**WEEK – 4 Assignments**

**Mandatory hands-on :-**

**1. WebApi\_Handson :**

**Objectives:**

· Explain the concept of RESTful web service, Web API & Microservice

o 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

o 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

o Ok, InternalServerError, Unauthorized, BadRequest - All thru the action result types

· Demonstrate creation of a simple WebAPI - With Read, Write actions

o Structure of a web api - Controller & its inheritance from ApiController, Action verbs, Action method

· Explain the types of Configuration files of WebAPI

o Startup.cs with depdency injection, appSettings.json, launchSettings.json, Explain Route.config & WebAPI.config in .Net 4.5 framework

**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.

**Answers :**

## RESTful Web Service :

* REST (Representational State Transfer): An architectural style for web services using standard HTTP methods.
* Features:
  + Stateless: Each request is independent; server does not store client state.
  + Messages: Data exchanged via HTTP (GET, POST, etc.), typically using JSON or XML.
  + Not restricted to XML: Can use JSON, XML, or other formats.
  + Uniform Interface: Standardized access to resources via URIs and HTTP verbs.

## Web API :

* Web API: Interface for applications to communicate over HTTP.
* Difference vs. WebService:
  + WebService (like SOAP) often uses only XML and is heavier.
  + Web API is lighter, supports multiple formats (JSON, XML), and is more flexible.

## Microservice :

* Microservice: Small, independent service focused on a specific business function, communicating over lightweight protocols (often REST/HTTP).

## HttpRequest & HttpResponse :

* HttpRequest: Represents what the client sends to the server (method, URL, headers, body).
* HttpResponse: What the server sends back (status code, headers, body with data or error).

Types of Action Verbs :

The main HTTP action verbs used in Web APIs are GET, POST, PUT, and DELETE. GET is used to read or retrieve data, POST is used to create new data, PUT is used to update existing data, and DELETE is used to remove data. In C# Web API, these are declared as attributes above the controller methods, such as [HttpGet] , [HttpPost] , [HttpPut] and [HttpDelete] .

Types of HttpStatusCodes Used in WebAPI :

Common HTTP status codes in Web API include 200 OK for successful requests, 400 BadRequest for invalid client requests, 401 Unauthorized when authentication is required, and 500 InternalServerError for server-side errors. In C#, these are returned using methods like ok () , BadRequest () , Unauthorized () and StatusCode (500) .

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

[**Valueltem.cs**](http://valueltem.cs) **:**

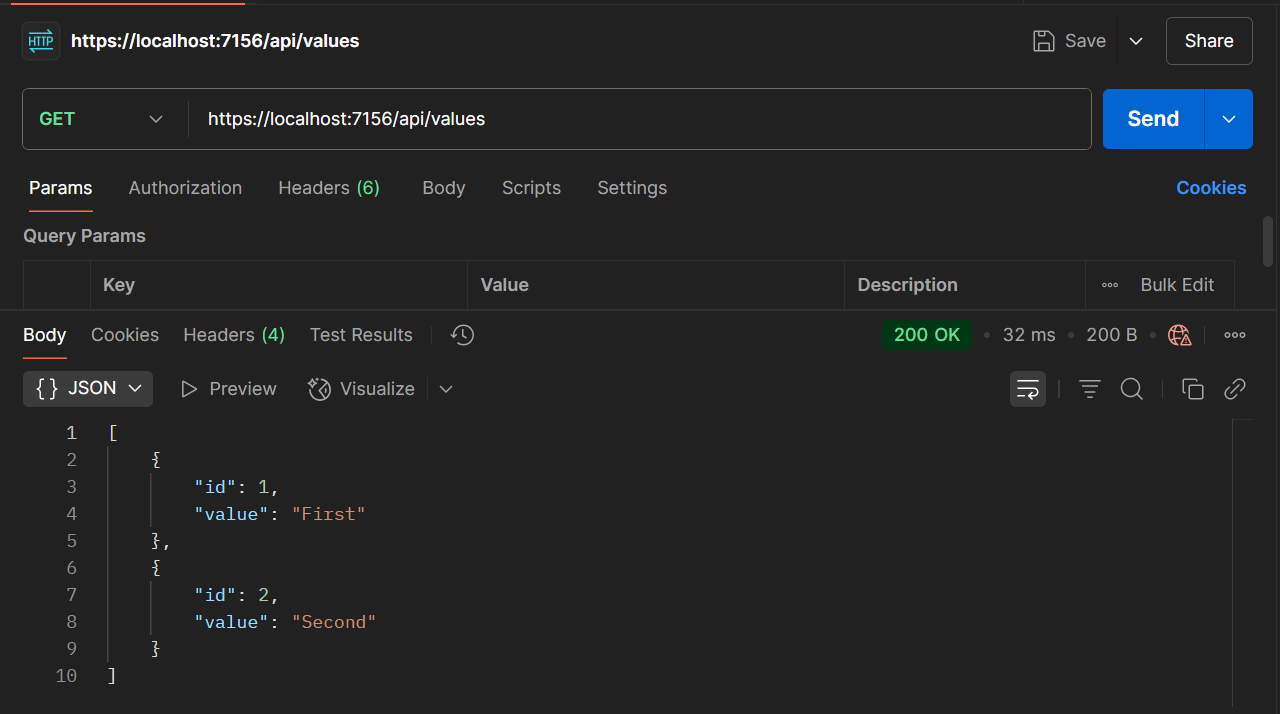
|  |
| --- |
| **namespace MyFirstWebAPI.Models {  public class ValueItem  {  public int Id { get; set; }  public string Value { get; set; }  } }** |

