**Objectives:**

* Explain the concept of RESTful web service, Web API & Microservice
  + Features of REST architecture - Representational State Transfer, Stateless, Messages, Concept of Microservice, Difference between WebService & WebAPI, Not restricted to send XML as response
* Explain what is HttpRequest & HttpResponse
* List the types of Action Verbs
  + HttpGet, HttpPost, HttpPut, HttpDelete - Meaning of action verbs and how that should be declared as attributes for Web API
* List the types of HttpStatusCodes used in WebAPI
  + Ok, InternalServerError, Unauthorized, BadRequest - All thru the action result types
* Demonstrate creation of a simple WebAPI - With Read, Write actions
  + Structure of a web api - Controller & its inheritance from ApiController, Action verbs, Action method
* Explain the types of Configuration files of WebAPI
  + Startup.cs with depdency injection, appSettings.json, launchSettings.json, Explain Route.config & WebAPI.config in .Net 4.5 framework

RESTful Web Service, Web API & Microservice

A RESTful web service is a style used to create web services that communicate over HTTP. It is based on Representational State Transfer (REST), which focuses on stateless communication. Each client request must include all the information the server needs to understand and process it. REST uses standard HTTP verbs like GET, POST, PUT, and DELETE to operate on resources identified by URIs. Unlike traditional web services, REST does not limit responses to XML; it can also send data in JSON, plain text, or other formats.

A Web API is a framework for creating HTTP services that various clients, including browsers, desktop applications, and mobile devices, can access. It usually follows REST principles and is lightweight and flexible. A Microservice is a small, independently deployable service that focuses on a specific business function. Microservices often use Web APIs to communicate in a distributed system.

The main difference between a Web Service and a Web API is that web services, like SOAP-based .asmx services, use heavier protocols and are mostly limited to XML. Web APIs are more modern, using HTTP or HTTPS, and support multiple data formats like JSON, XML, and text. Web APIs are faster, platform-independent, and easier to develop and use.

HttpRequest & HttpResponse

In Web APIs, an HttpRequest is an object that contains all the information sent by a client to the server. This includes the request method (GET, POST, etc.), URL, headers, query parameters, and body content. It shows the client's intent or input to the server.

An HttpResponse represents the server’s reply to the client's request. It includes information such as the status code (e.g., 200 OK, 400 Bad Request), headers, and the response body, which may contain data or an error message. The request and response objects work together to enable communication between the client and server over HTTP.

Action Verbs in Web API

Action verbs in Web API are standard HTTP methods used to perform tasks on resources. The most commonly used verbs are HttpGet, HttpPost, HttpPut, and HttpDelete. These are marked in Web API with attributes placed above controller action methods.

[HttpGet] is used to retrieve data.

[HttpPost] is used to create new resources.

[HttpPut] is used to update existing resources.

[HttpDelete] is used to delete resources.

Each verb corresponds to a specific operation and helps keep the Web API clear, meaningful, and RESTful. They allow clients to interact with server-side data in a standard way.

HTTP Status Codes in Web API

HTTP status codes indicate the result of a client’s request in Web API. Some commonly used status codes are:

200 OK, The request succeeded, and the server returned the expected response.

400 BadRequest, The server could not understand the request due to bad syntax or missing parameters.

401 Unauthorized, The client must authenticate itself to receive the requested response.

500 InternalServerError, The server faced an unexpected issue that stopped it from fulfilling the request.

These codes are used with methods like return Ok(), return BadRequest(), return Unauthorized(), and return StatusCode(500) in the controller actions to give helpful feedback to the client.

Creating a Simple Web API (Read and Write Actions)

To create a simple Web API in .NET Core, open Visual Studio and start a new project with the "ASP.NET Core Web API" template. Uncheck the "Use minimal API" option to use controllers. After creating the project, go to the Controllers folder, right-click, and add a new controller with the "API Controller - Empty" template. Name the controller ValuesController.

In this controller, define a static list of strings and implement the four main action methods: Get to read all values, Post to add a value, Put to update a value at a specific index, and Delete to remove a value at a specific index. Each method should have the correct HTTP verb attribute ([HttpGet], [HttpPost], [HttpPut], [HttpDelete]). These methods should return proper HTTP responses using methods like Ok() or BadRequest().

Configuration Files in Web API

.NET Core Web APIs use various configuration files to manage the application's behavior. The Program.cs (or Startup.cs in older templates) file is used to set up services, middleware, and routing through dependency injection. This file is central to starting the application.

The appsettings.json file contains configuration data like connection strings, API keys, and logging settings. It's often used for environment-specific settings. The launchSettings.json file in the Properties folder defines how the application is launched during development, including the URL, ports, and environment variables.

In older versions of .NET, like .NET Framework 4.5, configuration was managed through Web.config, which included settings and routing. The RouteConfig.cs file was used to define routing patterns for Web API and MVC applications.