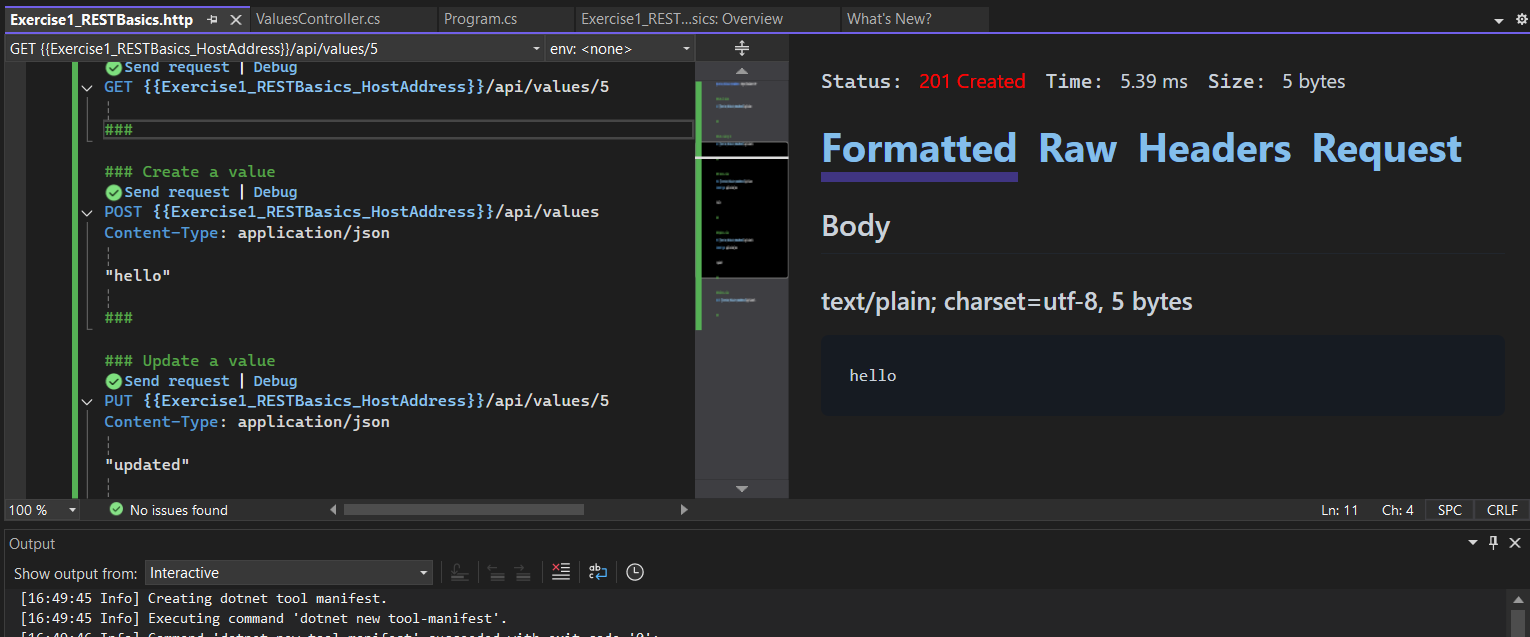
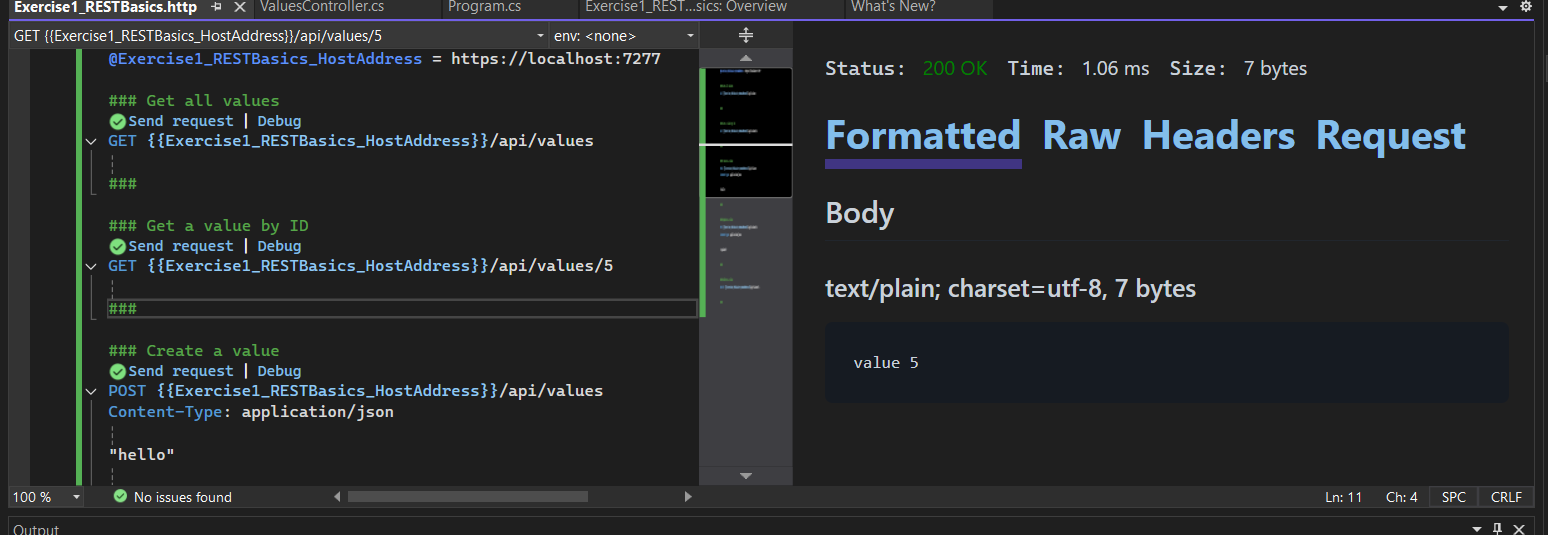
