Module 9: **Building Responsive Pages in ASP.NET MVC 5 Web Applications**

# ****Contents:****

|  |  |
| --- | --- |
|  | [Module Overview](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C09.html#P1) |
| **Lesson 1:** | [Using AJAX and Partial Page Updates](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C09.html#P2) |
| **Lesson 2:** | [Implementing a Caching Strategy](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C09.html#P3) |
| **Lab:** | [Building Responsive Pages in ASP.NET MVC 5 Web Applications](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C09.html#P4) |
|  | [Module Review and Takeaways](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C09.html#P5) |

# ****Module Overview****

Many web applications need to display large amount of information and graphics. Large volumes of data make web applications take longer to load. Instead of having all the elements on the page load multiple times, you can implement caching in your web applications to reduce the need to repeatedly load the same elements. You can also use partial page updates to reduce the need to load the entire webpage, by enabling the application to refresh only specific sections of the webpage.

### ****Objectives****

After completing this module, you will be able to:

|  |  |
| --- | --- |
| • | Implement partial page updates that help reduce the need to reload entire pages. |
| • | Implement caching to reduce the loading time of the different elements of a webpage. |

# Lesson 1: ****Using AJAX and Partial Page Updates****

While developing web applications, to update individual sections in the page, you may need to reload the entire webpage. Asynchronous JavaScript and XML (AJAX) in ASP.NET MVC 5 enables partial page updates, to help update sections of a webpage, without reloading the entire page. The **Ajax.ActionLink** helper helps implement partial page updates in your web application.

## ****Lesson Objectives****

After completing this lesson, you will be able to:

|  |  |
| --- | --- |
| • | Describe the benefits of using partial page updates. |
| • | Use AJAX in an ASP.NET MVC 5 web application. |
| • | Use the **Ajax.ActionLink** helper. |

## ****Why Use Partial Page Updates?****

ASP.NET and MVC facilitate server-side processing. Server-side processing enables HTML rendering to occur at the server side and the server usually generates the HTML. When you update a webpage, any updates or actions performed on the page require a round-trip request to the server. Such constant server requests affect the performance of the application.

The AJAX development model helps reduce the need for refreshing an entire webpage, each time an update of page content is required. AJAX uses JavaScript and XML to obtain information from the client system. AJAX creates webpages based on the XML information downloaded from the server. However, developing web applications by using AJAX is not easy, because it requires using complex technologies, such as JavaScript and XML. Microsoft includes a feature in ASP.NET called partial page updates that functions along with AJAX to reduce the need for refreshing an entire webpage, each time an update occurs.

Partial page updates use AJAX technologies to help update individual sections of a webpage, during postback. Partial page updates:

|  |  |
| --- | --- |
| • | Require fewer lines of code. |
| • | Help reduce the data sent to users, each time a webpage update occurs. |
| • | Increase the responsiveness of the web application. |

**Question**: How do partial page updates help in improving the responsiveness of a web application?

## ****Using AJAX in an MVC 5 Web Application****

To implement AJAX in your MVC 5 application, you need to create views that render only the updated content, and not the entire webpage. You can initially develop your web application without using AJAX, and then check the application for any functionality errors. This practice helps reduce the time required to troubleshoot the application. This practice also helps separate any application functionality errors from errors that occur while implementing AJAX.

To implement partial page updates, you need to create a view, called a partial view, which includes only the section that you need to update.

**An Example View**

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<div>

<div id="divMessage">@ViewBag.Message</div>

@Html.ActionLink("Refresh","HelloWorld")

</div>

</body>

</html>

In the preceding code example, **ActionLink** helps direct users to another view called **HelloWorld**.

**The Hello World View**

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>HelloWorld</title>

</head>

<body>

<div>

@ViewBag.Message

</div>

</body>

</html>

To implement AJAX in the preceding code example, you need to update the **HelloWorld** view to render only the content that is updated.

**A ViewBag Example**

@ViewBag.Message

With AJAX, the Javascript retrieves only a specific portion of a webpage, which you want to update, from the server. In the **ViewController** class, you need to update the **View** function to return the **PartialView** class, instead of the entire **View** class.

