## ****RESTful Web Service, Web API & Microservice****

### RESTful Web Service:

* **REST (Representational State Transfer)** is an architectural style for designing networked applications.
* REST uses standard HTTP methods (GET, POST, PUT, DELETE) to perform CRUD operations on resources.
* **Resources** are identified using URIs, and representations (usually JSON or XML) are used for data exchange.

### Web API:

* A **Web API** is an interface that allows interaction between two software applications over the web using HTTP.
* In .NET, Web API is a framework for building RESTful services.
* It supports multiple formats like JSON, XML, and even plain text.
* Web APIs are lighter and faster than traditional web services (SOAP-based).

### Microservice:

* **Microservices** are an architectural approach where a single application is composed of multiple loosely coupled services.
* Each service focuses on a single business capability and communicates via REST APIs or messaging queues.
* **Key benefits:** scalability, ease of deployment, and technology flexibility.

## ****Features of REST Architecture****

| **Feature** | **Description** |
| --- | --- |
| **Stateless** | No client context is stored on the server between requests. Each request must contain all the information needed. |
| **Representational** | Resources are represented in formats like JSON or XML. |
| **Client-Server** | Clear separation between UI (client) and data storage (server). |
| **Cacheable** | Responses can be cached to improve performance. |
| **Uniform Interface** | Standardized URIs and HTTP methods. |
| **Layered System** | APIs can be composed of hierarchical layers. |

## ****Difference: WebService vs WebAPI****

| **Feature** | **Web Service** | **Web API** |
| --- | --- | --- |
| Protocol | Mostly SOAP (uses XML) | Uses HTTP (REST), supports JSON, XML |
| Speed | Slower (XML parsing) | Faster (JSON) |
| Flexibility | Less (XML only) | More (multiple formats) |
| Platform Dependent | Tied to .NET (ASMX/WCF) | Platform-agnostic |

## ****HttpRequest & HttpResponse****

### HttpRequest

* Sent by the client to initiate communication with the server.
* Contains **URL**, **method (GET/POST)**, **headers**, **body** (in case of POST/PUT).

### HttpResponse

* Sent by the server back to the client.
* Contains **status code**, **headers**, **body** (data or error message).

## ****Types of Action Verbs (HTTP Methods)****

| **Verb** | **Description** | **Attribute in WebAPI** |
| --- | --- | --- |
| **GET** | Retrieve resource | [HttpGet] |
| **POST** | Create new resource | [HttpPost] |
| **PUT** | Update existing resource | [HttpPut] |
| **DELETE** | Delete resource | [HttpDelete] |

These are declared above controller methods to define the type of operation being performed.

**Types of HTTP Status Codes in WebAPI**

| **Code** | **Meaning** | **ActionResult Equivalent** |
| --- | --- | --- |
| 200 OK | Request successful | return Ok(data); |
| 400 Bad Request | Invalid client request | return BadRequest("message"); |
| 401 Unauthorized | Authentication failed | return Unauthorized(); |
| 500 Internal Server Error | Server-side error | return StatusCode(500, "error"); |