WebApi\_Handson\_1:

### **RESTful Web Services, Web API & Microservices**

#### **Concept of RESTful Web Services**

* **REST (Representational State Transfer)** is an architectural style for building web services using standard HTTP methods.
* It relies on:  
  + **Statelessness**: Each HTTP request from a client contains all the info needed to process it.
  + **Resources**: Everything is treated as a resource identified by URIs.
  + **Representation**: Resources can be represented in various formats like JSON, XML, etc.
  + **Standard HTTP Methods**: GET, POST, PUT, DELETE, etc.

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

* **Stateless**: No client context is stored on the server.
* **Uniform Interface**: Resources are accessed via standard URIs and HTTP methods.
* **Cacheable**: Responses can be cached to improve performance.
* **Layered System**: Client cannot tell if it is connected to an intermediary or the end server.
* **Multiple Formats**: Supports JSON, XML, plain text, etc. (not limited to XML).

#### **Web API**

* A **Web API** is a framework that makes it easy to build HTTP services that reach a broad range of clients.
* Built on top of REST principles.
* Can return data in multiple formats (JSON, XML).
* Typically used in ASP.NET Core and ASP.NET Web API.

#### **Microservices**

* An architectural style where the application is broken into small, independently deployable services.
* Each microservice:  
  + Has its own DB and logic
  + Communicates via lightweight protocols (HTTP/REST)
  + Focuses on a single business capability

#### **WebService vs WebAPI**

| **Feature** | **Web Service (ASMX/WCF)** | **Web API (ASP.NET Core)** |
| --- | --- | --- |
| Protocol | SOAP | HTTP/REST |
| Response Format | XML only | JSON, XML, etc. |
| Hosting | IIS only | Self/IIS/Kestrel |
| Lightweight | No | Yes |
| Best for | Enterprise SOAP services | RESTful APIs |

### **HTTP Request & Response**

* **HttpRequest**: Represents the incoming request from a client.  
  + Contains method, headers, URI, body.
* **HttpResponse**: Represents the outgoing response to the client.  
  + Contains status code, headers, response body.

### **Action Verbs in Web API**

* **HttpGet**: Retrieves data
* **HttpPost**: Creates a new resource
* **HttpPut**: Updates an existing resource
* **HttpDelete**: Deletes a resource

In ASP.NET Core Web API:

[HttpGet] public IActionResult GetItem() { ... }

[HttpPost] public IActionResult CreateItem(Item item) { ... }

[HttpPut] public IActionResult UpdateItem(int id, Item item) { ... }

[HttpDelete] public IActionResult DeleteItem(int id) { ... }

### **Common HTTP Status Codes in Web API**

| **Code** | **Description** | **Usage** |
| --- | --- | --- |
| 200 | OK | Successful request |
| 400 | BadRequest | Invalid input |
| 401 | Unauthorized | Authentication required |
| 500 | InternalServerError | Server-side error |

Return in ASP.NET Core:

return Ok(result);

return BadRequest("Invalid input");

return Unauthorized();

return StatusCode(500, "Something went wrong");

### **Simple Web API Demo (Read, Write)**

**Controller Example:**

[ApiController]

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

public class ProductsController : ControllerBase

{

private static List<string> products = new() { "Pen", "Notebook" };

[HttpGet]

public IActionResult Get() => Ok(products);

[HttpPost]

public IActionResult Post([FromBody] string item)

{

products.Add(item);

return Ok(products);

