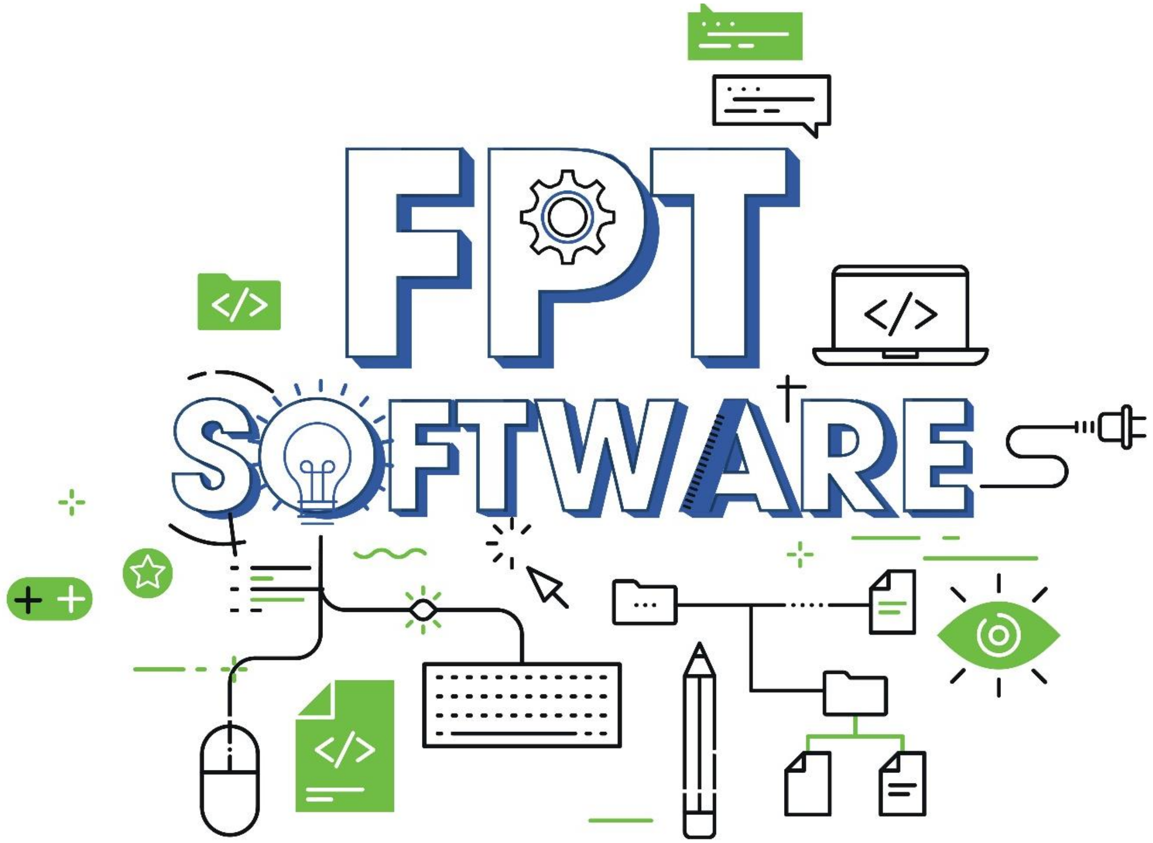
Introduction Web API



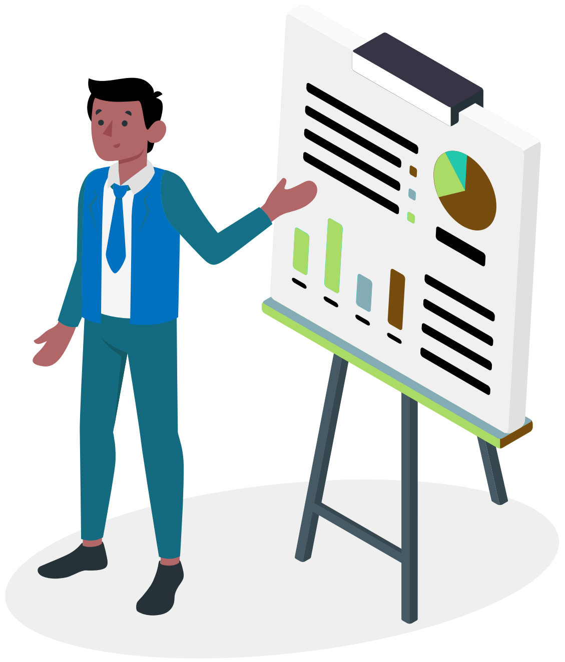
Agenda

1 • **Introduction Web API**

2 • **Web API Controller**

3 • **Parameter Binding in ASP.NET Web API**

4 • **Controller Action Return Type**



Lesson Objectives

- ❖ Understand the concept and purpose of Web API.
- ❖ Understand the role of a controller in a Web API.
- ❖ Understand the concept of parameter binding in ASP.NET Web API.
- ❖ Learn about the different types of parameter binding, such as model binding, query string binding, and route parameter binding.
- ❖ Learn about the different return types that can be used in controller actions in ASP.NET Core Web API.