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

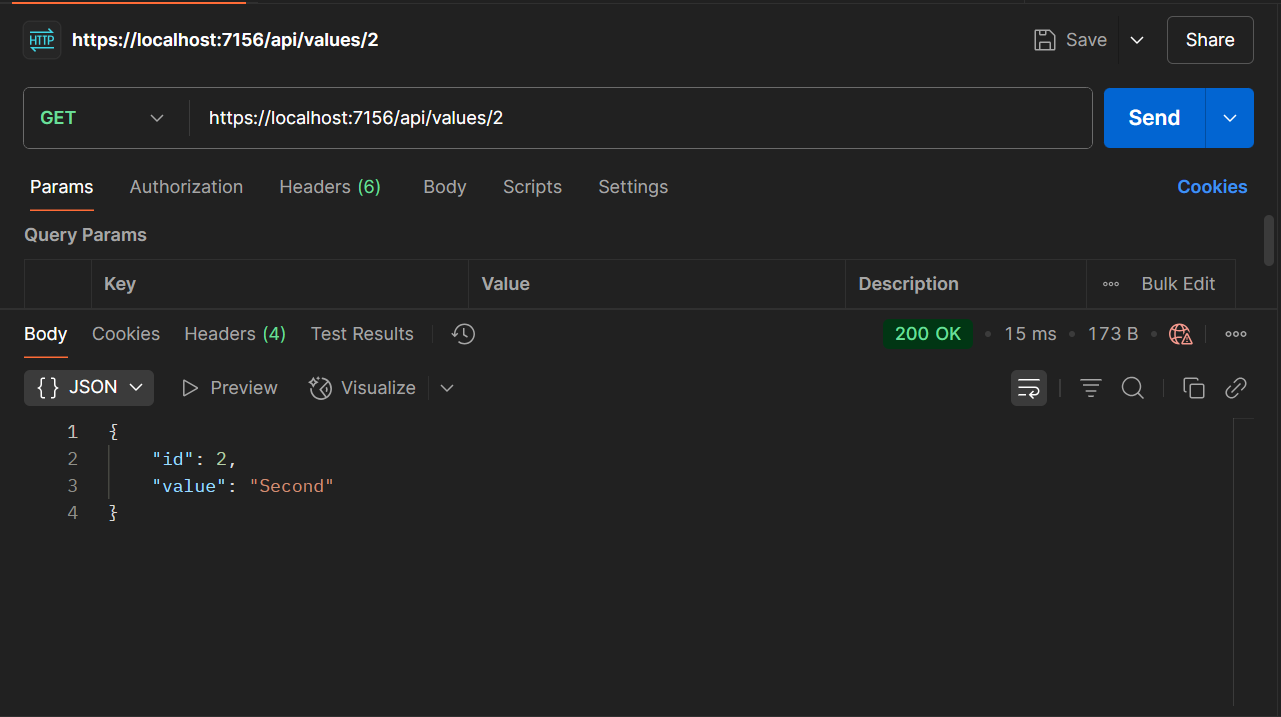
|  |
| --- |
| **using Microsoft.AspNetCore.Mvc; using MyFirstWebAPI.Models; using System.Collections.Generic; using System.Linq;  namespace MyFirstWebAPI.Controllers {  [ApiController]  [Route("api/[controller]")]  public class ValuesController : ControllerBase  {  // database  private static List<ValueItem> values = new List<ValueItem>  {  new ValueItem { Id = 1, Value = "First" },  new ValueItem { Id = 2, Value = "Second" }  };   // GET: api/values  [HttpGet]  public ActionResult<IEnumerable<ValueItem>> Get()  {  return Ok(values);  }   // GET: api/values/1  [HttpGet("{id}")]  public ActionResult<ValueItem> Get(int id)  {  var item = values.FirstOrDefault(v => v.Id == id);  if (item == null)  return NotFound();  return Ok(item);  }   } }** |

**OUTPUT :**

**GET : api/values :**

****

**GET : api/values/id :**

****