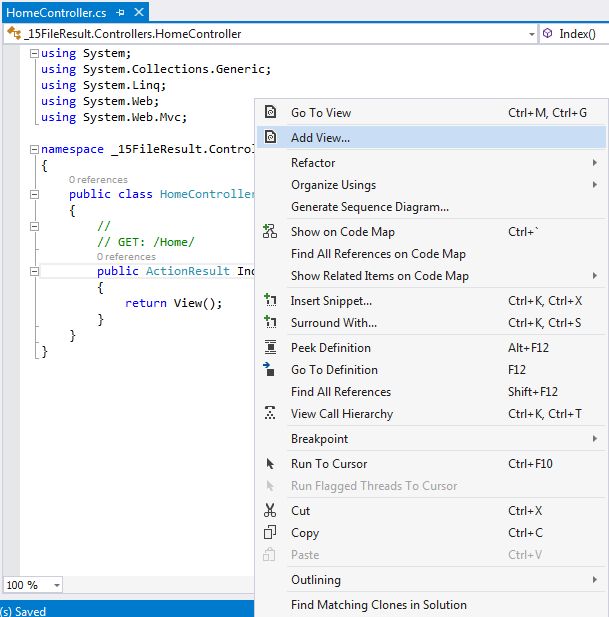
**File result in a controller in MVC.**  
**Step 1**  
Create a MVC project from the "Empty" template. Right-click on "Controllers" and select "Add" >> "Controller...".

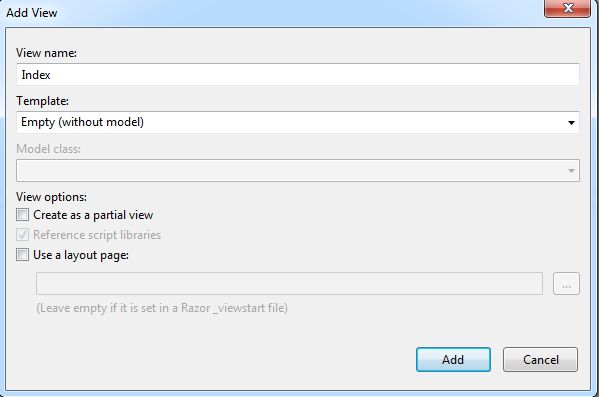
  
  
**Step 2**  
Select "MVC 5 Controller - Empty" to add an empty controller. Click on the "Add" button.

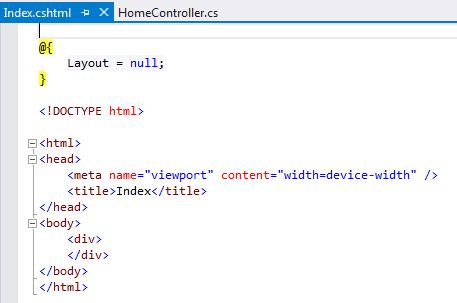
  
  
**Step 3**  
Name the controller as in the following:

  
  
And it will add an action result method to the controller.

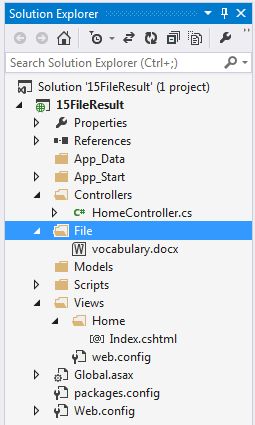
  
  
**Step 4**  
Now we need to create a view. Right-click on "Index" and select "Add View...".

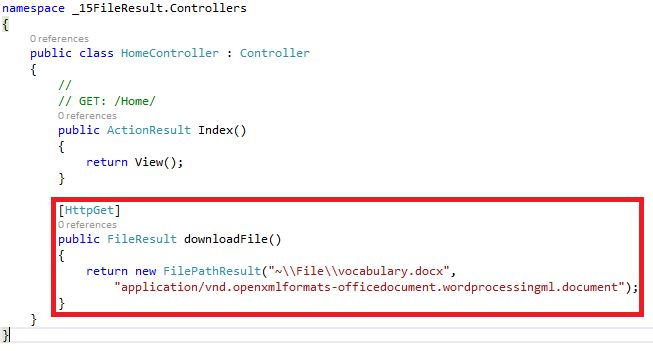
  
  
**Step 5**  
Name the view and select "Empty (without model)" as the template. Click on the "Add" button.

  
  