- ❖ Understand the conventions for returning data, such as IActionResult, HttpResponseMessage, and specific data types.
- ❖ Explore how to handle different scenarios, such as returning JSON, XML, or other content types from a Web API.

Section 1

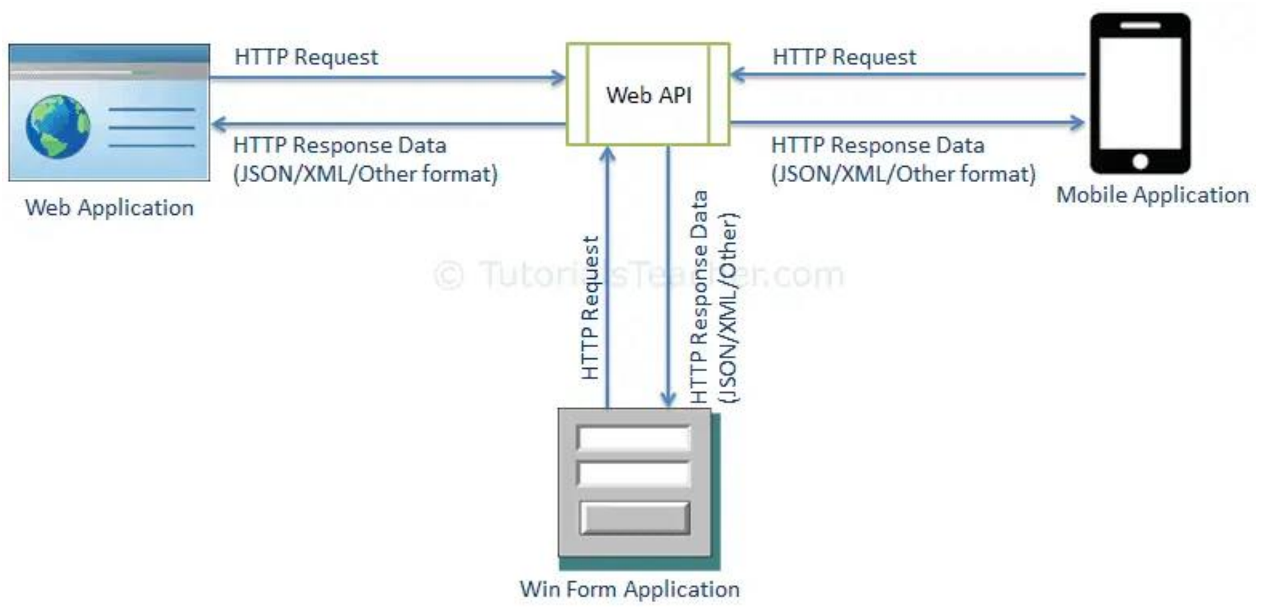
Introduction Web API

What is Web API?

- What is an API (**Application Programming Interface**): an application programming interface (**API**) is a set of subroutine definitions, protocols, and tools for building software and applications.
- Web API as the name suggests, is an API over the web that can be accessed using HTTP protocol.

