Module09

Building Responsive Pages in ASP.NET MVC Web Applications

Module Overview

- Using AJAX and Partial Page Updates
- Implementing a Caching Strategy

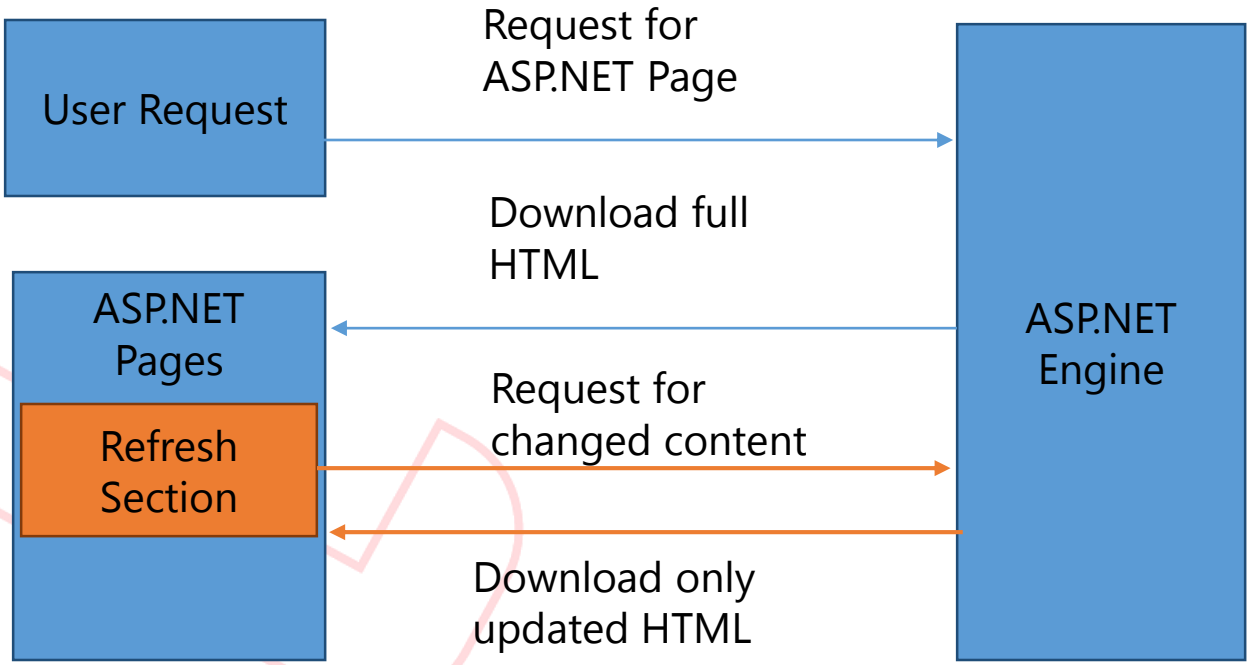
Lesson 1: Using AJAX and Partial Page Updates

- Why Use Partial Page Updates?
- Using AJAX in an MVC 4 Web Application
- The Ajax.ActionLink Helper

Why Use Partial Page Updates?

Partial page updates:

- Allow updates of individual sections of a webpage, during postback
- Increase the responsiveness of a web application



Using AJAX in an MVC 4 Web Application

To implement AJAX in your web application:

1. Create your web application without AJAX
2. Add or modify views, to render only the specific sections that you want to update on the webpage
3. Update the **ViewController** class to return the **PartialView** class

```
[HttpGet]
public PartialViewResult HelloWorld()
{
    ViewBag.Message = "Hello World";
    return PartialView();
}
```

The Ajax.ActionLink Helper

The **Ajax.ActionLink** helper:

- Helps obtain updated HTML information from the view
- Helps replace content in a specific location

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

Lesson 2: Implementing a Caching Strategy

- Why Use Caching?
- The Output Cache
- The Data Cache
- The HTTP Cache
- Preventing Caching
- Demonstration: How to Configure Caching

Why Use Caching?

Caching:

- Helps improve the performance of a web application by reducing the time needed to process a webpage

- Helps increase the scalability of a web application by reducing the workload on the server

The Output Cache

Benefits of caching in the output cache:

- The **OutputCache** attribute directs the rendering engine to the cache that contains results from the previous rendering process

```
[OutputCache(Duration = 60)]
```

- You can add the **VaryByParam** property to the **OutputCache** attribute to store a single copy of the most recent data in the cache

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

- You can add the **VaryByCustom** property to the **OutputCache** attribute to store multiple versions of the rendered content in the cache

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

The Data Cache

- You can use the **MemoryCache** object to store data in the memory

```
System.Data.DataTable dtCustomer =
System.Runtime.Caching.MemoryCache.Default
.AddOrGetExisting("CustomerData",this.GetCustomerData(),
System.DateTime.Now.AddHours(1));
```

- You can use the **AddOrGetExisting** function to reduce the code required to manage the cache

The HTTP Cache

Browser Cache:

- Includes a copy of the web application stored in local computer drive
- Allows only one user to access data, at a time

Proxy Cache:

- Includes a copy of the web application stored on a centralized server
- Allows multiple users to access data, at a time

Preventing Caching

- You can set the Cache-Control header value to **HttpCachePolicy.SetCacheability** to control the caching performance:

```
Response.Cache.SetCacheability(HttpCacheability.Private);
```

- You can set the Cache-Control header value to **NoCache** to prevent the caching performance:

```
Response.Cache.SetCacheability(HttpCacheability.NoCache);
```

Demonstration: How to Configure Caching

In this demonstration, you will see how to:

1. Configure the output cache for a controller action
2. Measure the time it takes to render an ASP.NET webpage
3. Clear the first network capture and request for a new webpage
4. Measure the time taken to render the requested webpage
5. Open **OperaController.cs** and configure the Index action to use the output cache
6. Measure the time it takes to render an ASP.NET webpage
7. Clear the network capture and request for a new webpage
8. Measure the time taken to render the requested webpage
9. Observe the improvement that the cache makes

Lab: Building Responsive Pages in ASP.NET MVC 4 Web Applications

- Exercise 1: Using Partial Page Updates
- Exercise 2: Optional—Configuring the ASP.NET Caches

Estimated Time: 60 minutes

Lab 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.

Lab Review

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

Module Review and Takeaways

- Real-world Issues and Scenarios
- Review Question(s)