}

}

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

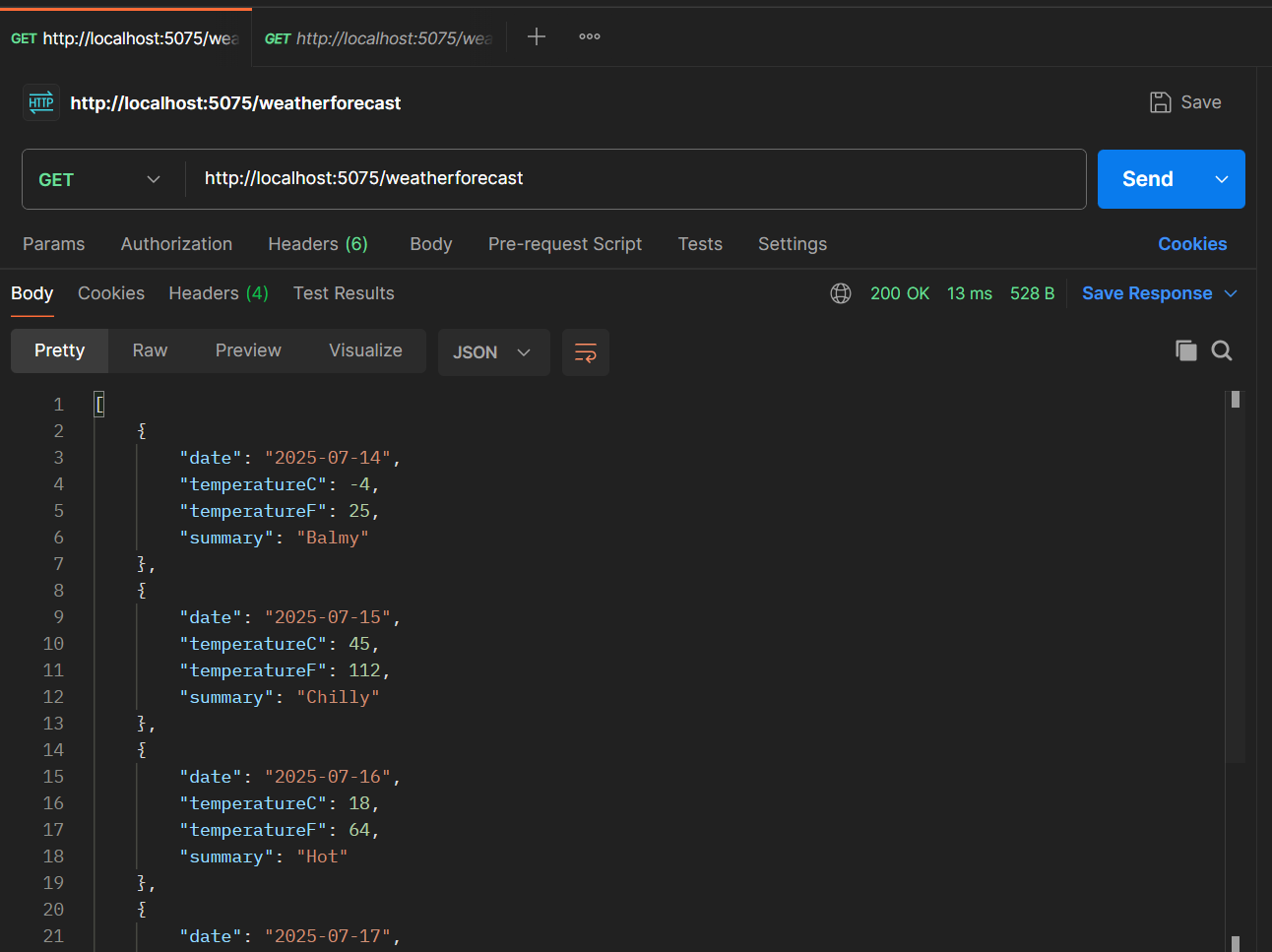
* **Startup.cs**
  + Handles dependency injection, middleware config
  + Includes ConfigureServices() and Configure() methods
* **appsettings.json**
  + Key-value config (e.g., connection strings, app-level settings)
* **launchSettings.json**
  + Controls environment (e.g., Development), URL, profiles for launching

### **Configuration in .NET Framework (4.5)**

* **Web.config**: App settings, connection strings, etc.
* **Route.config**: Optional, used to configure custom route logic for WebForms or MVC
* **WebApi.config**: Used to configure routes and formatters specifically for Web API