ASP.NET Web API

- The ASP.NET Web API is **an extensible framework** for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile, etc…

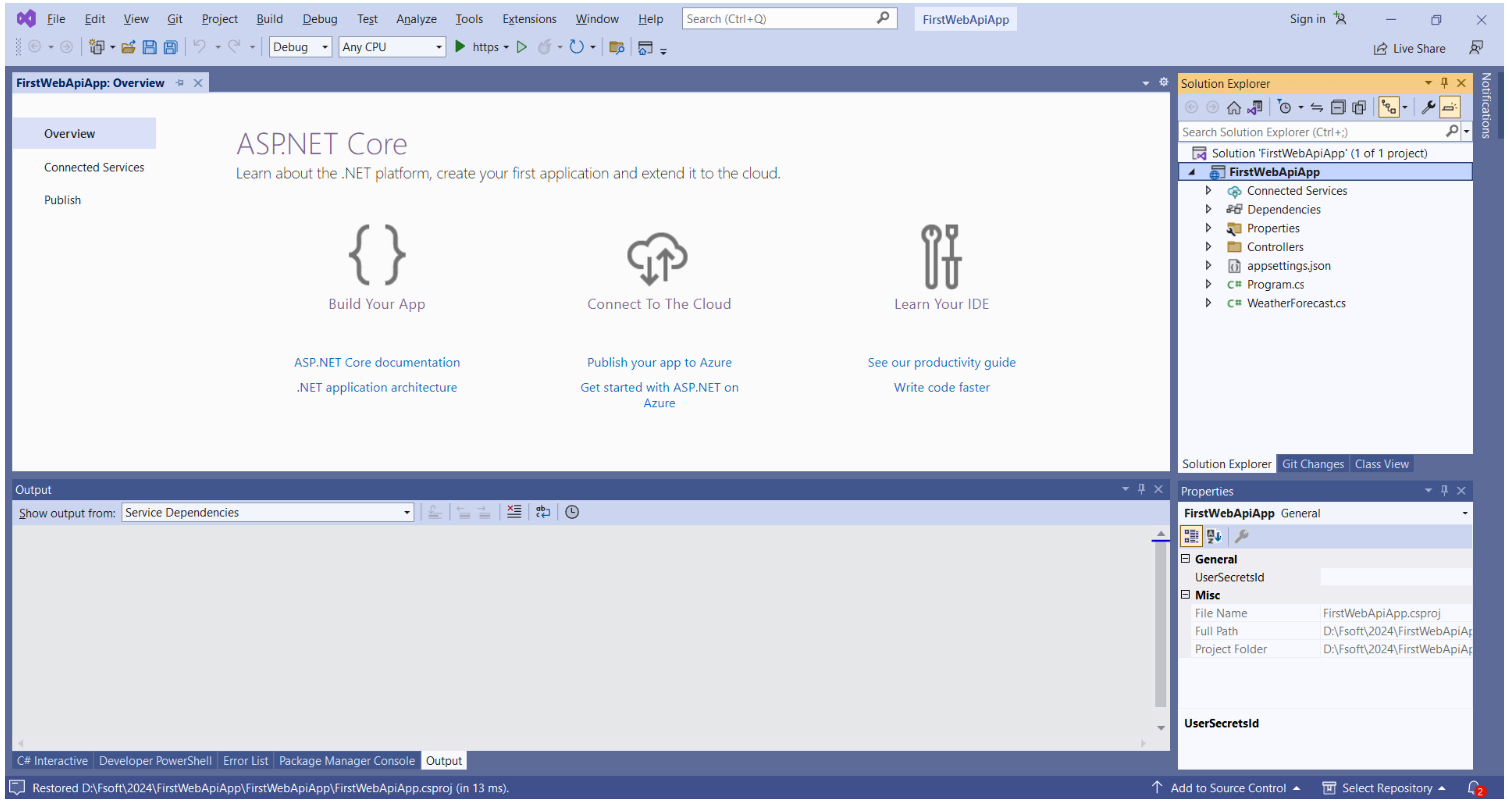

HTTP Request
HTTP Response Data (JSON/XML/Other format)
Web Application
Web API
HTTP Request
HTTP Response Data (JSON/XML/Other format)
Mobile Application
HTTP Request
HTTP Response Data (JSON/XML/Other)
Win Form Application
© TutorialsTeacher.com

ASP.NET Web API Characteristics

- ASP.NET Web API is an ideal platform for building RESTful services.
- ASP.NET Web API is built on top of ASP.NET and supports ASP.NET request/response pipeline
- ASP.NET Web API maps HTTP verbs to method names.
- ASP.NET Web API supports different formats of response data. Built-in support for JSON, XML, BSON format.
- ASP.NET Web API framework includes new HttpClient to communicate with Web API server. HttpClient can be used in ASP.MVC server side, Windows Form application, Console application or other apps.