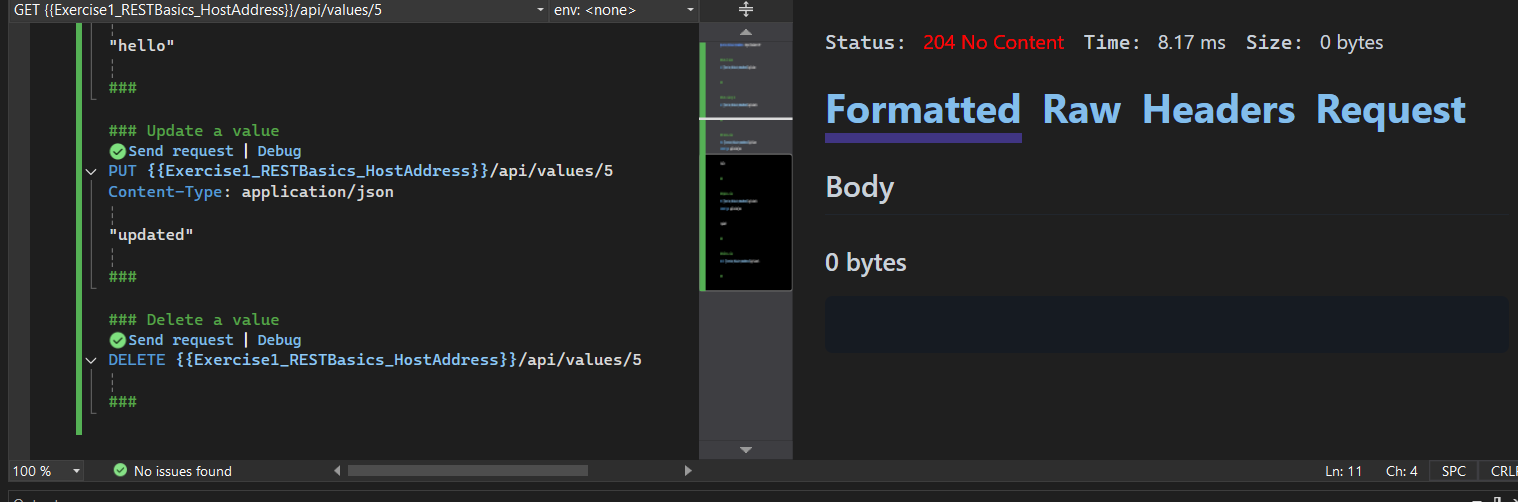
