

## Exercise13: Web API

### Objective

In this exercise you add a Web API controller to your project. You will use Swagger to test the controller.

This exercise will take around 30 minutes.

### Referenced material

This exercise is based on material from the chapter "Web API".

### Adding an API Controller

1	<p>Open the 'Begin' solution in Visual Studio and compile (Shift Ctrl+B).</p> <p>You will notice that it contains the model classes and the setup using the EntityFramework that you have used in previous labs to access the Forum.Data database.</p> <p>If you look in the Controllers folder you will see an api controller is already present. This was created when the the Web Api project template was chosen as the starting point for this "Begin" project. If you run the project you will see that Swagger has already gone to work creating documentation and giving you the option to "Try out" (test )this Weather forecast endpoint. Give it a go. Close the browser and have a look in the Program.cs file to see where the services and middleware for swagger are being configured. More on this later. Now let's create our own API.</p>
2	<p>We will start by creating data transfer objects, or DTOs, which will be used by our controller.</p> <p>Add a folder to the root of the project, called DTOs</p>
3	<p>Into the DTOs folder, add the following two classes:</p> <p><b>ThreadDTO:</b></p> <pre>public class ThreadDTO {     public int ThreadId { get; set; }     public string Title { get; set; }     public string UserName { get; set; }      public static ThreadDTO FromThread(Thread thread)     {         return new ThreadDTO         {             ThreadId = thread.ThreadId,             Title = thread.Title,             UserName = thread.UserName         };     }      public static IEnumerable&lt;ThreadDTO&gt; FromThreads(IEnumerable&lt;Thread&gt; threads)</pre>

```

    {
    return threads.Select(t => FromThread(t));
    }
}

```

### ForumDTO:

```

public class ForumDTO
{
    public int ForumId { get; set; }

    [Required]
    [MinLength(4)]
    public string Title { get; set; }

    public IEnumerable<ThreadDTO> Threads { get; set; }

    public static ForumDTO FromForum(Forum forum)
    {
        return new ForumDTO
        {
            ForumId = forum.ForumId,
            Title = forum.Title,
            Threads = ThreadDTO.FromThreads(forum.Threads)
        };
    }

    public static IEnumerable<ForumDTO> FromForums(IEnumerable<Forum> forums)
    {
        return forums.Select(f => FromForum(f));
    }
}

```

- |   |   |
|---|---|
| 4 | <p>In the ForumDTO class, note that we have added some data annotations to the Title property, to say that it is required and has a minimum length requirement. These requirements are specific to the Web API controller and would not apply to the MVC web pages that we have been working with up until now, as we have put them on the DTO rather than the model class itself. This is a slightly unrealistic difference to introduce into our Web API controller, but it serves to demonstrate that using a DTO allows you to specify very precisely where these attributes apply.</p> |
| 5 | <p>Right-click on the Controllers folder and chose Add – Controller... Make sure you select API (not MVC) from the list of options on the left pane. Select API Controller – Empty then click add.</p>  |

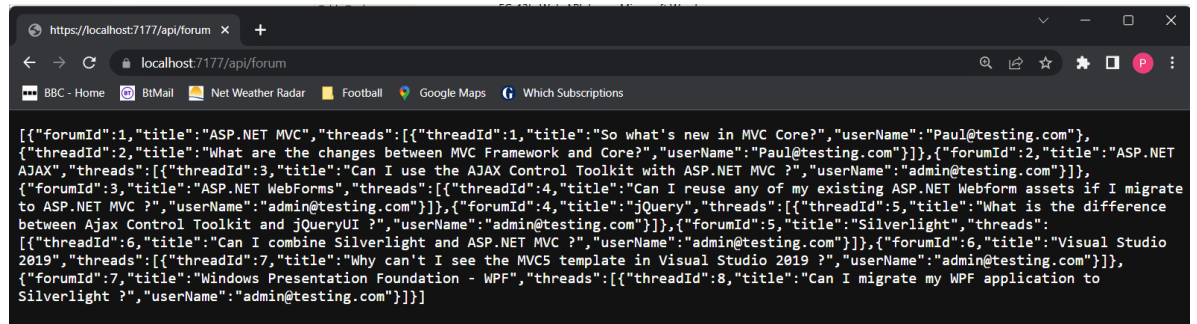
	 <p>Name the controller ForumController.</p>
6	Note that the Web API controller inherits ControllerBase (not Controller) and the attributes that the class has been decorated with.
7	<p>Add code to your new controller to inject a ForumDbContext into it:</p> <pre> private readonly ForumDbContext context;  public ForumController(ForumDbContext context) {     this.context = context; } </pre>
8	<p>Add two Get() methods to the controller, one to retrieve a list of forums, and one to retrieve a single forum.</p> <p>Note the use of the [HttpGet] attribute on both methods:</p> <pre> [HttpGet] public IEnumerable&lt;ForumDTO&gt; Get() {     Return ForumDTO.FromFourms(context.Forums); }  [HttpGet("{id}")] public ForumDTO Get(intid) {     var forum = context.Forums.Single(f =&gt; f.ForumId == id);     return ForumDTO.FromForum(forum); } </pre>
9	Run the application. Once it's running, modify the URL by appending the following