**Returning a Partial View in a Controller Action**

public class Default1Controller : Controller

{

//

// GET: /Default1/

public ActionResult Index()

{

ViewBag.Message = "Hello";

return View();

}

[HttpGet]

public PartialViewResult HelloWorld()

{

ViewBag.Message = "Hello World";

return PartialView();

}

}

Optionally, you can add the **HttpGet** or **HttpPost** attributes before the **View** function. These attributes help indicate if the partial page update should be performed over the **HTTP POST** or **HTTP GET** method.

**Question**: What is the mandatory action that you should perform to implement partial page updates in your web application?

## ****The Ajax.ActionLink Helper****

You can use the **Ajax.ActionLink** helper to trigger partial page updates. The **Ajax.ActionLink** helper helps initiate the Javascript, to obtain the updated HTML information from the view and replace or insert the updated HTML information at a specific location.

**Using the Ajax.ActionLink Helper**

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.8.3.js" type="text/javascript"></script>

<script src="http://ajax.aspnetcdn.com/ajax/mvc/3.0/jquery.unobtrusive-ajax.min.js"

type="text/javascript"></script>

</head>

<body>

<div>

<div id="divMessage">@ViewBag.Message</div>

@Ajax.ActionLink("Refresh","HelloWorld", new AjaxOptions{ HttpMethod = "POST", UpdateTargetId = "divMessage", InsertionMode = InsertionMode.Replace })

</div>

</body>

</html>

In the preceding example, parameters such as **HttpMethod** and **UpdateTargetId** are included along with the **Ajax.ActionLink** helper, to:

|  |  |
| --- | --- |
| • | Obtain the HTML information from the **HelloWorld** view, by using the **HTTP POST** method. |
| • | Replace the content in the **divMessage** HTML element. |

While using the **Ajax.ActionLink** helper, you need to include the jQuery and jQuery unobtrusive libraries in the same webpage, because the **Ajax.ActionLink** helper uses scripts from these two libraries.

**Question**: What is the primary function of the **Ajax.ActionLink** helper?

# Lesson 2: ****Implementing a Caching Strategy****

Web applications display information on a webpage by retrieving the information from a database. If the information that should be retrieved from the database is large, the application may take longer to display the information on a webpage. ASP.NET MVC 5 supports some caching techniques to help reduce the time required to process a user request.

Before implementing caching, you should first analyze if caching is relevant to your application, because caching is irrelevant to webpages whose content change frequently. To successfully implement caching in your web application, you need to familiarize yourself with the various types of caches, such as output cache, data cache, and HTTP cache.

## ****Lesson Objectives****

After completing this lesson, you will be able to:

|  |  |
| --- | --- |
| • | Describe the benefits of using caching. |
| • | Describe the output cache. |
| • | Describe the data cache. |
| • | Describe the HTTP cache. |
| • | Describe how to prevent caching for webpage content that changes frequently. |
| • | Configure caching. |

## ****Why Use Caching?****

Caching involves storing the information that is obtained from a database in the memory of a web server. If the content rendered by a webpage is static in nature, the content can be stored in caches or proxy servers. When a user requests content from a web application, caching ensures that the user receives content from the cache, thereby eliminating the need for repeated real-time processing.

Caching:

|  |  |
| --- | --- |
| • | Reduces the need to repeatedly retrieve the same information from the database. |
| • | Reduces the need to reprocess data, if a user places a request multiple times. |
| • | Helps improve the performance of a web application, by reducing the load on servers. |
| • | Helps increase the number of users who can access the server farm. |

However, caching does not help web applications that include frequent content changes. This is because, the content rendered from a cache may be outdated, when compared to the current information. Therefore, you should evaluate the content of your web application and analyze the impact of rendering outdated content, before implementing caching.

**Question**: How does caching help increase the scalability of a web application?

## ****The Output Cache****

Output cache allows ASP.NET engines to store the rendered content of a webpage in the memory of the web server. Therefore, when a user requests a specific page multiple times, the content is retrieved from the cache, thereby avoiding the execution of programming code in the server.

Output cache is a good complement to AJAX partial page updates. Output cache and partial page updates reduce the workload on the server and increase the number of user requests that a server can handle.

In ASP.NET MVC 5, you can implement output caching, by adding the **OutputCache** attribute to the controller.