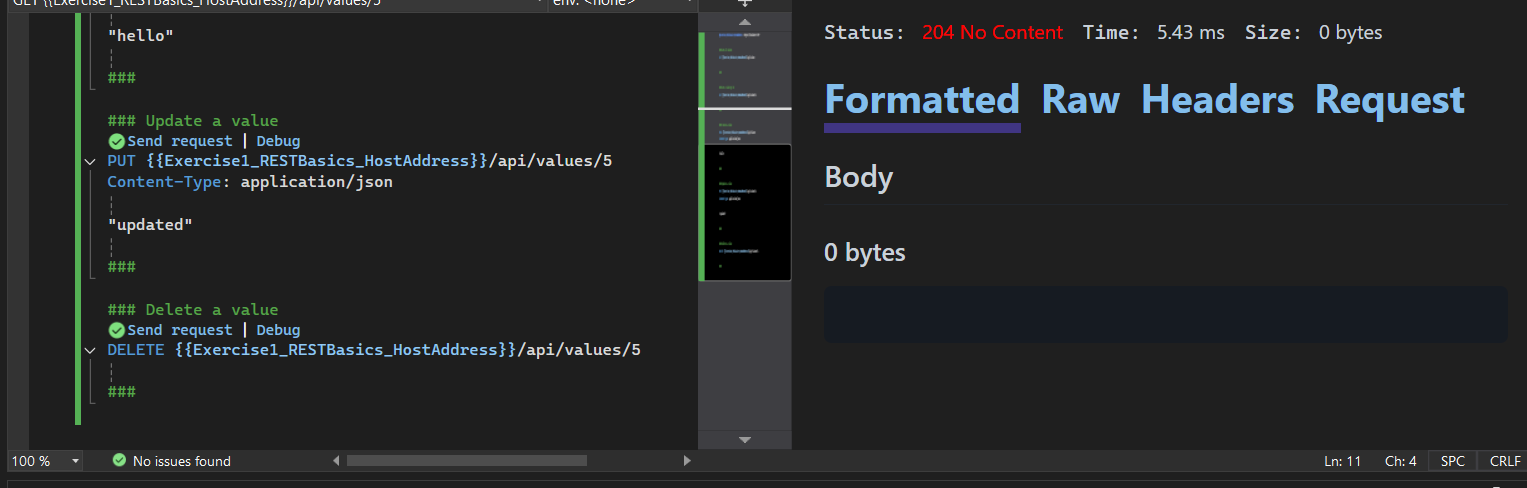
# Exercise 1: RESTful API Basics