Create New Web Api Project

- Demo create new Asp.NET Web API in Visual Studio
- Default Asp.Net Core Project Files



ASP.NET Core
Learn about the .NET platform, create your first application and extend it to the cloud.

Build Your App
ASP.NET Core documentation
.NET application architecture

Connect To The Cloud
Publish your app to Azure
Get started with ASP.NET on Azure

Learn Your IDE
See our productivity guide
Write code faster

Solution 'FirstWebApiApp' (1 of 1 project)
- FirstWebApiApp
  - Connected Services
  - Dependencies
  - Properties
  - Controllers
  - appsettings.json
  - Program.cs
  - WeatherForecast.cs

Show output from: Service Dependencies

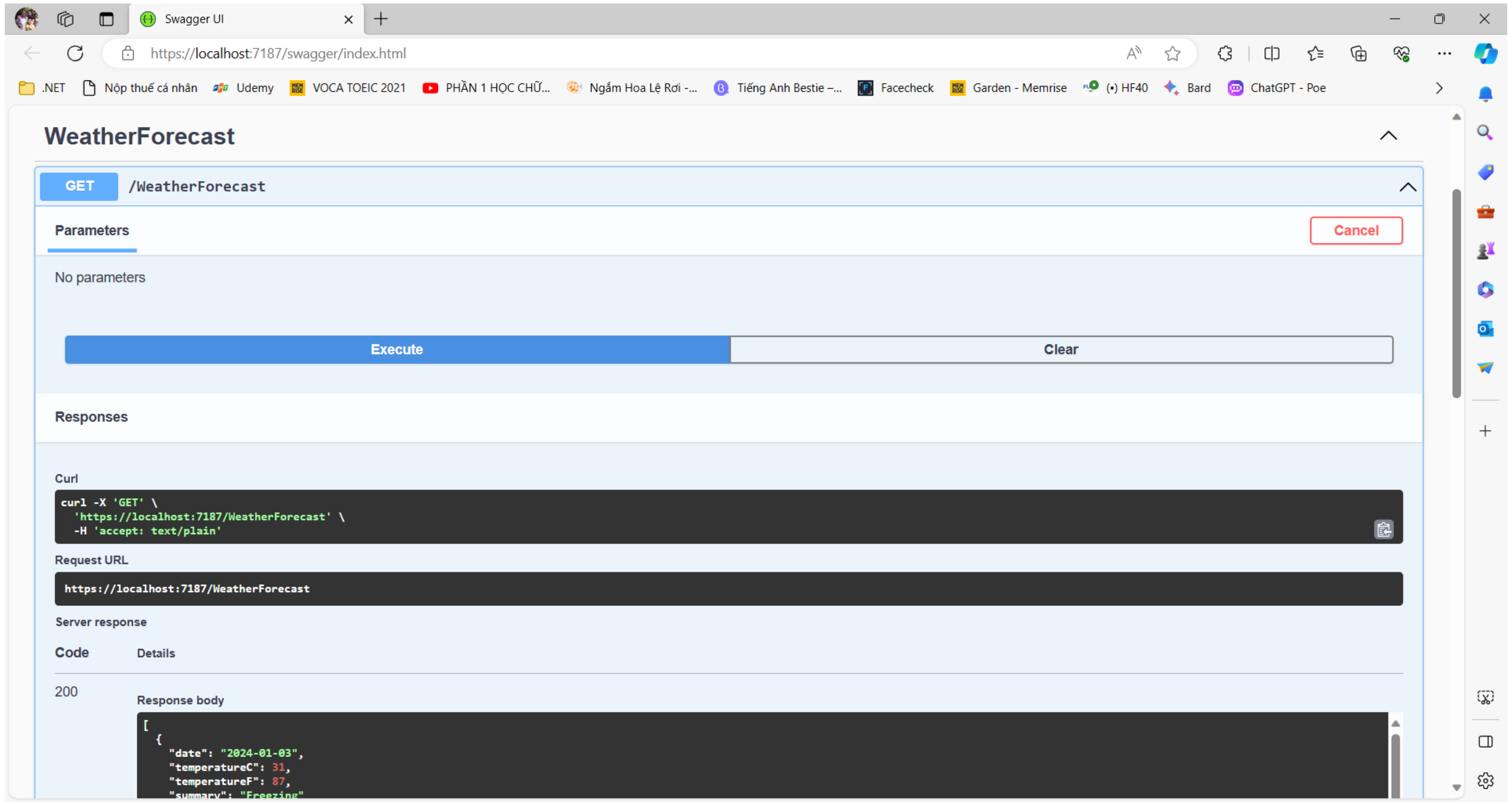
FirstWebApiApp General

General	
UserSecretsId	
Misc	
File Name	FirstWebApiApp.csproj
Full Path	D:\Fsoft\2024\FirstWebApiAp
Project Folder	D:\Fsoft\2024\FirstWebApiAp