**Configuring the Output Cache**

[OutputCache(Duration = 60)]

public PartialViewResult HelloWorld()

{

ViewBag.Message = "Hello World";

return PartialView();

}

The **OutputCache** attribute helps direct the rendering engine to the cache that contains results from the previous rendering process. The **Duration** parameter of the **OutputCache** attribute helps control the period of time in seconds for which data should be stored in the cache.

By default, the output cache stores only one copy of the rendered content, for each view. Consider a view with the **QueryString** input parameter that enables content to change based on the variable gathered from the database or a prior request. In this case, you can add the **VaryByParam** property to the **OutputCache** attribute, to store a copy of each unique combination of parameters in the cache.

**Caching by Parameter Value**

[OutputCache(Duration = 60, VaryByParam="ID")]

public PartialViewResult HelloWorld()

{

ViewBag.Message = "Hello World";

return PartialView();

}

In the preceding example, observe that the **VaryByParam** property refers to **QueryString**, instead of other MVC parameters. You can also use the **VaryByCustom** property.

**Using VaryByCustom**

[OutputCache(Duration = 60, VaryByCustom="browser")]

public PartialViewResult HelloWorld()

{

ViewBag.Message = "Hello World";

return PartialView();

}

You can add **browser** as the input parameter to the **VaryByCustom** property. The **browser** parameter helps store a copy of the rendered content corresponding to each browser that the application is run on. If you want to control and implement your own logic to determine when a new copy should be stored, you need to override the **GetVaryByCustomString** function in the **Global.asax** file of your project.

**Question**: How does the functioning of a web application that implements the output cache differ from an application that does not implement output cache?

## ****The Data Cache****

Web applications usually depend on the content in a database, to render content on a webpage. Databases sometimes encounter performance issues caused by poorly written queries, which can slow down the performance of database requests resulting in poor webpage performance. You can implement the data cache in your web application to avoid loading data from a database every time a user places a request. The **MemoryCache** class allows you to implement data cache in your web application. Implementing the data cache involves the following actions:

|  |  |
| --- | --- |
| 1. | Loading information from the database |
| 2. | Storing content in the **MemoryCache** object |
| 3. | Retrieving data from the **MemoryCache** object |
| 4. | Ensuring that content is available in the **MemoryCache** object; otherwise, reloading the content |

**Using the Memory Cache**

System.Data.DataTable dtCustomer = System.Runtime.Caching.MemoryCache.Default.

AddOrGetExisting("CustomerData", this.GetCustomerData(),

System.DateTime.Now.AddHours(1));

In the preceding example, the following parameters are specified:

|  |  |
| --- | --- |
| • | Key. The unique identifier of the object that should be stored in the memory cache. |
| • | Value. The object that should be stored in the memory cache. |
| • | AbsoluteExpiration. The time when the cache should expire. |

You can use the **AddOrGetExisting** function, instead of the **Add** function, to enable the application to refresh and retrieve data in one line of code. If the cache contains the relevant data, the **AddOrGetExisting** function retrieves the data from the cache. If the cache does not contain the relevant data, the **AddOrGetExisting** function allows adding the data to the cache, and then rendering the same data on the webpage.

**Question**: What are the benefits of implementing data caching in MVC applications?

## ****The HTTP Cache****

You can implement HTTP caching in the Browser Cache and the Proxy Cache.

**The Browser Cache**

Most web browsers store the content downloaded from web servers in their local cache. Storing data in the local cache helps remove the need to repeatedly download content from the server. Web browsers frequently check content for updates. If the content is updated in the server, web browsers download the content from the server, to attend to user requests. Otherwise, web browsers render content from the local cache.

**The Proxy Cache**

The functionality of the proxy cache is similar to the functionality of the browser cache. However, the cache is stored on a centralized server. Users can connect to the Internet or web servers by using this proxy server. Proxy servers store a copy of a web application in a manner similar to a web browser storing a copy of an application in the local drives. Many users can access the cache in a proxy server, while only one user can access the browser cache at a time.

**Question**: What is the difference between HTTP cache and output cache?

## ****Preventing Caching****

Caching can sometimes create issues in web applications, because if an application involves frequent content updates, caching prevents users from viewing these content updates. To resolve caching issues, you can implement an HTTP header called Cache-Control. The Cache-Control header indicates to the web browser how to handle the local cache. All HTTP clients, such as browsers and proxy servers, respond to the instructions provided in the Cache-Control header to determine how to handle the local cache of a web application.