**Step 6**  
Add a title to the index page.

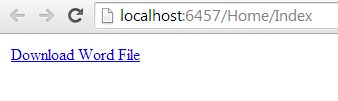
  
  
**Step 7**

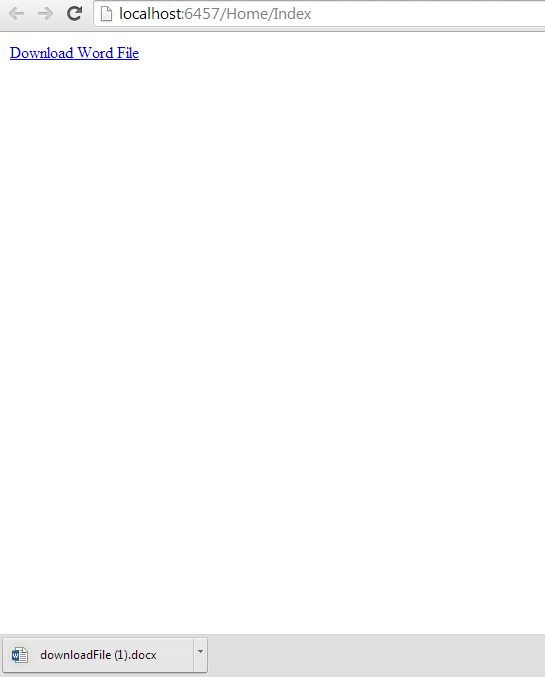
Create a folder "File" and save a Word document to test the FileResult sample.

  
  
**Step 8**  
Filepathresult initializes a new instance of the filepathresult class by using the specified file name and content type.

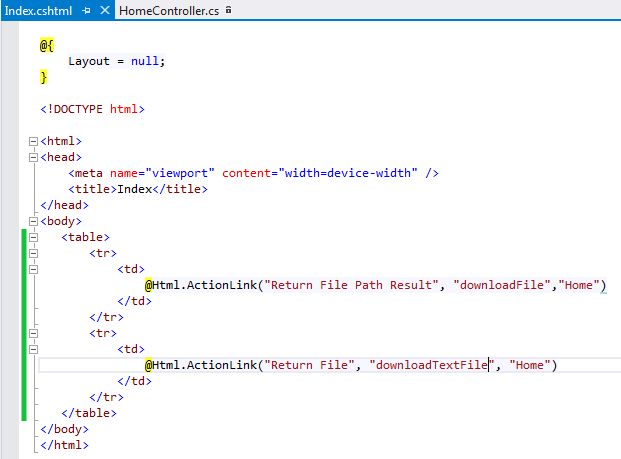
  
  
**Step 9**  
Create a link to call the downloadfile() method of the home controller.

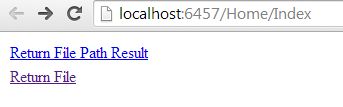
  
  
**Step 10**  
Run the project, click on "Download Word File" and it will download the Word file. You can find the file in the download folder.

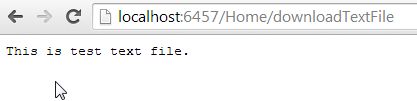


  
  
**Step 11**  
The file creates a filepathresult object using file name and content type. So create another action method and return the file.

