Module 14: **Handling Requests 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/20486C14.html#P1) |
| **Lesson 1:** | [Using HTTP Modules and HTTP Handlers](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C14.html#P2) |
| **Lesson 2:** | [Using Web Sockets](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C14.html#P3) |
| **Lab:** | [Handling Requests 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/20486C14.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/20486C14.html#P5) |

# **Module Overview**

ASP.NET MVC provides functionalities that help develop web applications. However, ASP.NET MVC does not include functionalities that help change the encoding of the output. In such cases, you need to know how to use an HTTP module or HTTP handler, to facilitate such specific requests. Most web applications require two-way communication between the client and server systems. The web sockets protocol facilitates two-way communications in a robust manner.

### **Objectives**

After completing this module, you will be able to:

|  |  |
| --- | --- |
| • | Use HTTP modules and HTTP handlers. |
| • | Use web sockets. |

# Lesson 1: **Using HTTP Modules and HTTP Handlers**

Applications perform actions on a request in the HTTP pipeline, before rendering a webpage. You need to know how to use an HTTP module to implement custom authentication mechanism for a webpage, before rendering the webpage. You can also use an HTTP module to create code that renders content in a non-default encoding format that fits application needs. Sometimes, you may need the application to handle requests by using application logic that differs from the built-in ASP.NET page rendering logic. In such cases, you should know how to use the HTTP handler to process such specific requests.

## **Lesson Objectives**

After completing this lesson, you will be able to:

|  |  |
| --- | --- |
| • | Describe how HTTP modules intercept web requests. |
| • | Create HTTP modules. |
| • | Describe how HTTP handlers intercept web requests. |
| • | Create HTTP handlers. |
| • | Determine when to use HTTP modules and HTTP handlers. |

## **What Is an HTTP Module?**

An HTTP module is a program that runs application logic on a webpage, before ASP.NET renders the webpage. To understand the relevance of an HTTP module, you need to know about the HTTP request pipeline. The IIS HTTP request pipeline is a sequence of programs that run on requests, to help the application perform tasks. When the application receives a request, it passes the request to the **isapi\_aspnet.dll** ISAPI library and starts the ASP.NET execution pipeline, to process the request.

Then, the request passes through the **HttpApplication** instance of the application and the **HttpModule** library.

HTTP modules provide the following features to an application:

|  |  |
| --- | --- |
| • | Security. Provide support for the custom authentication and authorization logic of your application |
| • | Logging and s tatistics. Gather low-level information about the web application execution cycle, for monitoring and logging purposes |
| • | Custom headers and footers. Allow insertion of custom header information in the response of each request |

The following is a list of HTTP modules in ASP.NET:

|  |  |
| --- | --- |
| • | OutputCache |
| • | Session |
| • | WindowsAuthentication |
| • | FormsAuthentication |
| • | PassportAuthentication |
| • | UrlAuthorization |
| • | FileAuthorization |
| • | DefaultAuthentication |

If you configure HTTP modules in an application, HTTP modules apply to all the HTTP requests that the application receives. You cannot configure an HTTP module for a specific page.

**Question**: What are the benefits of using HTTP modules in web applications?

## **Creating HTTP Modules**

You can create an HTTP module by adding the **CustomModule** class to your project; this class implements the **IHttpModule** interface in your application.

**Coding a Custom Module**

public class CustomModule : IHttpModule

{

public CustomModule ()

{

}

public String ModuleName

{

get { return "CustomModule"; }

}

public void Init(HttpApplication application)

{

context.Response.Write("<h1>Demo<h1>");

}

}

In the **CustomModule** class, you need to implement the following two methods:

|  |  |
| --- | --- |
| • | ModuleName property. Provides a display name that enables other application code to identify the HTTP module. |
| • | Init function. Provides the location to implement all logic for an HTTP module. The **HttpApplication** class triggers this function when the application receives a request. |

**Registering an HTTP Module for IIS**

<configuration>

<system.webServer>

<modules>

<add name="CustomModule" type="CustomModule"/>

</modules>

</system.webServer>

</configuration>

**Question**: You have developed a custom HttpModule. What must you do to ensure that your ASP.NET application uses the custom module developed?

