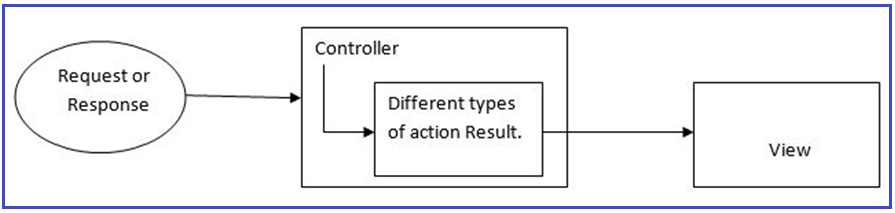
**Action Results in ASP.NET MVC**

In this article, I am going to give an overview of the **Action Results in the ASP.NET MVC**Application. In ASP.NET MVC Application, there are many 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. So, at the end of this article, you will understand the different types of Action Results and when to use which action results in MVC Application.



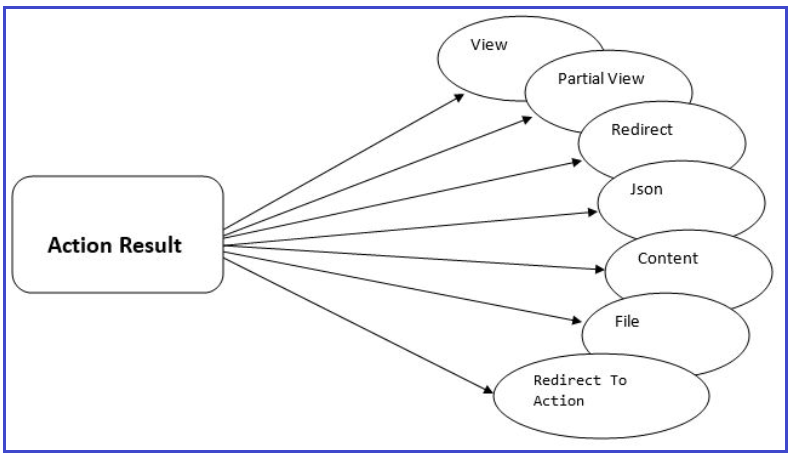
**What is the Action Method in ASP.NET MVC?**

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.

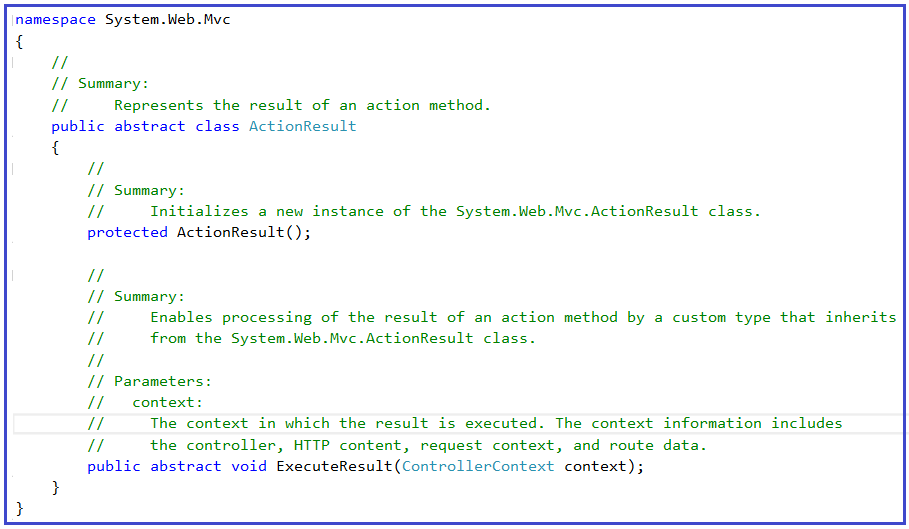
**What is the Action Result in ASP.NET MVC?**