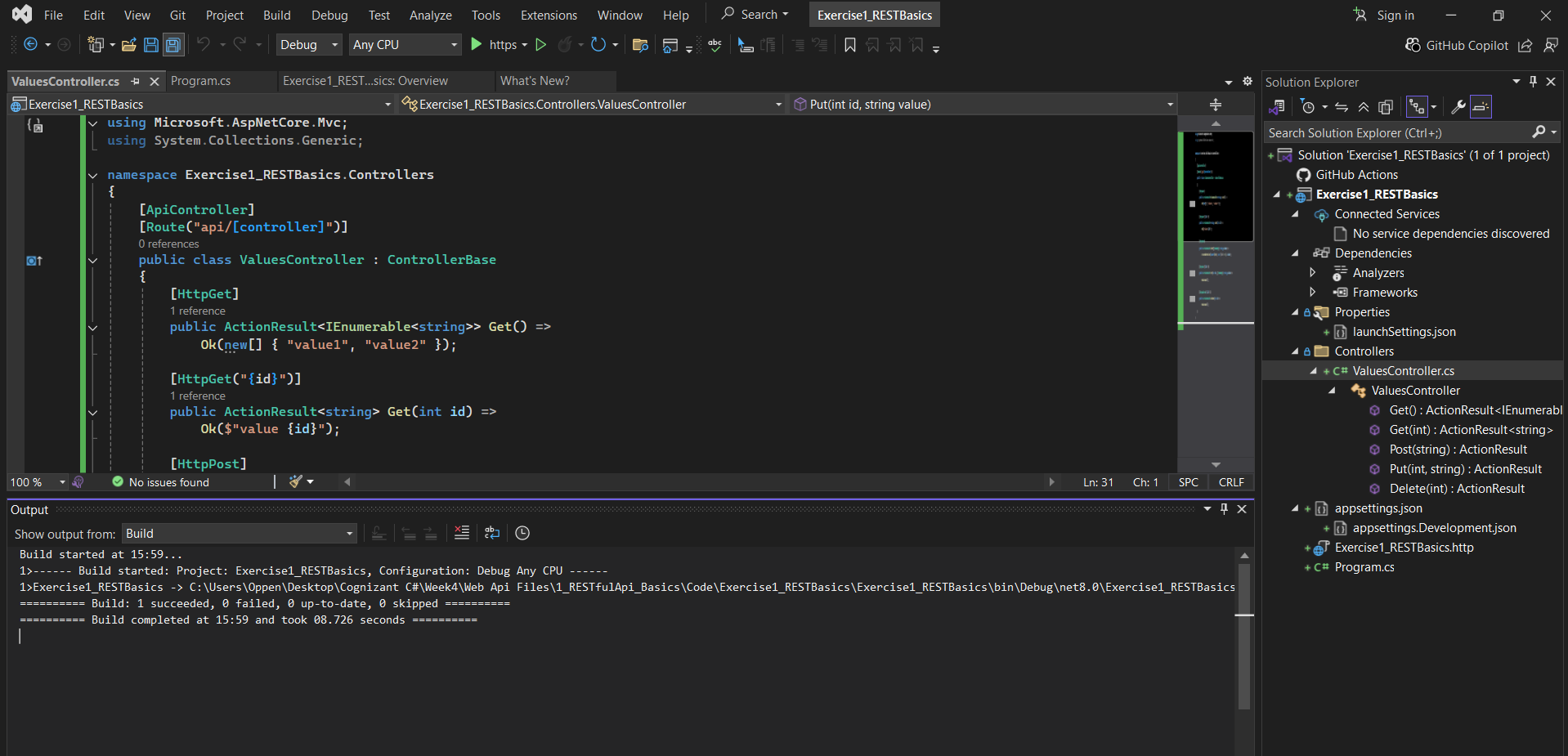
WEEK4\_EXERCISE1\_RESTful\_API\_Basics

## Steps Performed

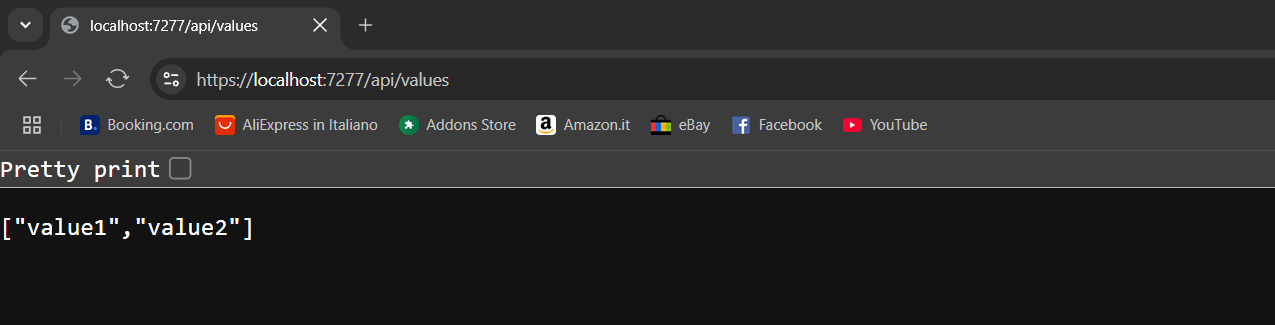
1. 1. Created an ASP.NET Core Web API project named "Exercise1\_RESTBasics" in Visual Studio 2022 under the folder Week4/Web Api Files/1\_RESTfulApi\_Basics/Code.
2. 2. Deleted the default WeatherForecastController from the Controllers folder.
3. 3. Updated Program.cs to include top-level statements and configured AddControllers and MapControllers.
4. 4. Added a new ValuesController.cs in Controllers with [HttpGet], [HttpPost], [HttpPut], and [HttpDelete] action methods.