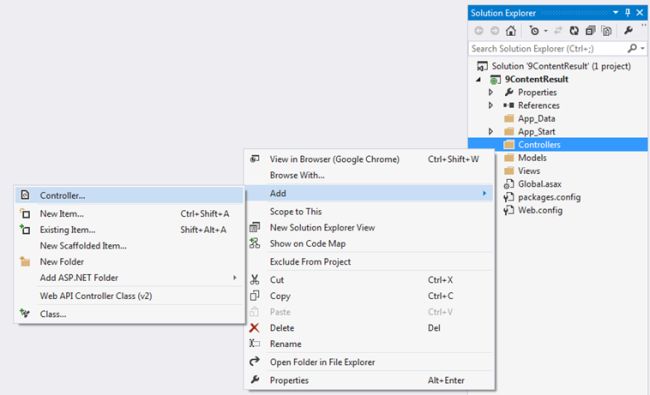
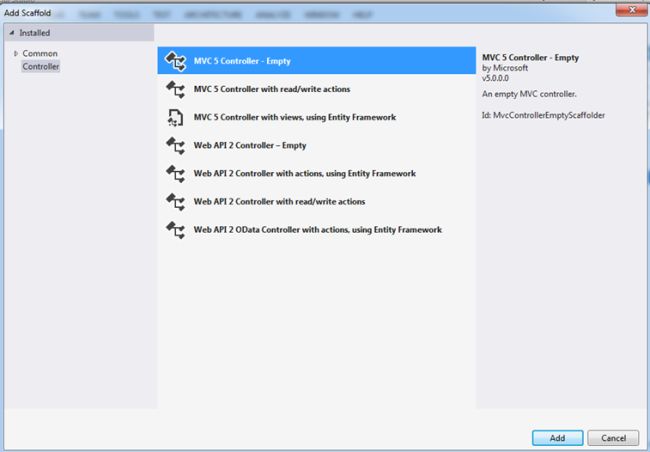
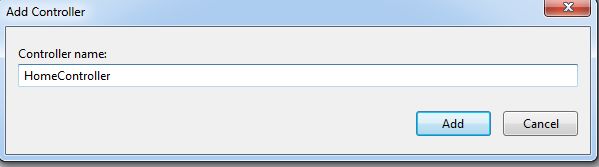
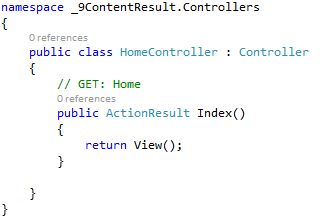
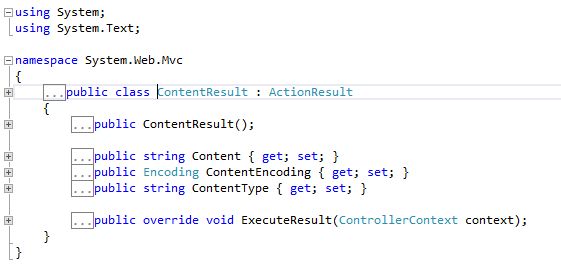
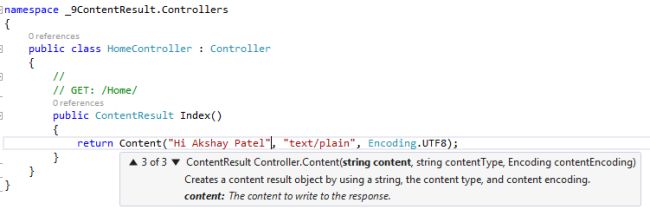
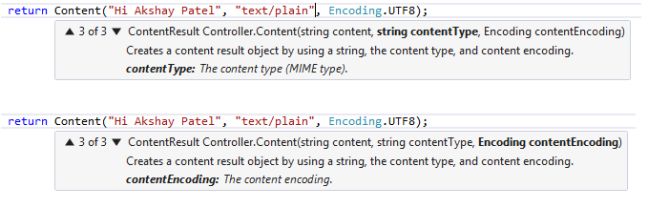
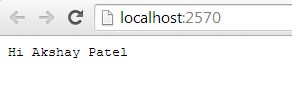
  
  
**Step 12**  
Create a link to call the downloadTextFile() method of the home controller.

  
  
**Step 13**  
Run the project, click on "Return File" and the text file will be opened in the browser.





**Content Result in Controller Sample in MVC**

**Step 1**  
  
Create a MVC project from the "Empty" template.  
  
Right-click on "Controllers" and select "Add" >> "Controller...".  
  
  
  
**Step 2**  
  
Select "MVC 5 Controller - Empty" to add an empty controller.  
  
Click on the "Add" button.  
  
  
  
**Step 3**  
  
Name the controller as in the following:  
  
  
  
And it will add an action result method in the controller.  
  
  
  
**Step 4**  
ContentResult represents a user-defined content type. It is inherited from ActionResult. ContentResult has the following three properties:  
  
Content that we want to render on browser  
ContentEncoding that defines the encoding of the content  
ContentType that informs the browser about the content like "text/plain", "text/html", "application/json" and so on.  
  
  
  
Here the ContentResult method returns an object of ContentResult. In the first parameter we pass the content that we want to render on the browser. The second parameter is of ContentType and the third one is content encoding.  
  