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**()**

**{**

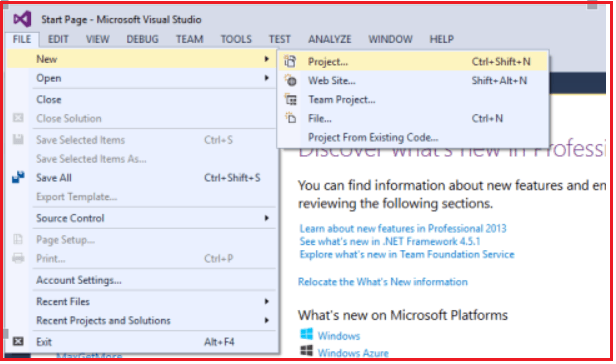
**return** Json**(**"Data"**)**; // returns JsonResult object

**}**

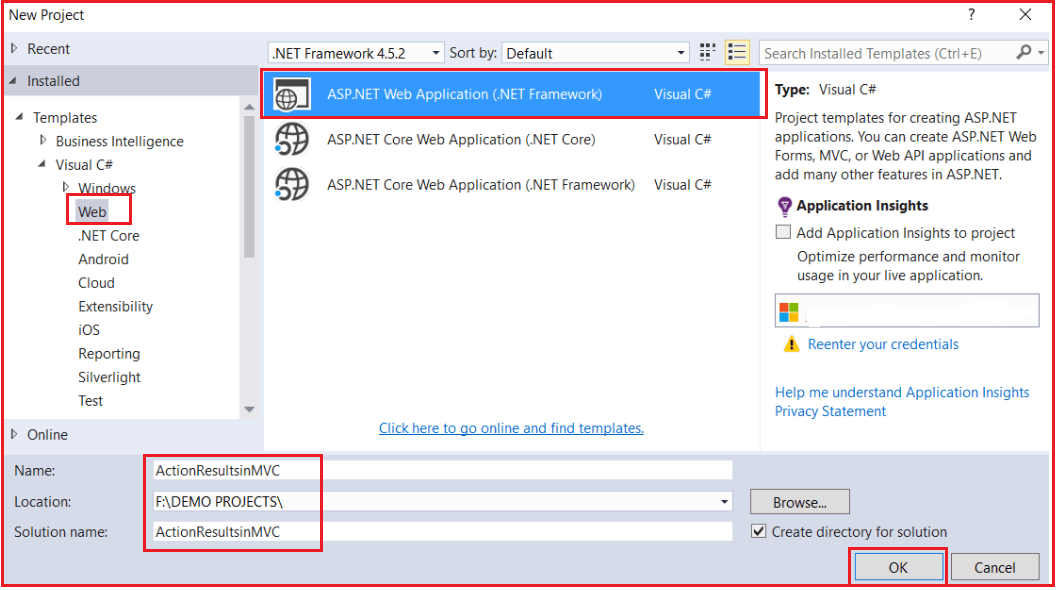
## ****View Result and Partial View Result in ASP.NET MVC****