Restored D:\Fsoft\2024\FirstWebApiApp\FirstWebApiApp\FirstWebApiApp.csproj (in 13 ms).

Test Web API with Swagger

- Demo



Curl

```
curl -X 'GET' \
  'https://localhost:7187/WeatherForecast' \
  -H 'accept: text/plain'
```

Request URL

```
https://localhost:7187/WeatherForecast
```

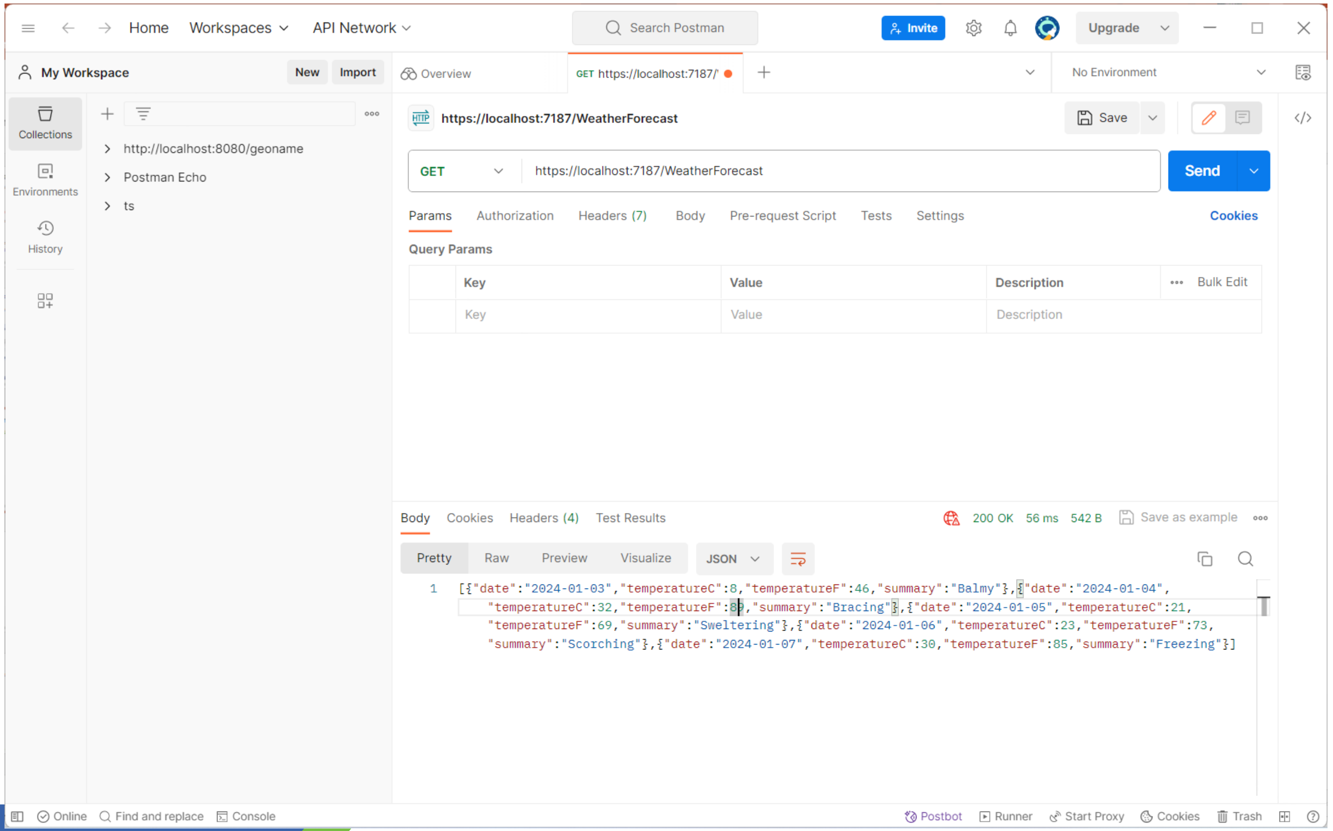
Server response

Code	Details
200	Response body

```
[
  {
    "date": "2024-01-03",
    "temperatureC": 31,
    "temperatureF": 87,
    "summary": "Freezing"
```

