Agenda: Web API

- Introduction to Web API
- Swagger / Open API and Swashbuckle
- Action Return Types
- Use Web API Analyzers.

Introduction to Web API

ASP.NET Core Web API is a framework for building and consuming HTTP services that can reach a broad range of clients including browsers, phones and tablets.

- ASP.NET Core Web API is an ideal platform for building RESTful applications on the .NET Framework.
- We can use either XML or JSON as output. JSON is good for mobile apps with slow connections, for example you can call an API from jQuery and better utilize the client's machine and browser.
- ASP.NET Core Web API takes the best features from WCF Web API and merges them with the best features from MVC. WCF needs lot of configurations, URI Templates, Contracts and endpoints. Service implementation and consumption is not simple.

Walkthrough

Step 01 - Create Project in Visual Studio

Step 02 – Right click on the controllers folder add → New Item → Web → ASP.NET → Web API Controller class.

Step 03 - Let's just modify the methods to have better test data.

```
[Produces("application/json")]
[Route("api/Values")]
public class ValuesController : ControllerBase
{
   private static List<string> list = new List<string>
   { "Item 1", "Item 2", "Item 3", "Item 4", "Item 5" };
   // GET: api/values
   [HttpGet]
   public IEnumerable<string> Get()
   {
      return list;
   }
   // GET api/values/5
   [HttpGet("{id}")]
   public JsonResult Get(int id)
```

```
    {
        return Json(list[id]);
    }
    // POST api/values
    [HttpPost]
    public IActionResult Post([FromBody]string value)
    {
        list.Add(value);
        return NoContent();
    }
    // PUT api/values/5
    [HttpPut("{id}")]
    public IActionResult Put(int id, [FromBody]string value)
    {
        list[id] = value;
        return NoContent();
    }
    // DELETE api/values/5
    [HttpDelete("{id}")]
    public IActionResult Delete(int id)
    {
        list.RemoveAt(id);
        return NoContent();
    }
}
```

WEB API is REST Complaint, so it typically consists of Get(), Put(), Post(), Delete() methods.

Method	**URL Structure**
Get()	api/Values
GetItem(int i)	api/Values/i
Post (i)	api/Values/i with Post method
Delete(i)	api/Values/i with Delete method
Put(i,value)	api/values/I with Put method

Step 04. Testing in browser: Type the following URL in browser and view the output in both IE and FF or Chrome.

http://localhost: XXXX/api/values/1

http://localhost:XXXX/api/values

Note: The reason for the difference is that Internet Explorer and Firefox send different Accept headers, so the web API sends different content types in the responses.

Annotation with ApiController attribute

ASP.NET Core 2.1 introduces the [ApiController] attribute to denote a web API controller class.

```
[Route("api/[controller]")]
[ApiController]
public class ProductsController : ControllerBase
```

A compatibility version of 2.1 or later, set via SetCompatibilityVersion, is required to use this attribute at the controller level.

```
services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
```

The [ApiController] attribute is commonly coupled with ControllerBase to enable REST-specific behavior for controllers. ControllerBase provides access to methods such as NotFound and File.

Automatic HTTP 400 responses

Model validation errors automatically trigger an HTTP 400 response. Consequently, the following code becomes unnecessary in your actions:

```
if (!ModelState.IsValid)
{
    return BadRequest(ModelState);
}
```

Swagger / Open API and Swashbuckle

When consuming a Web API, understanding its various methods can be challenging for a developer.

Swagger, also known as OpenAPI, solves the problem of generating useful documentation and help pages for Web APIs.

It provides benefits such as

- Interactive documentation
- Client SDK generation.
- API discoverability.

Add NuGet package Swashbuckle.AspNetCore

```
services.AddSwaggerGen(options =>
{
   options.DescribeAllEnumsAsStrings();
   options.SwaggerDoc("v1", new Swashbuckle.AspNetCore.Swagger.Info
   {
      Title = "HTTP Web API Demo",
      Version = "v1",
      Description = "The is a demo of how to use Swagger in Web API",
      TermsOfService = "Terms of Service"
   });
});
```

In the **Startup.Configure** method, enable the middleware for serving the generated JSON document and the Swagger UI:

```
app.UseSwagger();
app.UseSwagger().UseSwaggerUI(c => {
            c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");
});
```

Action Return Types

ASP.NET Core offers the following options for Web API controller action return types:

1. Specific type
2. IActionResult
3. ActionResult<T>

1. Specific type

The simplest action returns a primitive or complex data type (for example, string or a custom object type).

Consider the following action, which returns a collection of custom Product objects:

```
[HttpGet]
public IEnumerable<Product> Get()
{
   return _repository.GetProducts();
}
```

2. IActionResult type

The IActionResult return type is appropriate when multiple ActionResult return types are possible in an action. Some common return types falling into this category are BadRequestResult (400), NotFoundResult (404), and OkObjectResult (200).

Synchronous Action:

```
[HttpGet("{id}")]
[ProducesResponseType(typeof(Product), StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
public IActionResult GetById(int id)
{
   if (!_repository.TryGetProduct(id, out var product))
   {
      return NotFound();
   }
   return Ok(product);
}
```

Asynchronous action

```
[HttpPost]
[ProducesResponseType(typeof(Product), StatusCodes.Status201Created)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
public async Task<IActionResult> CreateAsync([FromBody] Product product)
{
   if (product.Description.Contains("XYZ Widget"))
   {
            return BadRequest();
   }
   await _repository.AddProductAsync(product);

   return CreatedAtAction(nameof(GetById), new { id = product.Id }, product);
}
```

3. ActionResult<T> type:

ASP.NET Core 2.1 introduces the ActionResult<T> return type for Web API controller actions. It enables you to return a type deriving from ActionResult or return a specific type.

ActionResult<T> offers the following benefits over the IActionResult type:

- The [ProducesResponseType] attribute's Type property can be excluded. For example, [ProducesResponseType(200, Type = typeof(Product))] is simplified to [ProducesResponseType(200)]. The action's expected return type is instead inferred from the T in ActionResult<T>.
- Implicit cast operators support the conversion of both T and ActionResult to ActionResult<T>. T converts to ObjectResult, which means return new ObjectResult(T); is simplified to return T;.

Synchronous action

```
[HttpGet("{id}")]
[ProducesResponseType(StatusCodes.Status404NotFound)]
public ActionResult<Product> GetById(int id)
{
  if (!_repository.TryGetProduct(id, out var product))
  {
    return NotFound();
  }
  return product;
}
```

Asynchronous action

```
[HttpPost]
[ProducesResponseType(StatusCodes.Status201Created)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
public async Task<ActionResult<Product>> CreateAsync(Product product)
{
  if (product.Description.Contains("XYZ Widget"))
  {
    return BadRequest();
  }
  await _repository.AddProductAsync(product);
  return CreatedAtAction(nameof(GetById), new { id = product.Id }, product);
}
```

Use web API analyzers

The analyzers work with controllers annotated with ApiControllerAttribute, while building on API conventions.

Nuget Package: **Microsoft.AspNetCore.Mvc.Api.Analyzers**

```
// GET api/contacts/{guid}
[HttpGet("{id}", Name = "GetById")]
[ProducesResponseType(typeof(Contact), StatusCodes.Status200OK)]
public IActionResult Get(string id)
{
   var contact = _contacts.Get(id);

   if (contact == null)
   {
      return NotFound();
   }

   return Ok(contact);
}
```

The preceding action documents the HTTP 200 success return type but doesn't document the HTTP 404 failure status code.

The analyzer reports the missing documentation for the HTTP 404 status code as a warning.

An option to fix the problem is provided.