**Additional Reading:**For more information about setting this header, go to: <http://go.microsoft.com/fwlink/?LinkID=288972&clcid=0x409>

You can use the **HttpCachePolicy.SetCacheability** method to specify the value of the Cache-Control header. The **HttpCachePolicy.SetCacheability** method helps control caching performance.

**Using SetCacheability**

Response.Cache.SetCacheability(HttpCacheability.Private);

In the preceding example, the **HttpCachePolicy.SetCacheability** method takes the **Private** enumeration value.

To prevent caching in your web application, you should set the Cache-Control header value to **NoCache**.

**Preventing Caching**

Response.Cache.SetCacheability(HttpCacheability.NoCache);

**Question**: What scenarios would require you to prevent caching for a web application?

## ****Demonstration: How to Configure Caching****

In this demonstration, you will see:

|  |  |
| --- | --- |
| • | How to configure the output cache for an MVC controller action. |
| • | Measure the difference that the configured cache makes to the delivery of the page. |

### ****Demonstration Steps****

|  |  |
| --- | --- |
| • | You will find the steps in the “Lesson 2: Implementing a Caching Strategy“ section on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD09_DEMO.md>. |

# ****Lab: Building Responsive Pages in ASP.NET MVC 5 Web Applications****

### ****Scenario****

Your manager has asked you to include comments for photos in the Photo Sharing application. Your manager has also highlighted that the performance of some pages in the application is too slow for a production site.

You want to ensure that comments for photos take minimal loading time, for which you decide to use partial page updates. You also want to return pages in quick time, while updated information is displayed, for which you decide to configure caching in your application.

### ****Objectives****

After completing this lab, you will be able to:

|  |  |
| --- | --- |
| • | Write controller actions that can be called asynchronously and return partial views. |
| • | Use common AJAX helpers to call asynchronous controller actions, and insert the results into Razor views. |
| • | Configure ASP.NET caches to serve pages in quick time. |

##### ****Lab Setup****

Estimated Time: 70 minutes

You will find the high-level steps on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD09_LAB_MANUAL.md>.

You will find the detailed steps on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD09_LAK.md>.

### ****Exercise 1: Using Partial Page Updates****

##### ****Scenario****

You have been asked to include a comment functionality on the photo display view of the Photo Sharing application. You want to ensure high performance by using AJAX partial page updates.

In this exercise, you will:

|  |  |
| --- | --- |
| • | Import a partially complete controller to add comments, and a view to delete comments. |
| • | Add code to the controller for partial page update. |

### ****Exercise 2: Optional-Configuring the ASP.NET Caches****

##### ****Scenario****

You have been asked to configure the ASP.NET caches in the Photo Sharing application to ensure optimal performance. Senior developers are particularly concerned that the All Photos gallery might render slowly because it will fetch and display many photos from the database at a time.

In this exercise, you will:

|  |  |
| --- | --- |
| • | Configure the output cache to store the photo index view. |
| • | Use the developer tools in Internet Explorer to examine the speed at which image files and pages render with and without caching. |
| • | Configure the output cache to store the results of the GetImage action so that image files can be returned from the cache. |

Complete this exercise if time permits.

### ****Review Question(s)****

**Check Your Knowledge**

**Discovery**

**In Exercise 2, why was the Request timing for /Photo not reduced for the first request when you configured the output cache for the index action?**

Show solution Reset

**Check Your Knowledge**

**Discovery**

**In Exercise 2, when you configured the output cache for the GetImage() action, why was it necessary to set VaryByParam="id"?**

Show solution Reset

# ****Module Review and Takeaways****

In this module, you used AJAX and partial page updates in MVC applications. AJAX and partial page updates help reduce the need for reloading the entire page, when a user places a request. Partial page updates also reduce the need for writing multiple lines of code, to update specific portions of a webpage. You also used caching to increase the performance of a web application.

### ****Real-world Issues and Scenarios****

Web applications usually run multiple queries to retrieve information from a database and render content on the webpages. Users sometimes complain that webpages take longer to load. Therefore, developers implement caching in the web application, to reduce the need to load data from a database, every time a user places a request. Caching helps webpages load faster, thereby increasing the performance of the application.

### ****Review Question(s)****