Test Web API with Postman

- Download Postman here: [Download Postman | Get Started for Free]
- Demo Test Web API with postman



https://localhost:7187/WeatherForecast

GET https://localhost:7187/WeatherForecast

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body | Cookies | Headers (4) | Test Results — 200 OK 56 ms 542 B — Save as example

Pretty | Raw | Preview | Visualize | JSON

```
[{"date":"2024-01-03","temperatureC":8,"temperatureF":46,"summary":"Balmy"},{"date":"2024-01-04",
    "temperatureC":32,"temperatureF":89,"summary":"Bracing"},{"date":"2024-01-05","temperatureC":21,
    "temperatureF":69,"summary":"Sweltering"},{"date":"2024-01-06","temperatureC":23,"temperatureF":73,
    "summary":"Scorching"},{"date":"2024-01-07","temperatureC":30,"temperatureF":85,"summary":"Freezing"}]
```

Lesson Summary

- ➢*What is a Web API*
- ➢*ASP.NET Web API Characteristics*
- ➢*Creating a new project*
- ➢*Default files*
- ➢*Testing web API with Postman and Swagger*

Section 2

Web API Controller

Web API Controllers

- Web API Controller is similar to ASP.NET MVC controller. It handles incoming HTTP requests and send response back to the caller.
- Based on the incoming request URL and HTTP verb (*GET/POST/PUT/PATCH/DELETE*), Web API decides which Web API controller and action method to execute.

  Example:

  - **Get()** method will handle HTTP GET request,
  - **Post()** method will handle HTTP POST request,
  - **Put()** method will handle HTTP PUT request
  - **Delete()** method will handle HTTP DELETE request for the above Web API.

What is a Controller?

- A controller in ASP.NET Core API is a C# class that handles incoming HTTP requests and produces the corresponding HTTP responses.
- Controllers are responsible for processing client requests, executing the necessary logic, and returning the appropriate responses.
- Each controller handles specific routes and actions defined by attributes, such as [HttpGet], [HttpPost], etc.

