WEEK\_4-Handson

Rajesh A

6384182

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lab 2: Swagger, Postman & Route Customization

# Objectives

- Install and configure Swagger in Web API project  
- Use ProducesResponseType for response documentation  
- Add SwaggerGen and UseSwaggerUI in Program.cs  
- Test APIs using Postman tool  
- Understand structure of Postman requests (Headers, Body, Auth)  
- Modify route using [Route] attribute  
- Understand ActionName for having multiple methods with same verb

# Step 1: Create or Reuse Existing Project

Use the project from Lab 1 or create a new one.

# Step 2: Add EmployeeController.cs

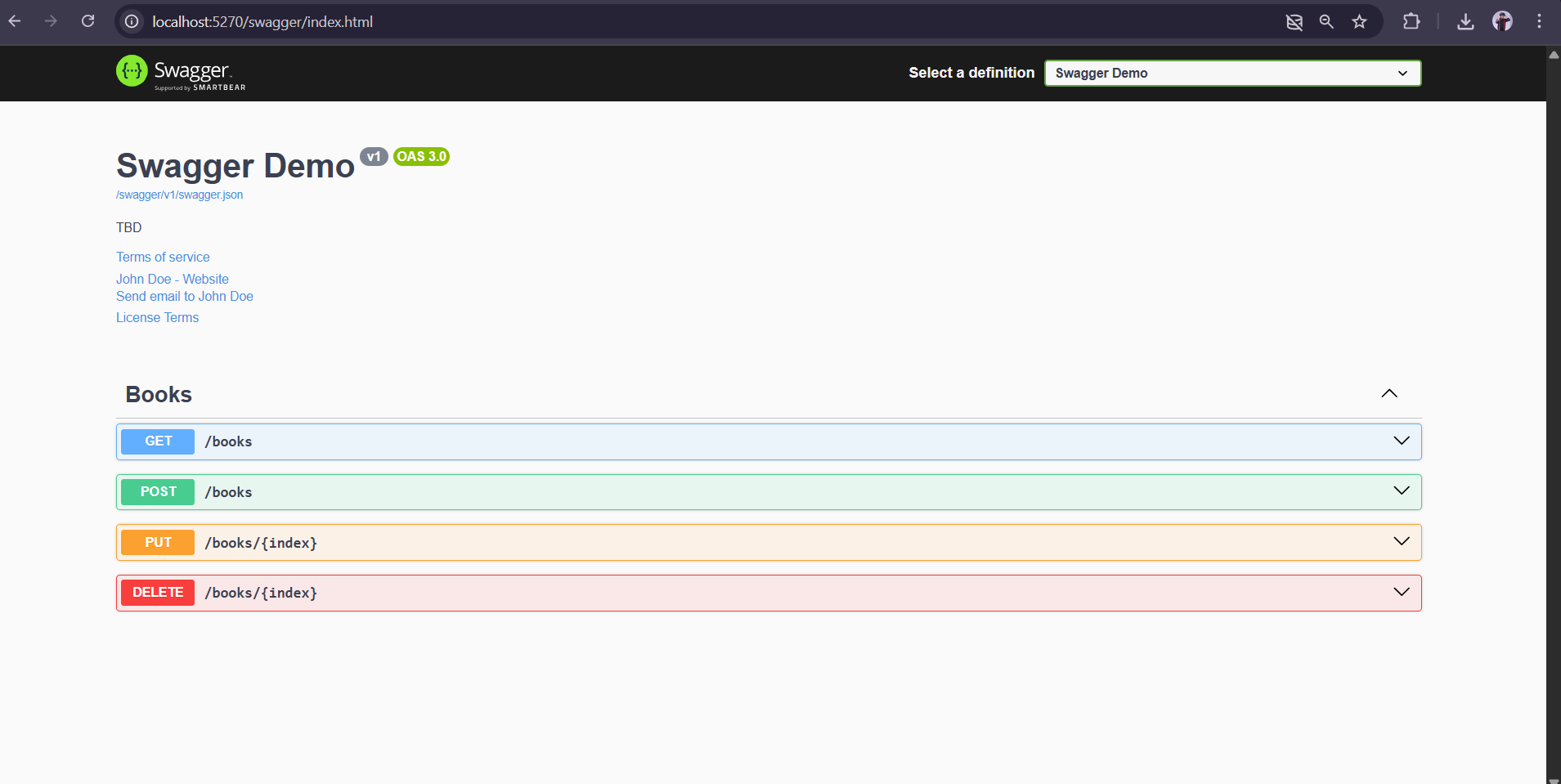
using Microsoft.AspNetCore.Mvc;  
  
namespace FirstWebApi.Controllers;  
  
[ApiController]  
[Route("emp")]  
public class EmployeeController : ControllerBase  
{  
 static List<string> employees = new() { "Alice", "Bob", "Charlie" };  
  
 [HttpGet]  
 public ActionResult<IEnumerable<string>> GetEmployees() => Ok(employees);  
  
 [HttpPost]  
 public ActionResult AddEmployee([FromBody] string emp)  
 {  
 employees.Add(emp);  
 return Ok("Employee added");  
 }  
  
 [HttpPut("{index}")]  
 public ActionResult UpdateEmployee(int index, [FromBody] string name)  
 {  
 if (index < 0 || index >= employees.Count) return BadRequest("Invalid index");  
 employees[index] = name;  
 return Ok("Employee updated");  
 }  
  
 [HttpDelete("{index}")]  
 public ActionResult DeleteEmployee(int index)  
 {  
 if (index < 0 || index >= employees.Count) return NotFound("Invalid index");  
 employees.RemoveAt(index);  
 return Ok("Employee deleted");  
 }  
}

# Step 3: Install Swagger

dotnet add package Swashbuckle.AspNetCore

# Step 4: Add Swagger in Program.cs

builder.Services.AddSwaggerGen(c =>  
{  
 c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo  
 {  
 Title = "Swagger Demo",  
 Version = "v1",  
 Description = "TBD",  
 TermsOfService = new Uri("https://example.com/terms"),  
 Contact = new Microsoft.OpenApi.Models.OpenApiContact  
 {  
 Name = "John Doe",  
 Email = "john@xyzmail.com",  
 Url = new Uri("https://www.example.com")  
 },  
 License = new Microsoft.OpenApi.Models.OpenApiLicense  
 {  
 Name = "License Terms",  
 Url = new Uri("https://www.example.com")  
 }  
 });  
});  
  
app.UseSwagger();  
app.UseSwaggerUI(c =>  
{  
 c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");  
});



# Step 5: Test in Postman

Open Postman and send GET to https://localhost:PORT/emp

# 

# Theory and Concepts

Swagger:

Swagger is a tool for documenting and testing APIs. It auto-generates UI to interact with your API.  
- Swashbuckle.AspNetCore is the NuGet package for Swagger in .NET Core.  
- ProducesResponseType can be used to describe response codes.  
- Swagger UI is accessible at /swagger.

Postman Tool:

Postman is an API testing tool. It allows you to:  
- Choose HTTP verbs (GET, POST, etc.)  
- Set headers (Content-Type, Authorization)  
- Send body (JSON)  
- View response status and body  
- Organize requests into collections

Route Customization in WebAPI:

- [Route("emp")]: Sets custom URL path.  
- [ActionName("GetEmployeesList")]: Allows multiple methods with same verb but different names.  
- Helps in making APIs more readable and user-friendly.

Thank you