to the root:

/api/Forum

(Ignore Swagger for now).

You should see a JSON representation of all the forums and their threads:

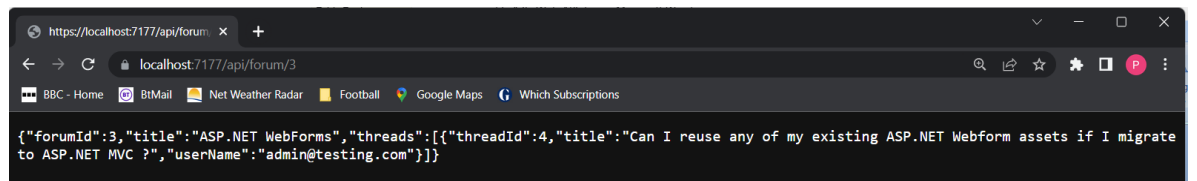


```
[{"forumId":1,"title":"ASP.NET MVC","threads":[{"threadId":1,"title":"So what's new in MVC Core?","userName":"Paul@testing.com"}, {"threadId":2,"title":"What are the changes between MVC Framework and Core?","userName":"Paul@testing.com"}]},{"forumId":2,"title":"ASP.NET AJAX","threads":[{"threadId":3,"title":"Can I use the AJAX Control Toolkit with ASP.NET MVC ?","userName":"admin@testing.com"}]},{"forumId":3,"title":"ASP.NET WebForms","threads":[{"threadId":4,"title":"Can I reuse any of my existing ASP.NET Webform assets if I migrate to ASP.NET MVC ?","userName":"admin@testing.com"}]},{"forumId":4,"title":"jQuery","threads":[{"threadId":5,"title":"What is the difference between Ajax Control Toolkit and jQueryUI ?","userName":"admin@testing.com"}]},{"forumId":5,"title":"Silverlight","threads":[{"threadId":6,"title":"Can I combine Silverlight and ASP.NET MVC ?","userName":"admin@testing.com"}]},{"forumId":6,"title":"Visual Studio 2019","threads":[{"threadId":7,"title":"Why can't I see the MVC5 template in Visual Studio 2019 ?","userName":"admin@testing.com"}]},{"forumId":7,"title":"Windows Presentation Foundation - WPF","threads":[{"threadId":8,"title":"Can I migrate my WPF application to Silverlight ?","userName":"admin@testing.com"}]}]
```

Append onto the new URL:

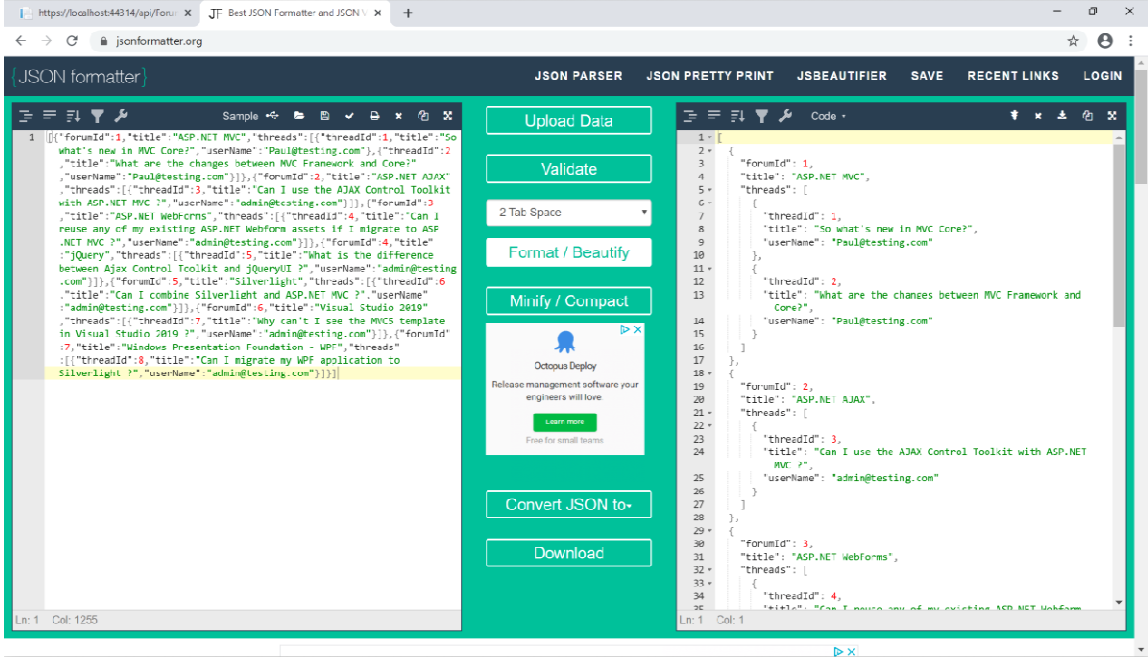
/3

Now you should see a JSON representation of just a single forum:



```
{"forumId":3,"title":"ASP.NET WebForms","threads":[{"threadId":4,"title":"Can I reuse any of my existing ASP.NET Webform assets if I migrate to ASP.NET MVC ?","userName":"admin@testing.com"}]}
```

- |    |   |
|----|---|
| 10 | <p>The data in the web browser is very hard to read! Copy it, then visit this web site:</p> <p><a href="https://jsonformatter.org">https://jsonformatter.org</a></p> <p>Paste the data into the left hand side of this site, and click the button marked "Format/Beautify". The right hand side of the site now contains the same data, but in a format which is much easier to read.</p> |
|----|---|

	
11	<p>Close the browser and let's add another endpoint to create a new Forum. Add the following code to the ForumController.</p> <pre> [HttpPost] public ActionResult&lt;ForumDTO&gt; Post(ForumDTO ForumDTO) {     Forum newForum = new Forum() { Title= ForumDTO.Title };      context.Add(newForum);     context.SaveChanges();     return CreatedAtAction(nameof(Get), new { id = ForumDTO.ForumId }, ForumDTO); } </pre>
12	<p>There is a limit to how much more we can do with a web browser when we're trying to see what's going on inside the HTTP messages. We could use other tools to help with this. In the past we could use Postman or Fiddler to help with this. Many still use these tools but now we have Swagger which comes out of the box.</p>
13	<p>Run the application again. As we've already seen the Swagger Documentation is displayed by default as the opening "page".</p> <p>Notice it documents each API controller and each endpoint they have.</p>
14	<p>Click the link for the Get /api/forum endpoint. Click Try it out. Click Execute.</p>

