Module 13: **Implementing Web APIs 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/20486C13.html#P1) |
| **Lesson 1:** | [Developing a Web API](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C13.html#P2) |
| **Lesson 2:** | [Calling a Web API](https://www.skillpipe.com/api/2.1/content/urn:uuid:4ed7876c-8864-4cf5-91ee-491726dfe322@2020-12-11T17:05:09Z/OPS/html/20486C13.html#P3) |
| **Lab:** | [Implementing APIs 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/20486C13.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/20486C13.html#P5) |

# ****Module Overview****

Most web applications require integration with external systems such as mobile applications. You need to know how to use Web APIs to promote application interaction with external systems. You can use the Web API to implement Representational State Transfer (REST) services in your application. REST services help reduce application overhead and limit the data that is transmitted between client and server systems. You need to know how to call Web API services by using server-side code, jQuery code, and JSON.NET library to effectively implement REST-style Web APIs in your application.

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

After completing this module, you will be able to:

|  |  |
| --- | --- |
| • | Develop a Web API. |
| • | Call a Web API. |

# Lesson 1: ****Developing a Web API****

You need to know how to develop Web API for applications, because Web API facilitates creating APIs for mobile applications, desktop applications, web services, web applications, and other applications. By creating a Web API, you make the information in your web application available for other developers to use in their systems. Each web application has a different functional methodology; this difference can cause interoperability issues in applications. REST services have a lightweight design, and Web API helps implement REST services to solve the interoperability issues. You need to know how to use the different routing methods that ASP.NET provides to implement REST services.

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

After completing this lesson, you will be able to:

|  |  |
| --- | --- |
| • | Describe Web APIs. |
| • | Create a Web API. |
| • | Describe REST services. |
| • | Describe data return formats. |
| • | Explain how to use routes and controllers to implement REST in Web APIs. |
| • | Explore a Web API by using Internet Explorer as a client. |

## ****What Is a Web API?****

Web API is a framework that enables you to build Representational State Transfer (REST)-enabled APIs. REST-enabled APIs help external systems use the business logic implemented in your application to increase the reusability of the application logic. Web API facilitates two-way communication between the client system and the server through tasks such as:

|  |  |
| --- | --- |
| • | Instructing an application to perform a specific task |
| • | Reading data values |
| • | Updating data values |

Web API enables developers to obtain business information by using REST, without creating complicated XML requests such as Simple Object Access Protocol (SOAP). Web APIs use URLs in requests, thereby eliminating the need for complicated requests. For example, the following URL obtains information for a customer entity with the ID **1**: http://api.contoso.com/api/customers/1

**A Web API JSON Response**

[{"Id":1,"Name":"Tomato soup","Category":"Groceries","Price":1.0},{"Id":2,"Name":

"Yo-yo","Category":"Toys","Price":3.75},{"Id":3,"Name":"Hammer","Category":

"Hardware","Price":16.99}]

REST and Web API enable all kinds of different applications, including mobile device applications, to interact with services. In particular, REST and Web API provide the following benefits for mobile applications:

|  |  |
| --- | --- |
| • | They reduce the processing power needed to create complex request messages for data retrieval. |
| • | They enhance the performance of the application by reducing the amount of data exchange between client and server. |

**Question**: What is the key benefit of using REST with Web APIs?

## ****Routing in Web API****

When you create a new project by using the Web API template in Visual Studio 2017, it includes a default routing rule. This routing rule helps map HTTP requests to the Web API controllers and actions by using HTTP verbs and the request URL. You can make use of a naming convention to map requests to actions, or you can control the behavior of the mapping by using annotations on action methods.

**The Default API Route**

Like MVC web applications, Web API applications use routes to map requests to the right API controller and action. In the Visual Studio project templates, the default API route is defined in the WebApiConfig.cs file in the App\_Start folder.

**The Default API Route**

routes.MapHttpRoute(

name: "API Default",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

In the preceding code sample, observe that the default route includes the literal path segment **api**. This segment ensures that Web API requests are clearly separate from MVC controller routes, because Web API requests must start with **api**.