  
  
  
**Step 5**  
  
Run the project and you will see, content is rendered in the browser  
  


**EmptyResult Return Type In ASP.NET MVC 5**

MVC controller returns many types of output to the view according to the data we need for the application. In this article we will learn about EmptyResult return type of MVC. So instead of going into the depth on the subject, let us start with its practical implementation.

To know more about the Action result types please refer my previous article,

* [ActionResults in ASP.Net MVC](http://www.compilemode.com/2015/06/action-result-in-asp-net-mvc.html)
* [JsonResult Type in MVC](http://www.compilemode.com/2015/10/json-result-type-in-mvc.html)
* [PartialViewResult Return Type In MVC](http://www.compilemode.com/2015/11/partialviewresult-return-type-in-mvc.html)

**What is ActionResult?**

It is one of the type of output format in ASP.NET MVC which is shown to the client.

**What is EmptyResult?**

The EmptyResult is a class in MVC which does not return anything at client site, its just like Void method .  
  
EmptyResult is used when you want to execute logic return inside the controller action method but does not want any result back to the view then EmptyResult return type is very important .  
  
 **Key points**

* It does not return any output to the browser.
* It shows the empty result set to the browser without adding the view.
* Does not require to add the view.

**Methods of EmptyResult**

The following are the methods of EmptyResult class:

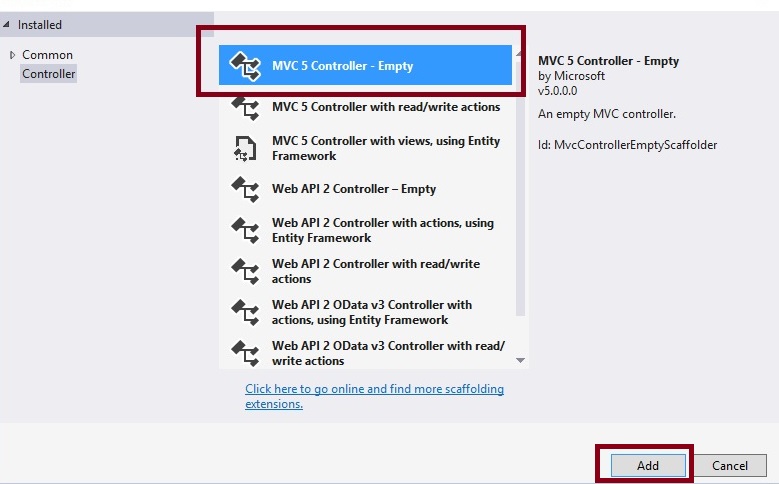
* **Equals**: This method is used to check whether the two objects are equal or not.
* **ExecuteResult**: This method is used to execute the specific result context.
* **Finalize**: This method is used to free the memory which is occupied by object and allow to allocate another object in freed memory .
* **GetHashCode**: It is used to get the numeric value which is used to identify and insert an object in hash based collection.
* **GetType**: This method is used to check the what is the type of object .
* **MemberwiseClone**: This method is used to create the shallow copy of the current object .
* **ToString**: This method is used to convert the current result to the string .

I hope you have understand about the EmptyResult type from preceding brief summary, now let's implement it practically.

**Step 1:**[Create an MVC application](http://www.compilemode.com/2015/09/how-to-create-aspnet-mvc-application.html)

1. "*Start*", then "All *Programs*" and select "Microsoft Visual Studio 2015".
2. "*File*", then "*New*" and click "*Project*" then select "*ASP.NET Web Application Template*", then provide the Project a name as you wish and click on *OK.*
3. Choose MVC empty application option and click on OK

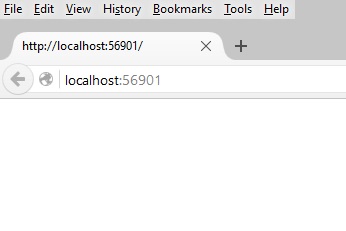
**Step 3**: Add controller class,  
  
Add user and admin controller controller. Right click on Controller folder in the created MVC application and add the controller class as.

  
  
**HomeController.cs**

1. **public** **class** HomeController : Controller
2. {
3. // GET: for main view
4. **public** EmptyResult EmptyData()
5. {
6. **return** **new** EmptyResult();
7. }
9. }

In the above controller class we have added action method that is EmptyData which returns EmptyResult means nothing.

Lets run the application and see the output.



From above examples we have learned about the EmptyResult return type and its use.

**Note**

* Apply proper validation before using it in your project.