**What is a RESTful Web Service?**

* REST (Representational State Transfer) is an architectural style for building web services.
* Operates over HTTP and uses stateless communication.
* RESTful APIs expose resources (data) via URLs and use HTTP verbs to perform actions.

**Features of REST Architecture**

| **Feature** | **Description** |
| --- | --- |
| **Stateless** | Each request contains all the information needed (no session memory on server). |
| **URI-based** | Resources are identified via URLs. |
| **Use of Verbs** | HTTP methods (GET, POST, PUT, DELETE) represent actions. |
| **Multiple formats** | REST can return JSON, XML, HTML, etc. (not limited to XML like SOAP). |
| **Layered architecture** | Client doesn’t care where the service is hosted. |

**Web Service vs Web API**

| **Feature** | **Web Service (.asmx / WCF)** | **Web API (REST)** |
| --- | --- | --- |
| Protocol | SOAP | HTTP/HTTPS |
| Format | XML | JSON, XML, etc |
| Platform Dependency | .NET Framework only | Cross-platform |
| Lightweight | ❌ | ✅ |
| REST Support | Limited | Native |

**What is a Microservice?**

* A Microservice is a small, independent, deployable unit focused on a single responsibility.
* Communicates with other services using HTTP, gRPC, or messaging (e.g., RabbitMQ).
* Built using REST APIs, making Web API a building block for microservices.

**HttpRequest & HttpResponse**

**HttpRequest**

Represents the incoming request from a client.

**Contains:**

* URL
* Headers
* Query string
* Body (e.g., JSON payload)
* Method (GET, POST...)

**HttpResponse**

Represents the outgoing response from the server.

**Contains:**

* Status code (200 OK, 404 Not Found)
* Headers
* Response body (e.g., JSON or string)

**HTTP Action Verbs in Web API**

| **Verb** | **Purpose** | **Usage in Controller** |
| --- | --- | --- |
| HttpGet | Read or retrieve a resource | [HttpGet] |
| HttpPost | Create a new resource | [HttpPost] |
| HttpPut | Update an existing resource | [HttpPut("{id}")] |
| HttpDelete | Delete a resource | [HttpDelete("{id}")] |

**Common HTTP Status Codes**

| **Code** | **Name** | **Usage in Web API** |
| --- | --- | --- |
| 200 | OK | return Ok(data); |
| 201 | Created | return Created(...); |
| 400 | BadRequest | return BadRequest(msg); |
| 401 | Unauthorized | return Unauthorized(); |
| 404 | NotFound | return NotFound(); |
| 500 | InternalServerError | return StatusCode(500); |

**Structure of a Simple Web API**

**Components:**

* Program.cs: Entry point
* Startup.cs (if using .NET 5 or earlier): Configures services and middleware
* Controllers/: Contains API controllers
* Models/: Holds data classes

**Configuration Files in Web API (ASP.NET Core)**

| **File** | **Purpose** |
| --- | --- |
| Program.cs | Bootstraps and runs the app (top-level configuration). |
| Startup.cs | (Optional in .NET 6+) Configures services (DI) and middleware. |
| appsettings.json | Holds config values like connection strings, JWT keys. |
| launchSettings.json | Used only by Visual Studio to configure the environment, ports, profiles for debug. |

**In .NET Framework (Web API 2)**

| **File** | **Purpose** |
| --- | --- |
| Web.config | Central config file (like appsettings). |
| WebApiConfig.cs | Route setup using config.Routes. |
| Global.asax | App startup events. |

**CODE:**

**ValuesController.cs**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System.Linq;

namespace FirstWebApi.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class ValuesController : ControllerBase

{

private static readonly List<string> \_values = new()

{

"Apple",

"Banana",

"Cherry",

"Dragonfruit",

"Elderberry"

};

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

{

return Ok(\_values);

}

[HttpGet("{id}")]

public ActionResult<string> Get(int id)

{

if (id < 0 || id >= \_values.Count)

return NotFound($"Value at index {id} not found.");

return Ok(\_values[id]);

}

}

}

Check using Swagger