The first placeholder variable, {controller} helps identify the API controller to forward the request to. As for MVC controllers, Web API appends the word, Controller, to this value to locate the right API controller class. For example, Web API routes a request to the URI, **api/products**, to the controller called, **ProductsController**. As for MVC controllers, the optional placeholder variable, {id}, is sent to the action as a parameter.

You can also define your own API routes in the same manner as you do for MVC routes. Observe, however, that Web API routes can handle requests from many types of client systems, including mobile device applications, desktop applications, web applications, and web services. MVC routes only handle web browser requests.

**Using the Action Naming Convention**

The default Web API route does not include a placeholder variable for the action name. This is because Web API uses the HTTP verb and a naming convention to route requests to the right action within a given controller.

Clients can make HTTP requests with one of four standard verbs: **GET**, **POST**, **PUT**, and **DELETE**. Other verbs are possible. Web API looks for an action whose name begins with the requested HTTP verb. For example, if the client sends a **DELETE** request to the URI **api/products/23**, Web API looks for a controller called **ProductsController**. Within this controller, it locates an action whose name begins with **Delete**. According to the default route, the segment **23** is the **{id}** parameter. If there is more than one action whose name begins with **Delete**, Web API chooses the action that accepts a parameter called **id**.

**The HttpGet, HttpPut, HttpPost, and HttpDelete Attributes**

You can use the **HttpGet**, **HttpPut**, **HttpPost**, or **HttpDelete** attributes in your controller action to override the action naming convention. You can also use these verbs to specify that a function is mapped to a specific HTTP verb. The following table describes how the HTTP attributes map to the HTTP verbs.

| **Attribute** | **HTTP Verb** |
| --- | --- |
| HttpGet | GET |
| HttpPut | PUT |
| HttpPost | POST |
| HttpDelete | DELETE |

**Specifying the HTTP Verb**

public class ProductsController : ApiController

{

[HttpGet]

public Product FindProduct(id) {}

}

In the preceding code sample, observe that the HTTP attributes only allow mapping of one HTTP verb to an action in the controller.

**The AcceptVerbs Attribute**

The use of the **AcceptVerbs** attribute allows you to specify custom HTTP Verbs or multiple HTTP verbs to the same actions in the controller.

**The AcceptVerbs Attribute**

public class ProductsController : ApiController

{

[AcceptVerbs("GET", "HEAD")]

public Product FindProduct(id) { }

[AcceptVerbs("MKCOL")]

public void MakeCollection() { }

}

**The ActionName attribute**

By default, the action name is the name of the action you specify in the controller. You can use the **ActionName** attribute to specify the action name to be used in the routing.

**Mapping an Action**

public class ProductsController : ApiController

{

[HttpGet]

[ActionName("Thumbnail")]

public HttpResponseMessage GetThumbnailImage(int id);

[HttpPost]

[ActionName("Thumbnail")]

public void AddThumbnailImage(int id);

}

**Question**: What is the purpose of using the HTTP attributes?

## ****Creating a Web API****

Visual Studio 2017 provides a Web API project template that helps implement Web API in a project.

The following image shows the list of project templates available in Microsoft Visual Studio.

**FIGURE 13.1: THE NEW PROJECT DIALOG BOX**

To implement a Web API template in your project, you need to perform the following steps:

|  |  |
| --- | --- |
| 1. | In the **New Project** dialog box, click **ASP.NET Web Application (.NET Framework)**. |
| 2. | In the **New ASP.NET Web Application** dialog box, click **Web API**. |

The following image shows the New ASP.NET Web Application dialog box

**FIGURE 13.2: THE NEW ASP.NET WEB APPLICATION DIALOG BOX**

After selecting the Web API template, you need to add a new API controller class that derives from **ApiController**. The API controller class hosts application code for handling Web API requests. ASP.NET engine maps the URL together with the HTTP verb and the controller or the action function of a controller, in the following format.