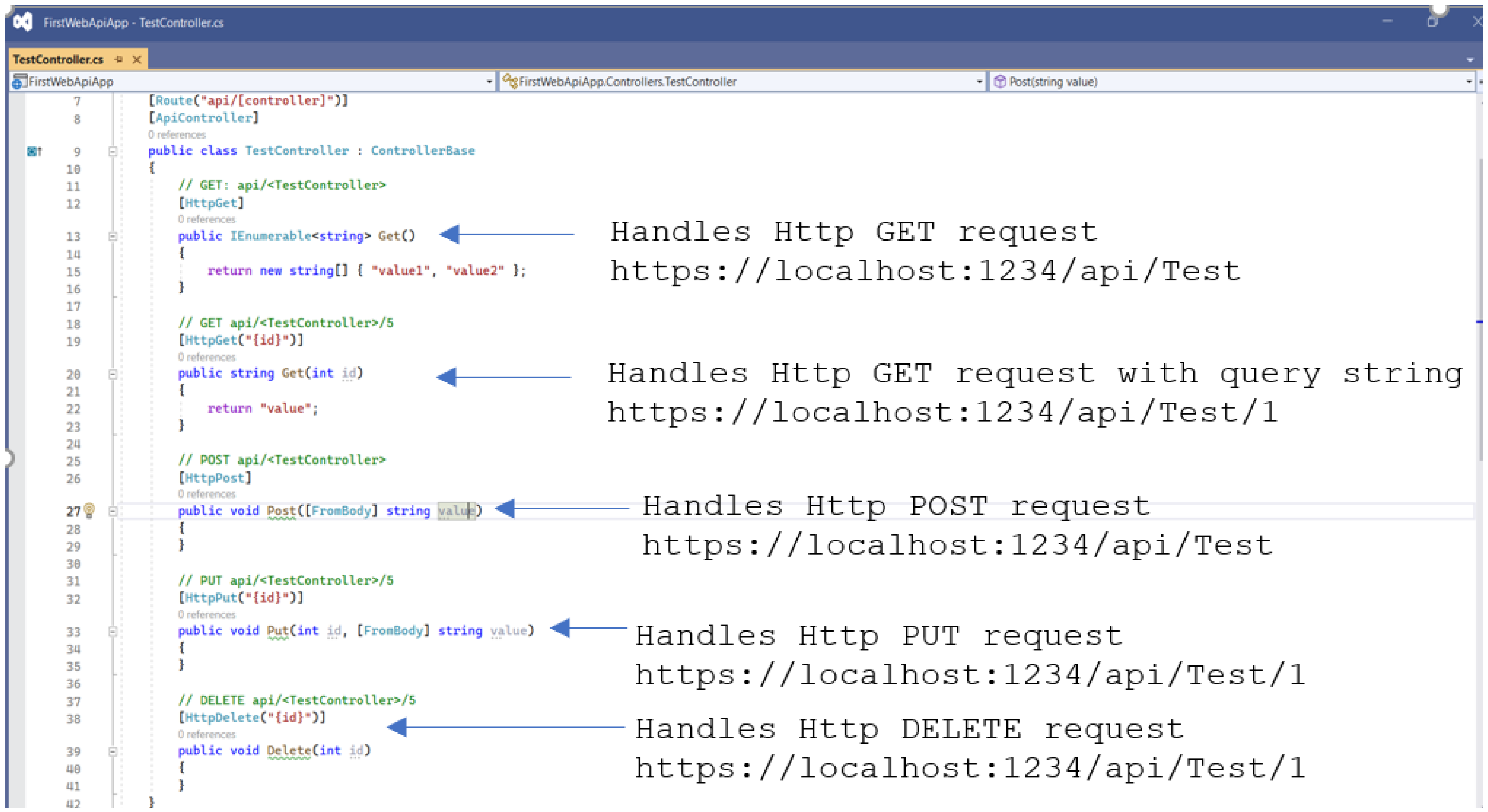
Anatomy of a Controller

- A controller class is typically derived from the ControllerBase class.
- It can have one or more action methods, which are public methods with specific attributes defining their HTTP verb and route.
- These action methods are invoked when a matching route is requested by the client.
- Controllers can also have constructor injection for services and dependencies.

Routing and Actions

- ASP.NET Core API provides flexible routing mechanisms to map incoming requests to controllers and actions.
- Routes can be defined using attributes like [Route] and [HttpGet] on the controller and action methods.
- Actions can have parameters, which are automatically mapped from the request data, such as query string, route values, or request body.
- Actions return various types of results, such as ActionResult, OkResult, BadRequestResult, etc.

Example: Simple Web API Controller


FirstWebApiApp - TestController.cs
TestController.cs
FirstWebApiApp
FirstWebApiApp.Controllers.TestController
Post(string value)

```
[Route("api/[controller]")]
[ApiController]
0 references
public class TestController : ControllerBase
{
    // GET: api/<TestController>
    [HttpGet]
    0 references
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }

    // GET api/<TestController>/5
    [HttpGet("{id}")]
    0 references
    public string Get(int id)
    {
        return "value";
    }

    // POST api/<TestController>
    [HttpPost]
    0 references
    public void Post([FromBody] string value)
    {
    }

    // PUT api/<TestController>/5
    [HttpPut("{id}")]
    0 references
    public void Put(int id, [FromBody] string value)
    {
    }

    // DELETE api/<TestController>/5
    [HttpDelete("{id}")]
    0 references
    public void Delete(int id)
    {
    }
}
```

Handles Http GET request
https://localhost:1234/api/Test

Handles Http GET request with query string
https://localhost:1234/api/Test/1

Handles Http POST request
https://localhost:1234/api/Test

Handles Http PUT request
https://localhost:1234/api/Test/1

Handles Http DELETE request
https://localhost:1234/api/Test/1

Action Method Naming Conventions

HTTP Method	Possible Web API Action Method Name	Usage

GET	Get()
	get()
	GET()
	GetAllStudent()
	*any name starting with Get *
		Retrieves data.

POST	Post()
	post()
	POST()
	PostNewStudent()
	any name starting with Post
		Inserts new record.

PUT	Put()
	put()
	PUT()
	PutStudent()
	any name starting with Put
		Updates existing record.

