**Handson 1: First Web Api using .Net core**

**Superset id: 6363535**

### **RESTful Web Service**

* **REST**: *Representational State Transfer* – An architectural style for designing networked applications.
* **Key Features**:  
  + **Stateless**: Each HTTP request contains all necessary information (no session).
  + **Resource-Based**: Uses URIs to access resources.
  + **Message-Based**: Communicates via HTTP messages (mostly JSON or XML).
  + **Uniform Interface**: Same approach for different resources.

### **Web API**

* A **framework** for building HTTP services using REST principles.
* Built on top of ASP.NET Core or .NET Framework.
* Can serve data in **JSON, XML**, or custom formats.
* Typically used in **web and mobile apps** for backend communication.

### **Microservice**

* An **independent, small, and self-contained service** that performs a specific business function.
* Multiple microservices together build a complete system.
* Communicate over REST APIs or messaging systems.

## **WebService vs WebAPI**

| **Feature** | **Web Service (.asmx)** | **Web API (.NET Core/Framework)** |
| --- | --- | --- |
| Protocol | SOAP | REST over HTTP |
| Data Format | XML only | JSON, XML, others |
| Hosting | IIS only | IIS, Kestrel, self-hosting |
| Lightweight | No | Yes |
| REST Support | Limited | Full REST support |

## **What is HttpRequest & HttpResponse?**

* **HttpRequest**: Represents the client's request to the server.  
  + Includes: URL, method (GET/POST etc.), headers, body, cookies, query parameters.
  + Accessed in Web API viaHttpContext.Response.
* **HttpResponse**: Represents the server's response back to the client.  
  + Includes: status code, headers, body (JSON/XML), cookies.
  + Accessed via HttpContext.Response or returned using action results like Ok(), Bad Request().

## **Types of Action Verbs in Web API**

### **Common HTTP Verbs:**

| **Verb** | **Purpose** | **Web API Attribute** |
| --- | --- | --- |
| HttpGet | Read data | [HttpGet] |
| HttpPost | Create new resource | [HttpPost] |
| HttpPut | Update entire resource | [HttpPut] |
| HttpDelete | Delete a resource | [HttpDelete] |

## 

## 

## **HttpStatusCodes in Web API**

| **Status Code** | **Meaning** | **ActionResult Usage** |
| --- | --- | --- |
| 200 OK | Request succeeded | return Ok(data); |
| 400 BadRequest | Bad input/request | return BadRequest("error"); |
| 401 Unauthorized | Auth required/invalid | return Unauthorized(); |
| 500 InternalServerError | Server crash | return StatusCode(500); |

**Handson 1: First Web Api using .Net core**

Create a .Net core web application with API template. Use the option to create controller with Read Write permissions. Notice the ValuesController creation with Action methods corresponding to the Action verbs.

On creation of the Web API, execute the application and check if the GET action method result is returned as expected.

[**ValuesController.cs**](http://valuescontroller.cs)

using Microsoft.AspNetCore.Mvc;

[ApiController]

[Route("[controller]")]

public class ValuesController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

return Ok(new string[] { "value1", "value2" });

}

[HttpGet("{id}")]

public IActionResult Get(int id)

{

return Ok("value" + id);

}

[HttpPost]

public IActionResult Post([FromBody] string value)

{

return Ok("Posted: " + value);

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] string value)

{

return Ok($"Updated {id} with {value}");

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

return Ok("Deleted " + id);

}

}