<http verb> http://<hostname>/api/<entity name>/<parameters>

The HTTP verb communicates to the Web API about the operations that it should perform; whereas, the rest is to communicate which entity and operations to perform on. Therefore, HTTP plays an important role. For example, consider the following API controller class.

public class ProductsController : ApiController

{

public IEnumerable<Product> GetAllProducts()

{

}

public Product GetProductById(int id)

{

}

}

In the preceding code sample, note the controller classes **GetAllProducts** and **GetProductById**. The **GetAllProducts**controller class helps obtain a full list of products from the database. You can map the following URL and HTTP verb with the **GetAllProducts**controller class.

GET /api/products

The **GetProductById** controller class helps obtain a specific product by using the ID detail. You can map the following URL with the **GetProductById** controller class.

GET /api/products/id

**Question**: What is the syntax that the ASP.NET engine uses for mapping API controllers and action functions?

## ****RESTful Services****

REST uses URLs and HTTP verbs to uniquely identify the entity that it operates on and the action that it performs. REST helps retrieve business information from the server. However, in addition to data retrieval, business applications perform more tasks such as creating, updating, and deleting information on the database. Web API and REST facilitate handling such additional tasks. They use the HTTP method to identify the operation that the application needs to perform.

The following table provides information on some HTTP methods that Web API and REST use.

| **HTTP Verb** | **Description** |
| --- | --- |
| **GET** | Use this method with the following URL to obtain a list of all customers.  /api/customers |
| **GET** | Use this method with the following URL to obtain a customer by using the ID detail.  /api/customers/id |
| **GET** | Use this method with the following URL to obtain customers by using the category detail.  /api/customers?country=country |
| **POST** | Use this method with the following URL to create a customer record.  /api/customers |
| **PUT** | Use this method with the following URL to update a customer record.  /api/customers/id |
| **DELETE** | Use this method with the following URL to delete a customer record.  /api/customers/id |

**CRUD Operations**

public HttpResponseMessage PostCustomer(Customer item)

{

}

public void PutCustomer(int id, Customer item)

{

}

public void DeleteProduct(int id)

{

}

**Question**: What is the mandatory requirement of create and update requests?

## ****Data Return Formats****

When a client makes a request to a Web API controller, the controller action often returns some data. For GET requests, for example, this data might be all the properties of a specific product or all the properties of all the products. Web API can return this data in one of two formats: JavaScript Object Notation (JSON) or XML.

**JSON and XML Data Formats**

Both JSON and XML are text formats that represent information as strings. You can also use JSON outside the JavaScript code.

**A JSON Response**

{"Name":"Albert","Age":29,"Height":145,"Skills":["Programming", "Technical Writing"]}

**An XML Response**

<Employee Name="Albert" Age="29" Height="145">

<Skills>

<Skill Name="Programming" />

<Skill Name="Technical Writing" />

</Skills>

</Employee>

When a client makes a request, the client can specify the data format for the response. If the data format is not specified, Web API formats data as JSON by default.

**Media Formatters**

Web API uses a media formatter to format or serialize the information that a Web API REST service returns. Web applications usually use the JSON format to format the data that functions return. However, you can alternatively use the XML media formatter or add a custom media formatter to control the data returned. Media formatters format the content that the server renders to the client systems.

For example, consider that you want Web API to return CSV files. In this case, you need to create a custom media formatter, to create as output data files in the CSV format, instead of XML files in the JSON format.

**A Custom Media Formatter**

public class CsvFormatter : BufferedMediaTypeFormatter

{

}

Media formatter classes inherit from the following classes:

|  |  |
| --- | --- |
| • | MediaTypeFormatter. This is the fundamental class for all formatter classes. |
| • | BufferedMediaTypeFormatter. This is an extended class of formatter classes, which provides support for buffering and asynchronous operations. |

**Additional Reading:**For more information on creating media formatters, go to: <http://go.microsoft.com/fwlink/?LinkID=288992&clcid=0x421>

**Question**: Why should you use a media formatter for Web API REST services?

## ****Using Routes and Controllers in Web APIs****

ASP.NET uses a route table to map a URL and an API controller. When you create a project, ASP.NET adds a default route by using the Web API template. This default route helps support the operations of the REST-style Web APIs.