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

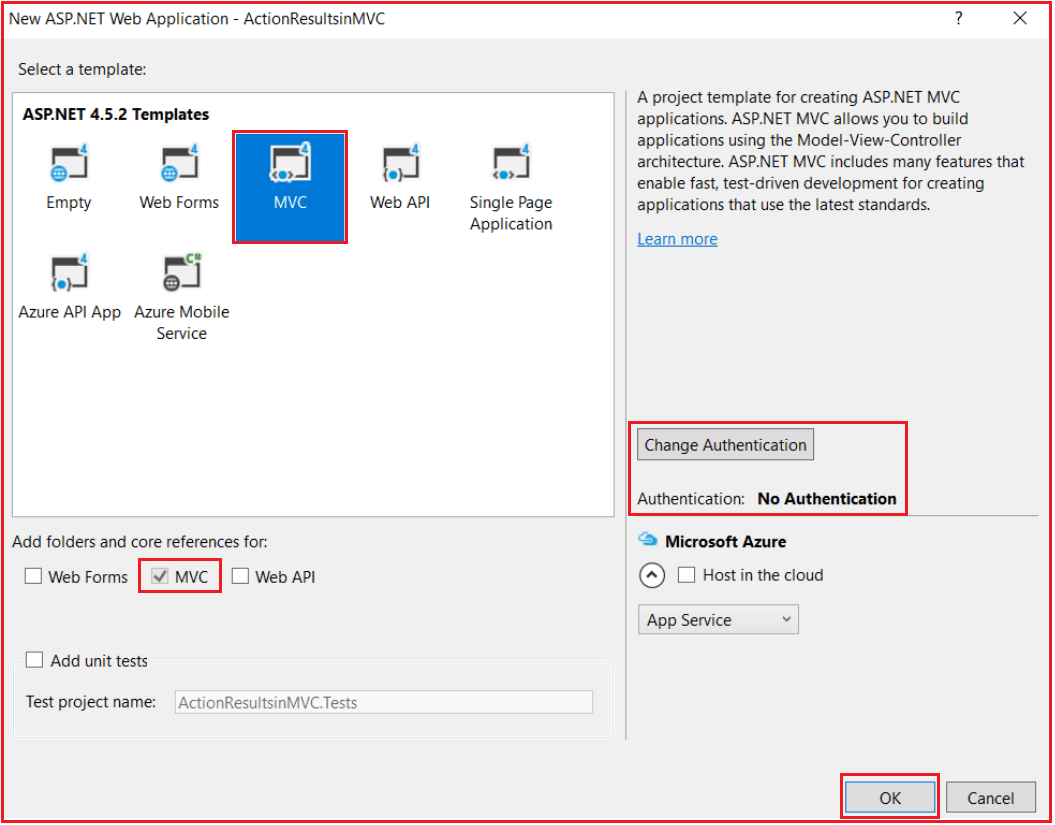
Open Visual Studio and create a new project. To do so, select **File => New => Project**as shown in the below image.



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.



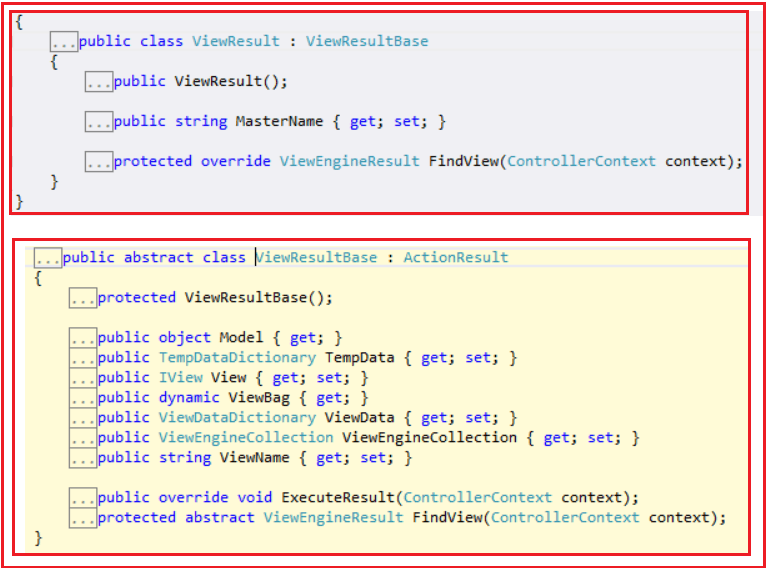
Once you click on the **OK** button a new dialog will pop up with the Name “**New ASP.NET Project**” for selecting project Templates. In this dialog, we are going to choose the MVC project template and then we are going to choose the Authentication type for doing that just click on the Change Authentication 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**()**