## **What Is an HTTP Handler?**

HTTP modules help apply logic to all HTTP requests that an application receives. The **HttpHandler** function is a processing engine that processes specific HTTP requests. For example, you can configure an **HttpHandler** function to handle \*.ashx files. You can map an **HttpHandler** function to a URL based on the file extension of the requested server-side page, such as \*.aspx. The following list describes some common **HttpHandler** functions:

|  |  |
| --- | --- |
| • | ASP.NET page handler (\*.aspx). This is the default HTTP handler for ASP.NET pages. |
| • | Web service handler (\*.asmx). This is the default HTTP handler for web service pages. |
| • | Generic w eb handler (\*.ashx). This is the default HTTP handler for all web handlers that do not have a user interface and include the **@WebHandler** directive. |
| • | Trace handler (trace.axd). This is a handler that helps display page trace information. |

You can use the \*.ashx file extension with the Http handler because:

|  |  |
| --- | --- |
| • | The extension does not include any page-rendering logic. |
| • | The extension allows developers to write logic to send responses to the client systems. |

You can use a custom HTTP handler with the **\*.rss** extension, to generate Real Simple Syndication (RSS) feed content for user requests. You can also use a custom handler to request images from the database and send the images to the client systems.

**Question**: What is the primary use of the HTTP generic handler (\*.ashx)?

## **Discussion: Scenarios for HTTP Modules and Handlers**

Consider the following scenarios. In each case, discuss with the rest of the class to determine whether an HttpModule or HttpHandler best suits the case.

|  |  |
| --- | --- |
| • | You are creating a photo sharing application, and you want to enable each user to discuss photos and cycling trips with their friends. You need to ensure that the application renders photos to users, from the database table, without requiring users to first save the file on their system. |
| • | You are creating a Representational State Transfer (REST)-based business application programming interface (API) that requires a custom HTTP header, before accepting a request. |
| • | You are creating a business API to provide content in custom XML format. |
| • | You developing an application that requires saving diagnostic information about the header of an HTTP request. |

# Lesson 2: **Using Web Sockets**

AJAX technologies are constantly evolving. The HTTP protocol does not cater to crucial requirements such as real-time information updates from the server. You need to know how to use web sockets to create two-way communication between the client and server systems. You also need to know when to use the traditional HTTP model and the web sockets protocol, based on the needs of your application.

## **Lesson Objectives**

After completing this lesson, you will be able to:

|  |  |
| --- | --- |
| • | Describe the WebSocket protocol. |
| • | Describe SignalR. |
| • | Use SignalR in an ASP.NET MVC 5 web application. |

## **What Is the WebSocket Protocol?**

The WebSocket protocol is a standard from the World Wide Web Consortium (W3C). The HTML 5 specification includes support for web sockets.

The protocol facilitates two-way communication between web browsers and web servers. The protocol helps overcome the issue of HTTP not providing two-way communication.

Developers often build two-way communication applications by using inefficient techniques such as long running loops and polling. These techniques consume large amounts of memory and other resources of both the server and client systems. The WebSocket protocol resolves the problems that arise with using such techniques by creating a constant socket connection between the client and server. All communication between the client and server systems then occurs through the socket. This socket eliminates the need to re-create a request multiple times, during real-time communications between the client and server systems. Older browsers may not support web sockets.

The web sockets work in a similar manner as traditional network sockets. The only difference is that during the initial handshake, web sockets use the Upgrade HTTP request that includes the **Upgrade** HTTP header. If the server accepts web sockets, the server returns a response with the status code, 101. Then, the client and server systems send the payload by using the socket. When the client system no longer requires communication with the server, the application sends a close connection payload, to notify the server to close the web socket.

**Question**: What is the key difference between traditional HTTP and web sockets?

## **What Is SignalR?**

SignalR is a set of components that simplifies the development of bidirectional real-time web applications, such as chat rooms in websites. SignalR uses WebSockets wherever possible to connect. This means that it supports all HTML5 compatible browsers. A key advantage of using SignalR is that it supports remote procedure calls (RPCs) with HTML or even .NET applications and can automatically switch to periodic polling for older browsers.

To install the package to your application, you can use NuGet to find the package, **Microsoft.AspNet.SignalR**.

After installation, you can implement the server-side code for the server logic.