**The Default API Route**

routes.MapHttpRoute(

name: "API Default",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

Consider that you want to include multiple actions, such as creating customers with XML and JSON, in the same HTTP method. In this case, you cannot use the default route because it requires a new request for each HTTP method and URL combination. Therefore, you need to update the routing by modifying the Route Map in the **WebApiConfig** class.

**Supporting Multiple Operations**

routes.MapHttpRoute(

name: "ActionApi",

routeTemplate: "api/{controller}/{action}/{id}",

defaults: new { id = RouteParameter.Optional }

);

You can use the **WebApiConfig** class to enable multiple versions of API to coexist in the same project. For example, you can include **/api/v1/{controller}**as Version One of your API and include **/api/v2/{controller}** as a new version or Version Two of the API.

You may want to include supporting functions in the controller class, and hide the supporting functions from the REST functions. You can eliminate the exposure of the function to the REST interface by adding the **NonAction** attribute to the action function.

**Using NonAction**

[NonAction]

public string GetPrivateData()

{

}

By default, Web API exposes all public methods as REST services. You can prevent this by making the function **private**, but this action prevents application code in the same project from accessing the function. Therefore, you can use the **NonAction** attribute for functions that need to be in public, but do not need to be exposed in REST.

**Question**: What is the key benefit of using the routing map?

## ****Demonstration: How to Explore a Web API by Using Internet Explorer****

In this demonstration, you will see how to:

|  |  |
| --- | --- |
| • | Create a simple Web API for an existing ASP.NET web application. |
| • | Download and examine the JSON files that the Web API generates by using Internet Explorer. |

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

|  |  |
| --- | --- |
| • | You will find the steps in the “Lesson 1: Developing a Web API“ section on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD13_DEMO.md>. |

# Lesson 2: ****Calling a Web API****

After you complete the development of the Web API services, you need to create the client applications to call these services. Calling Web API services is different from calling WCF services. However, the methods that you need to use to call these services are similar, regardless of the platform. You need to know how to call Web API services by using server-side code, jQuery code and JSON.NET library, to effectively implement Web API services in most application platforms.

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

After completing this lesson, you will be able to:

|  |  |
| --- | --- |
| • | Call Web APIs by using server-side code. |
| • | Call Web APIs by using jQuery. |

## ****Calling Web APIs by Using Server-Side Code****

You can call REST-style services by using ASP.NET server-side code. You can use the **HttpWebRequest** class to create a manual HTTP request to the REST services. ASP.NET provides a .NET library that you can use in web applications to call REST-enabled Web API services from the .NET server. To use the .NET library, you need to install the **Microsoft.AspNet.WebApi.Client**NuGet package. This NuGet package provides access to the **HttpClient** class. The **HttpClient** class simplifies interacting with Web APIs, because it reduces coding efforts.

**Initializing the HttpClient Class**

HttpClient client = new HttpClient();

client.BaseAddress = new Uri("http://localhost/");

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

The last line of code in the preceding code sample informs the client system about the media type that the client system should use. The default media type that applications use is **application/json**. However, applications can use any other media type, based on the media type that the REST-style services support.

**Calling the API from Server-Side Code**

HttpResponseMessage response = client.GetAsync("api/customers").Result;

if (response.IsSuccessStatusCode)

{

var products = response.Content.ReadAsAsync<IEnumerable<Customer>>().Result;

}

else

{

Console.WriteLine("{0} ({1})", (int)response.StatusCode, response.ReasonPhrase);

}

After running the code in the preceding code sample, you need to define a data model that aligns itself with the one used by the Web API service to enable the .NET library to:

|  |  |
| --- | --- |
| • | Process the results of the server-side code. |
| • | Return results as .NET objects for the application to use. |

Then, you can use the **GetAsync** and **ReadAsAsync** methods to:

|  |  |
| --- | --- |
| • | Create requests to Web API REST services. |
| • | Parse the content into .NET objects. |

The **PostAsJsonAsync** function uses the HTTP POST method to call Web API services that support the POST method.