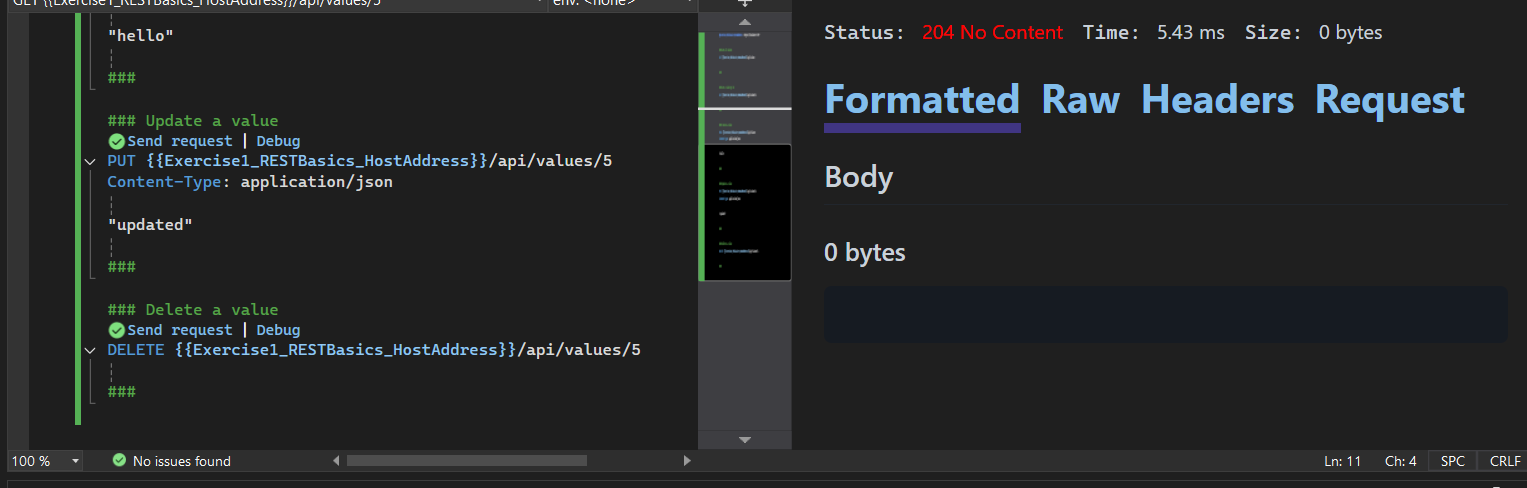
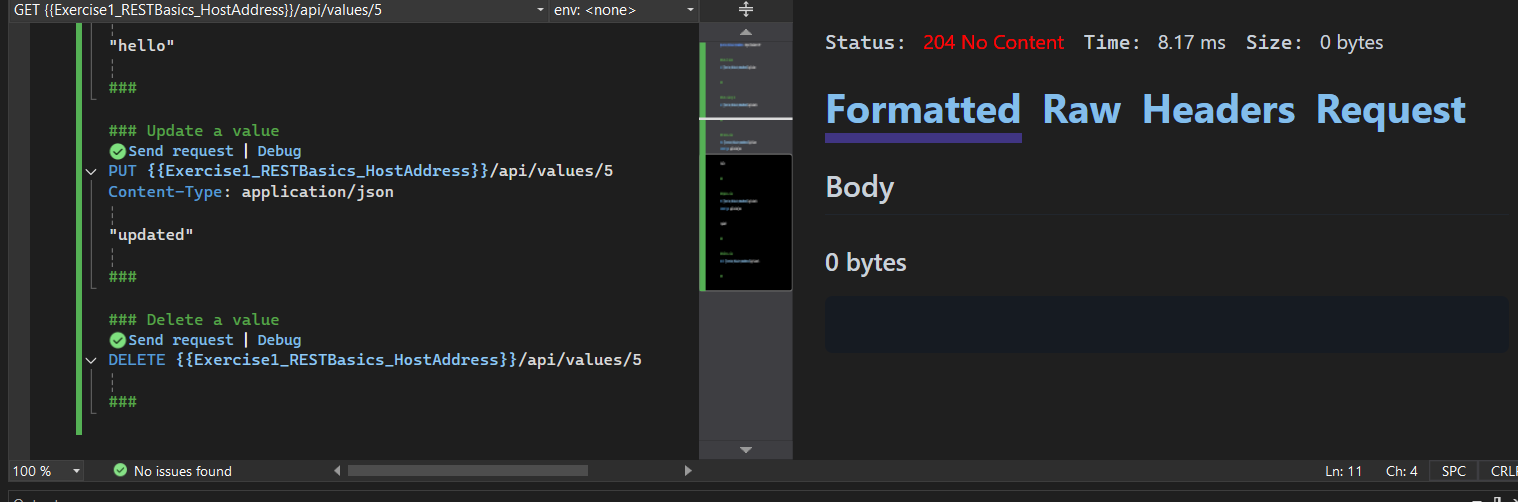
1. 5. Built the solution (Ctrl+Shift+B) and confirmed "Build succeeded" in the Output pane.