**Using SignalR**

using Microsoft.AspNet.SignalR;

public class ChatHub : Hub {

public void Send(string name, string message)

{

Clients.All.broadcastMessage(name, message);

}

}

**Registering SignalR Middleware**

public class Startup

{

public void Configuration(IAppBuilder app)

{

app.MapSignalR();

}

}

SignalR also includes a JavaScript library that can be used by clients to help perform communications between client and server. The SignalR library depends on the jQuery library.

**Client-Side SignalR Code**

<script src="@Url.Content("~/Scripts/jquery.signalR-2.2.2.min.js")"></script> <script src="@Url.Content("~/signalr/hubs")"></script>

<script type="text/javascript">

$(function () {

var chat = $.connection.chatHub;

chat.client.broadcastMessage = function (name, message) {

alert(message);

});

$.connection.hub.start().done(function() {

$("#broadcast").click(function () {

chat.send(displayname, 'Hello');

});

});

});

</script>

**Question:**What is the key benefit of using SignalR, instead of WebSockets directly?

## **Demonstration: How to Add a Chat Room to a Web Application by using SignalR**

In this demonstration, you will see how to add a simple chat room to a web application by using SignalR.

### **Demonstration Steps**

|  |  |
| --- | --- |
| • | You will find the steps in the “Lesson 2: Using Web Sockets“ section on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD14_DEMO.md>. |

# **Lab: Handling Requests in ASP.NET MVC 5 Web Applications**

### **Scenario**

The Adventures Works board and managers are pleased with the Photo Sharing application, but have requested that interactivity should be maximized to encourage users to register and participate fully in the community. Therefore, you have been asked to add chat functionality to the application. Authenticated members should be able to start a chat on a particular photo from the Display view. Chat rooms for each photo should be separated from each other. Users in the chat room should be able to send a message to all other users in that chat room, and they should be able to see all the messages that have been sent since they joined the chat room.

You have decided to use SignalR to implement the chat room over Web Sockets.

### **Objectives**

After completing this lab, you will be able to:

|  |  |
| --- | --- |
| • | Install SignalR in an ASP.NET MVC 5 web application. |
| • | Configure SignalR on the server and create a SignalR hub. |
| • | Link to the required script files for SignalR in an MVC view. |
| • | Create the script for SignalR connections and send messages to groups. |

##### **Lab Setup**

Estimated Time: 60 minutes

You will find the high-level steps on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD14_LAB_MANUAL.md>.

You will find the detailed steps on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD14_LAK.md>.

### **Exercise 1: Creating a SignalR Hub**

##### **Scenario**

Before you can write JScript code on the client to connect to SignalR, you must configure and code a SignalR hub on the web server.

In this exercise, you will:

|  |  |
| --- | --- |
| • | Install SignalR in the Photo Sharing application. |
| • | Configure routing. |
| • | Create a SignalR hub to accept messages from clients and forward those messages to other clients who are chatting about the same photo. |

### **Exercise 2: Creating a Photo Chat View**

##### **Scenario**

Now that you have set up and configured SignalR and a SignalR hub on the server side, you must use JScript and the SignalR JScript library to send and receive messages on the client side.

In this exercise, you will:

|  |  |
| --- | --- |
| • | Create a new MVC controller action and Razor view to display the chat user interface for a particular photo. |
| • | Link to the JScript libraries that SignalR requires and write a client-side script to call the **Join()** and **Send()** methods on the hub. |
| • | Test the chat functionality. |

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

**Check Your Knowledge**

**Discovery**

**In the chat functionality that you created, each photo in the Photo Sharing application has a separate chat room. How is this separation possible with one SignalR hub?**

Show solution Reset

# **Module Review and Takeaways**

Most web applications do not limit themselves to rendering HTML content. They require functionalities that support custom logic such as custom authentication. To suit such cases, you can use HTTP modules and HTTP handlers to run custom logic before or after rendering a webpage. The web socket technology in MVC 5 provides support for two-way communication between client and server systems. This technology is useful for web applications that require constant updates between the client and server systems.

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

You create an application that obtains the latest product pricing information from the internal database. The pricing is constantly updated every time business users feel the need for updates. Therefore, you need to ensure that the application updates pricing information every five minutes. In such cases, you can to use the web socket technology to implement price update. You can also add code to download the product image stored in the product database by using a generic handler (\*.ashx file).

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