**Question**: What is the benefit of using the **Microsoft.AspNet.WebApi.Client**NuGet package?

## ****Calling Web APIs by Using jQuery Code****

You can call Web API services in the same manner as you call other services that use technologies such as WCF. You can also call Web API services by using the jQuery **ajax** function.

**Using the jQuery ajax Function**

$.ajax({

url: 'http://localhost/api/customers/,

type: 'GET',

dataType: 'json',

success: function (data) {

},

error: function (e) {

}

});

In the preceding code sample, observe the **dataType** parameter of the **ajax** function. You should set this parameter to **json** or another data type that the Web API service supports. Most applications use JSON because it is light weight. The **ajax** function has built-in functionalities that parse JSON results for the ease of developers.

**Using the stringify function**

var customer = {

ID:'1',

CustName: 'customer 1'

};

$.ajax({

url: 'http://localhost/api/customer',

type: 'POST',

data:JSON.stringify(customer),

contentType: "application/json;charset=utf-8",

success: function (data) {

},

error: function (x) {

}

});

**Question**: What is the benefit of using **JSON.stringify()** in the **ajax** function?

# ****Lab: Implementing APIs in ASP.NET MVC 5 Web Applications****

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

Your manager wants to ensure that the photos and information stored in the Photo Sharing application can be integrated with other data in web mash-ups, mobile applications, and other locations. To re-use such data, while maintaining security, you need to implement a RESTful Web API for the application. You will use this Web API to display the locations of photos on a Bing Maps page.

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

After completing this lab, you will be able to:

|  |  |
| --- | --- |
| • | Create a Web API by using the new features of ASP.NET MVC 5. |
| • | Add routes and controllers to an application to handle REST requests. |
| • | Call a REST Web API from jQuery client-side code. |

##### ****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_MOD13_LAB_MANUAL.md>.

You will find the detailed steps on the following page: <https://github.com/MicrosoftLearning/20486-DevelopingASPNETMVCWebApplications/blob/master/Instructions/20486C/20486C_MOD13_LAK.md>.

### ****Exercise 1: Adding a Web API to the Photo Sharing Application****

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

You have been asked to implement a Web API for the Photo Sharing application to ensure that photos can be used in third-party websites, mobile device applications, and other applications.

In this exercise, you will:

|  |  |
| --- | --- |
| • | Add a Web API controller for the Photo model class. |
| • | Configure formatters and routes to support the Web API. |
| • | Test the API by using Internet Explorer. |

### ****Exercise 2: Using the Web API for a Bing Maps Display****

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

You need to use the new Web API to obtain the photos in the client-side jQuery code. You will use the latitude and longitude properties to display these photos as pins on a Bing API map.

To create the map display in the Photo Sharing application, you must add a new view and action for the photo controller. You must also add a new template view because the Bing Maps AJAX control requires a different **<!DOCTYPE>** directive to the one in use elsewhere in the Photo Sharing application. You will import a JavaScript file with basic Bing Maps code in it. To this JavaScript file, you will add code to call the Web API, obtain photo details, and display them on the map.

In this exercise, you will:

|  |  |
| --- | --- |
| • | Create a new template view. |
| • | Create a map action, view, and script file. |
| • | Obtain and display photos. |
| • | Test the Bing Maps control. |

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

**Check Your Knowledge**

**Discovery**

**How do the API actions you added to the PhotoApiController controller in Exercise 1 differ from other actions in MVC controllers?**

Show solution Reset

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

You can use the Web API framework to facilitate creating REST-style Web API calls in applications. REST-style Web API is recommended for mobile applications because of the light weight design of REST services.

REST services use HTTP methods such as **GET**, **POST**, and **PUT** to notify the API of the action that it needs to perform. Web APIs use the media formatter and the JSON.NET library to serialize and deserialize information, respectively. You can call Web API services by using server-side code, jQuery code, and the JSON.NET library.

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

Consider that you develop a mobile application by using Web APIs and the application needs to use currency rate services. For this application, you cannot use WCF, because WCF can impede the performance of the application by using XML for data exchanges. Therefore, you should use REST and JSON in the application to reduce the data that is transmitted between the client system and the server.

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