1. 6. Ran the API (F5) and navigated to https://localhost:<port>/api/values to verify the GET endpoint returned ["value1","value2"].



1. Tested GET by ID, POST ("hello"), PUT ("updated"), and DELETE endpoints using the .http file, confirming appropriate status codes (200, 201, 204).



## Conceptual Explanations

### 1. RESTful Web Service, Web API & Microservice

- RESTful Web Service: An architectural style using HTTP for communication, stateless operations, and resource-based URLs.  
- Web API: Framework for building HTTP services that return JSON/XML and are not tied to SOAP; ideal for REST.  
- Microservice: Small, independent services that focus on a single business capability and communicate over lightweight protocols.

### Features of REST Architecture

- Representational State Transfer: Resources are represented via URIs and manipulated through representations.  
- Stateless: Each request contains all information needed; server does not store client context.  
- Messages: Uses standard HTTP methods and status codes for communication.

### Difference between WebService & Web API

- WebService: Often SOAP-based, relies on WSDL, typically XML-only.  
- Web API: Uses HTTP directly, can return JSON/XML, lighter-weight, ideal for web and mobile clients.

### HttpRequest & HttpResponse

- HttpRequest: Message sent by client containing method, URL, headers, and optional body.  
- HttpResponse: Server’s reply containing status code, headers, and optional body.

### HTTP Action Verbs

- HttpGet: Retrieve a resource.  
- HttpPost: Create a new resource.  
- HttpPut: Update an existing resource.  
- HttpDelete: Remove a resource.  
These are declared with attributes [HttpGet], [HttpPost], etc., on controller methods.

### HTTP Status Codes in Web API

- Ok (200): Request succeeded and response contains result.  
- Created (201): Resource created successfully.  
- BadRequest (400): Client sent an invalid request.  
- Unauthorized (401): Authentication required or failed.  
- InternalServerError (500): Server encountered an error.

### Structure of a Simple Web API

- Controller: Class inheriting from ControllerBase or decorated with [ApiController].  
- Action Methods: Public methods with HTTP verb attributes handle requests.  
- Program.cs / Startup.cs: Configure services and middleware (AddControllers, MapControllers).

### Web API Configuration Files

- Program.cs (or Startup.cs): Sets up dependency injection and middleware pipeline.  
- appsettings.json: Stores configuration such as connection strings, logging settings.  
- launchSettings.json: Defines launch profiles for IIS Express or Kestrel, including launch URLs.  
- Route.config & WebAPI.config (ASP.NET 4.x): Define route patterns and Web API settings in older frameworks.