File Result in ASP.NET MVC

Back to: ASP.NET MVC Tutorial For Beginners and Professionals

File Result in ASP.NET MVC

In this article, I am going to discuss **File Result in the ASP.NET MVC** application. Please read our previous article where we discussed **JavaScriptResult in ASP.NET MVC**. 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.

File Result in ASP.NET MVC:

The FileResult actions are used to read and write files. FileResult is the parent of all file-related action results. The File Result returns different types of file formats. We implement the file download concept in ASP.NET MVC using file results. Let us understand the File Result with an example. Please modify the HomeController as shown below.

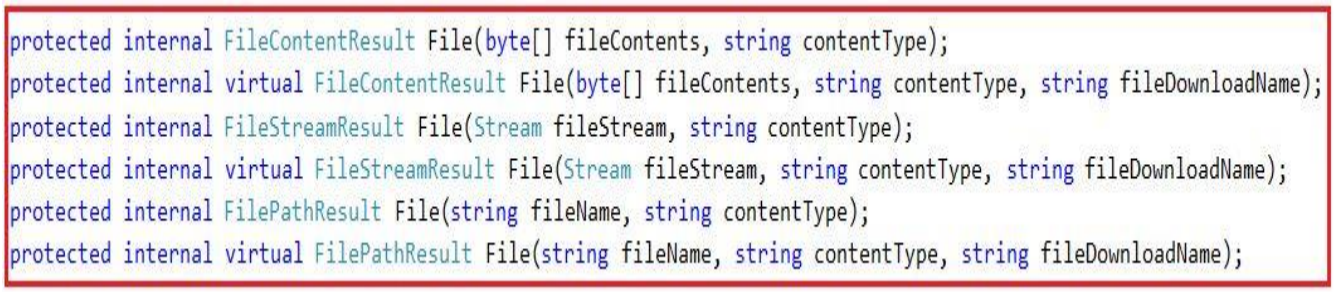
```
public class HomeController : Controller
{
public FileResult Index()
{
return File("Web.Config", "text");
}
}
```

As you can see in the above code, we are using the following overloaded version of the FileResult method. This method takes two parameters, the first parameter is the file name and the second parameter is the content type i.e. MIME type. Here, in our example, we are passing the file name web.config and the content type as Text. So, when you run the above, then you will see that, it will display the content of the web.config file in the browser.

protected internal FilePathResult File(string fileName, string contentType);

If you want to return a file from your action method then you need to use either FileResult or ActionResult as the return type of your action method. Moreover, depending on the overload version of the File you use, you can specify what action the browser is going to take with the

downloaded file. There are six overloaded versions of the File methods are available in ASP.NET MVC Framework which are shown in the below image.



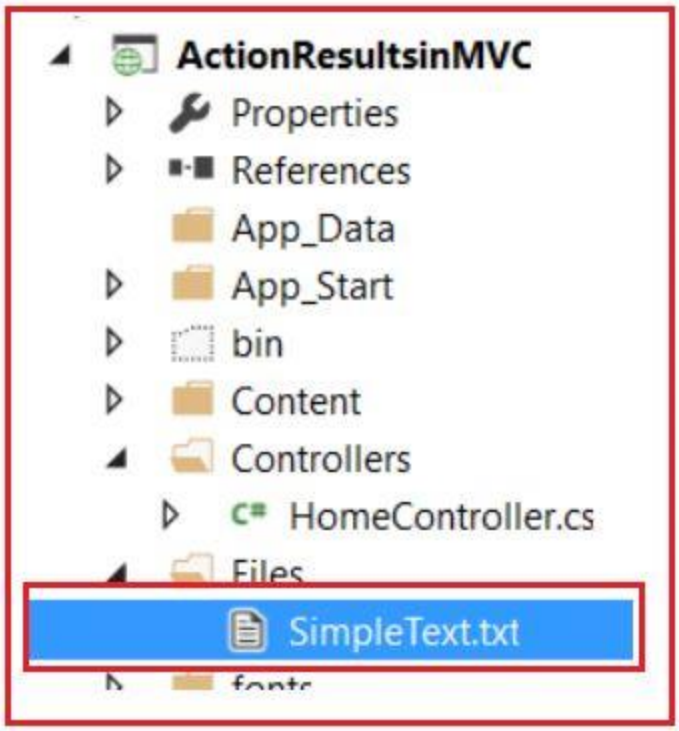
```
protected internal FileContentResult File(byte[] fileContents, string contentType);
protected internal virtual FileContentResult File(byte[] fileContents, string contentType, string fileDownloadName);
protected internal FileStreamResult File(Stream fileStream, string contentType);
protected internal virtual FileStreamResult File(Stream fileStream, string contentType, string fileDownloadName);
protected internal FilePathResult File(string fileName, string contentType);
protected internal virtual FilePathResult File(string fileName, string contentType, string fileDownloadName);
```

Parameters:

1. **fileName**: The path of the file to send to the response.
2. **contentType**: The content type (MIME type).
3. **fileStream**: The stream to send to the response.
4. **fileContents**: The binary content to send to the response.
5. **fileDownloadName**: The filename to use in the file-download dialog box that is displayed in the browser.

Example to Understand File Result in ASP.NET MVC Application:

For example, if you just specify a **URL** and a **MIME (Multipurpose Internet Mail Extensions)** type, then the browser attempts to display the file specified (the supported browsers are Firefox, Chrome, and IE10). Let's create a folder called "**Files**" and a text file to this folder with the name "**SimpleText.txt**" as shown below.



Within the SimpleText.txt file write down the following

This is a Text file

Next, modify the Home Controller to use File Result as shown below

```
public class HomeController : Controller
{
public FileResult Index()
{
return File(Url.Content("~/Files/SimpleText.txt"), "text/plain");
}
}
```

We can also just return the byte array for the file content, which (in Chrome at least)

will display the file in the browser:

```
public class HomeController : Controller
{
public FileResult Index()
{
byte[] fileBytes = System.IO.File.ReadAllBytes(Server.MapPath("~/Files/SimpleText.txt"));
return File(fileBytes, "text/plain");
}
}
```

Another overload will specify a download name, and using this overload causes the browser to download the file, rather than just display it:

```
public class HomeController : Controller
{
public FileResult Index()
{
return File(Url.Content("~/Files/SimpleText.txt"), "text/plain", "testFile.txt");
}
}
```

Note: The FileResult in ASP.NET MVC is one of the most open-ended ActionResults and can handle a lot of different scenarios. It can accept byte arrays, FileStreams, and URLs of files and in all scenarios return or download the file specified. So, in simple words, we can say that, when we need to return and display the file content or when we want to download the file in ASP.NET MVC Application, then we need to use the FileResult.

In the next article, I am going to discuss **ContentResult in ASP.NET MVC Application**. Here, in this article, I try to explain **FileResult in ASP.NET MVC** application with examples. I hope this article will help you with your need. I would like to have your feedback. Please post your feedback, question, or comments about this article.