PATCH	Patch()
	patch()
	PATCH()
	PatchStudent()
	any name starting with Patch
		Updates record partially.

DELETE	Delete()
	delete()
	DELETE()
	DeleteStudent()
	any name starting with Delete
		Deletes record.

Lesson Summary

❖*Understand the role of a controller in a Web API.*

❖*Learn how to create and configure a controller in ASP.NET Core.*

❖*Explore the different types of actions that can be defined in a Web API controller.*

❖*Understand how to handle HTTP requests and route them to the appropriate controller actions.*

Section 3

Parameter Binding in ASP.NET Web API

Parameter Binding in ASP.NET Web API

- Welcome to the Parameter Binding in ASP.NET Web API.
- In this presentation, we will explore the concept of parameter binding in ASP.NET Web API and how it enables seamless integration of client requests with server-side actions.
- Let's dive into the details of parameter binding and its various techniques.

What is Parameter Binding?

- Parameter binding is the process of mapping incoming request data to parameters in Web API actions.
- Web API automatically binds the request data to action parameters based on the parameter's type, name, and attributes.
- By leveraging parameter binding, developers can easily access and utilize client data sent in the request.

Route Data Binding

- Route data binding is a type of parameter binding that extracts values from the route template and maps them to action parameters.
- Web API matches the route template with the incoming request URL and binds the corresponding values to the parameters.
- This type of binding is useful when you want to include route parameters in the URL itself.

Query String Binding

- Query string binding is another type of parameter binding that extracts values from the query string and maps them to action parameters.
- Web API automatically parses the query string parameters and binds them to the corresponding action parameters.
- Query string binding is commonly used when you want to pass optional or additional parameters to the server.

Request Body Binding

- Request body binding allows you to bind complex data sent in the request body to action parameters.
- Web API automatically deserializes the request body content based on the specified parameter type.
- This type of binding is commonly used when you want to send structured data, such as JSON or XML, to the server.

Form Data Binding

- Form data binding enables binding of HTML form data sent in the request to action parameters.
- Web API automatically maps form data fields to the corresponding action parameters.
- This type of binding is useful when you want to handle form submissions or upload files.

Lesson Summary

➢*Understand the concept of parameter binding in ASP.NET Web API.*

➢*Learn about the different types of parameter binding, such as model binding, query string binding, and route parameter binding, Request Body Binding*

Section 4

Controller Action Return Type

Controller Action Return Type in ASP.NET Web API

- Welcome to the Controller Action Return Type in ASP.NET Web API slide deck.
- In this presentation, we will explore the different return types available for controller actions in ASP.NET Web API.
- Understanding the appropriate return type is crucial for building robust and efficient APIs.
- Let's delve into the details of the controller action return types and their usage.

IActionResult Interface

- The IActionResult interface is the most flexible return type for controller actions in ASP.NET Web API.
- It represents an HTTP response with a status code, headers, and optionally a response body.
- By returning an IActionResult, you can customize the response based on various conditions and requirements.

ActionResult<T> Generic Type

- The ActionResult<T> generic type is a specialized version of IActionResult that allows you to return a specific type as the response body.
- By specifying the desired type as the generic parameter, you can benefit from automatic serialization and content negotiation.

Specific Return Types

- ASP.NET Web API also supports specific return types for common scenarios.
- Some examples include:
  - ➢ **OkResult**: Returns an empty 200 OK response.
  - ➢ **BadRequestResult**: Returns a 400 Bad Request response.
  - ➢ **NotFoundResult**: Returns a 404 Not Found response.
  - ➢ **UnauthorizedResult**: Returns a 401 Unauthorized response.

HttpResponseMessage

- In certain cases, you may need fine-grained control over the HTTP response.
- The HttpResponseMessage class allows you to manually create and customize the entire response, including status code, headers, and content.

IActionResult vs Specific Return Types

- When choosing between IActionResult and specific return types, consider the level of flexibility and customization required.
- IActionResult provides more control and flexibility for handling various scenarios, while specific return types offer simplicity for common responses.

Lesson Summary

- *Learn about the different return types that can be used in controller actions in ASP.NET Core Web API.*
- *Understand the conventions for returning data, such as IActionResult, HttpResponseMessage, and specific data types.*
- *Explore how to handle different scenarios, such as returning JSON, XML, or other content types from a Web API.*

THANK YOU!