	<p>Scroll down and you can see the json data returned in the response and other request and response details.</p>
15	<p>Scroll further and click the link for the GET /api/Forum/{id}</p> <p>Click Try it out.</p> <p>Notice Swagger has worked out that this endpoint has int32 input parameter and that it is required .</p> <p>Enter 3 and click execute.</p> <p>Scroll down and see the response.</p>
16	<p>Scroll back up and replace the 3 with a 0.</p> <p>This time, an exception occurs in the program. If you started the program with F5, it will go into Debug mode – press F5 to continue. (If you started the program with Ctrl-F5 it won't go into Debug mode).</p> <p>Now, Swagger shows that the body of the response contains the exception type and message (System.InvalidOperationException: Sequence contains no elements), and that the status code is 500 Internal Server Error.</p> <p>The reason this has happened is that there is no forum with an ID of 0. But the response that we've got is not correct. The URL ends with /api/Forum/0, and the problem is that there is no resource which corresponds to this URL.</p>
17	<p>Stop the program from running and change the second Get() method as follows:</p> <pre> [HttpGet("{id}")] public ActionResult&lt;ForumDTO&gt;Get(int id) {     var forum = context.Forums.SingleOrDefault(f =&gt; f.ForumId == id);     if (forum == null)     {         return NotFound();     }     return ForumDTO.FromForum(forum); } </pre> <p>Start the program again, then re-send the request. Now, a request with an invalid ID returns a status of 404 Not Found, which makes much more sense.</p>
18	<p>Now let's create a new Forum.</p> <p>Scroll back up and click the link to POST /api/Forum.</p> <p>Click Try it out.</p>

	<p>Being a POST request we'll need to send the new Forum data as a json object in the body.</p> <p>Notice Swagger already knows this and suggests an appropriate schema for the body.</p> <p>Edit the body as follows:-</p> <pre> {   "title": "A new Test Forum" } </pre> <p>Click Execute.</p> <p>Scroll up and try the the link to GET /api/Forum again. Try it out and execute it.</p> <p>Hopefully you can see the newly added Forum.</p>
19	<p>Try posting another new Forum with a title of less than 3 characters (remember the validation attributes on our ForumDTO class.</p> <p>See the error response that comes back.</p>

### If You Have Time – Investigating Web API Clients

20	<p>We create Web API controllers in order for other programs to be able to connect to them.</p> <p>In this section, we will look at how to create clients in JavaScript/JQuery and in a c# console app.</p>
21	<p>Run the program.</p> <p>While the program is running, locate the file WebApiClientCSharp.sln, which can be found in the Assets/WebApiClientCSharp folder. Open it in a new instance of Visual Studio. (Don't close down the instance of Visual Studio in which QAForumWebApi is loaded – this should currently be running your application, including your Web API controller.)</p> <p>Have a look through the code, which shows how to access a Web API controller from a C# program – a simple console application, in this case.</p> <p>Run the client program, and check it works as expected. Then, stop both programs from running</p>
22	<p>Next, we're going to look at a JavaScript example which uses JQuery.</p> <p>In the folder wwwroot you will see there is a file called client_demo.html. Open the file and study the Javascript code at the bottom of the file.</p> <p>Run the QAForum application, then, when it starts, change the web browser's URL</p>

	to end with /client_demo.html. Try out both the buttons on the page, and check they work as expected.
23	<p>Take a moment to understand exactly what's happening in the Javascript demo.</p> <p>When you click a button, it does not reload the web page – if we wanted that behaviour, we would have simply used MVC.</p> <p>It does not fetch HTML from the server at all – it's possible to create an MVC action that returns a partial view, and then to write some Javascript that adds that HTML to the existing web page, but that is not what we've done here.</p> <p>Instead, the web page fetches JSON data from the server. The server has no knowledge of what's going to be done with that data. It's purely down to the web page to decide what action to take. What this particular web page does is create some HTML based on that JSON data, but there's nothing to stop another web site from using the same data in a different way. The code that generates this HTML is as follows:</p> <pre>var output = \$('&lt;div&gt;'); output.append(\$('&lt;h3&gt;').text('Here are the forums:')); \$.each(data, function (i, forum) {     output.append(\$('&lt;p&gt;').text(forum.title)); }); \$('#output').html(output);</pre> <p>This is an extremely common way to build modern websites, although the client code is often written using a Javascript framework such as React , Angular or even in c# usng Blazor WASM rather than plain Javascript/JQuery.</p> <p>All that's required of the client is that it has some way of sending http requests. All the above frameworks have a HTTP service out-of-the-box that provides this capability. The code in each framework would be very similar.</p>

You have now added a Web API controller to your project. You have understood how the most common HTTP verbs can be implemented in Web API, and how to test your controller using Postman.

You have also looked at two different techniques for consuming Web API data in client applications.