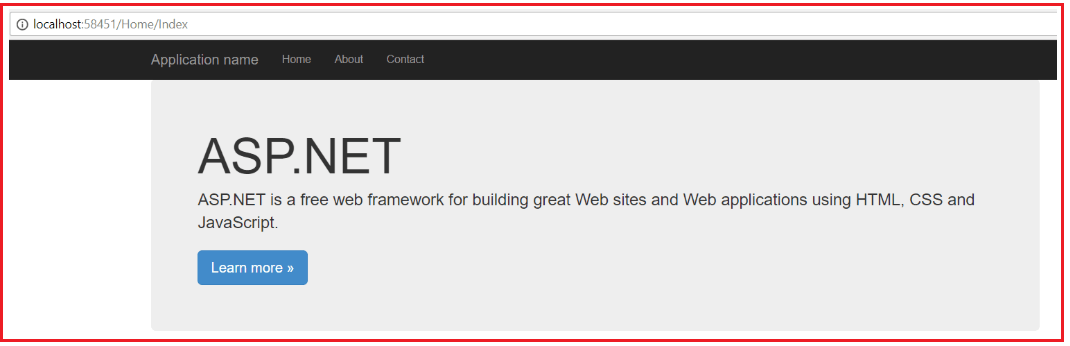
**{**

**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"**)**;

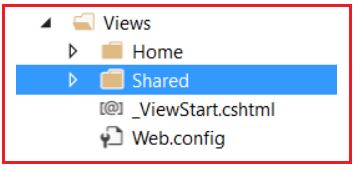
**}**

**}**

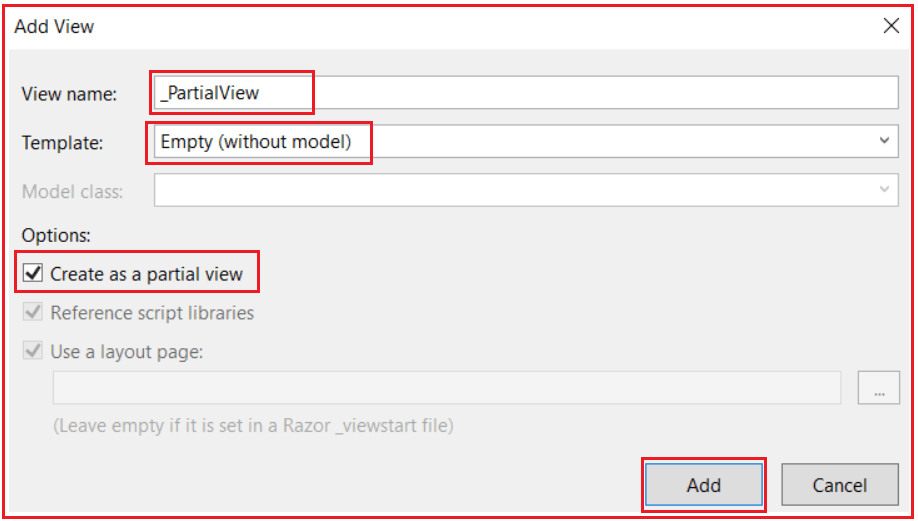
Now it will attempt to find a view with the name “About” in the Views/Home folder and if it is not found there, then it will search the Views/Shared subfolder.

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

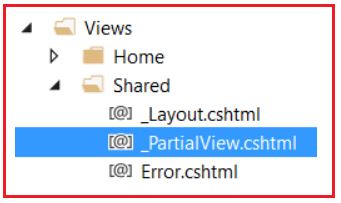
It is also possible to return a Partial View instead of View from an Action Method in the ASP.NET MVC Application. The Partial View Result in MVC is returning the result to a Partial View Page. A partial view is one of the views that we can call inside a Normal view page. First, let’s add a Partial View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

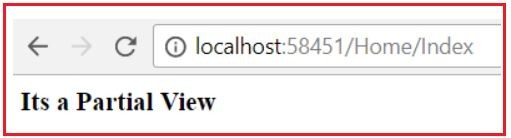
**{**

**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.