1. **First Web API**

**Output:**

****

WebApi\_Handson\_2:

1. **Web Api using .NET core with Swagger:**

Code:

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://example.com/terms")

},

License=new Microsoft.OpenApi.Models.OpenApiLicense

{

Name="License",

Url=new Uri("https://example.com/terms")

}

});

});

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

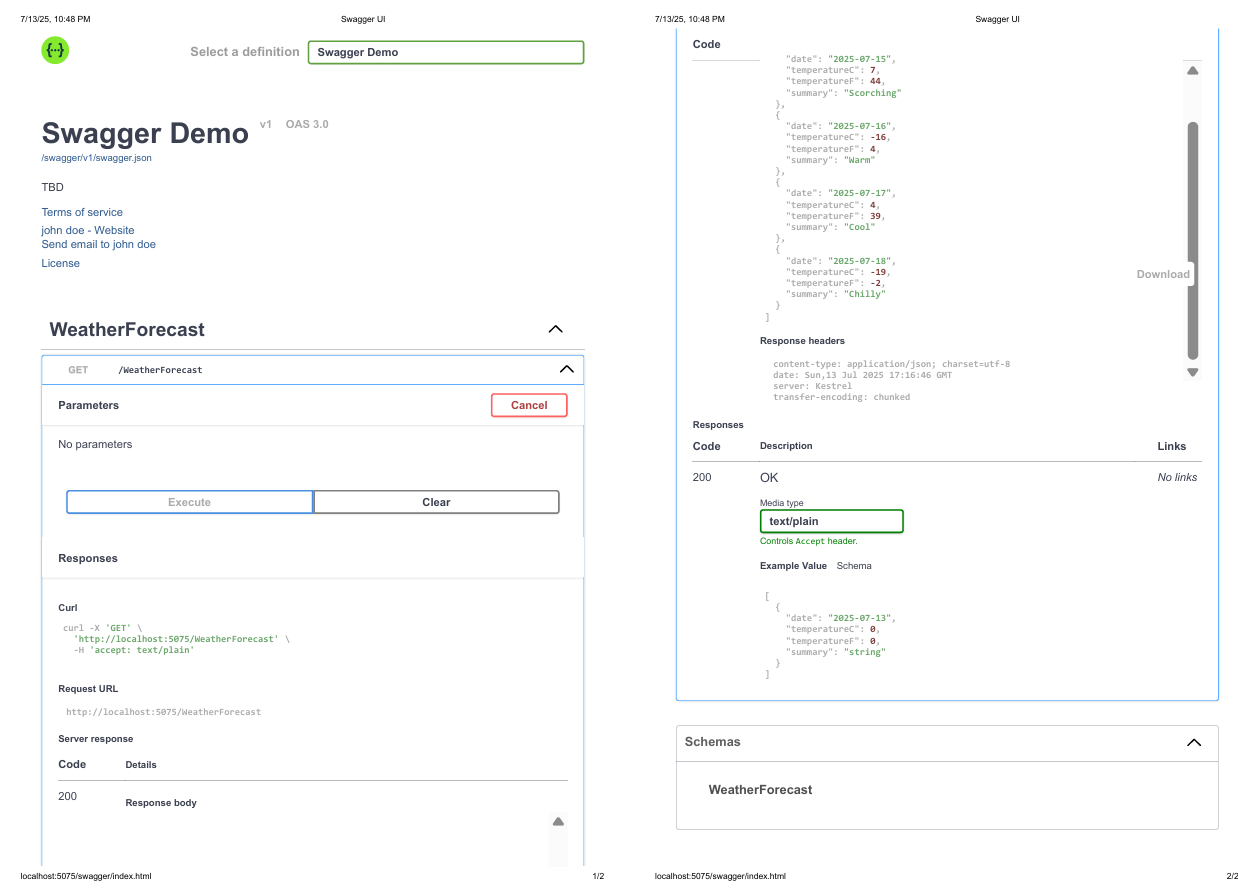
app.UseSwaggerUI(c =>

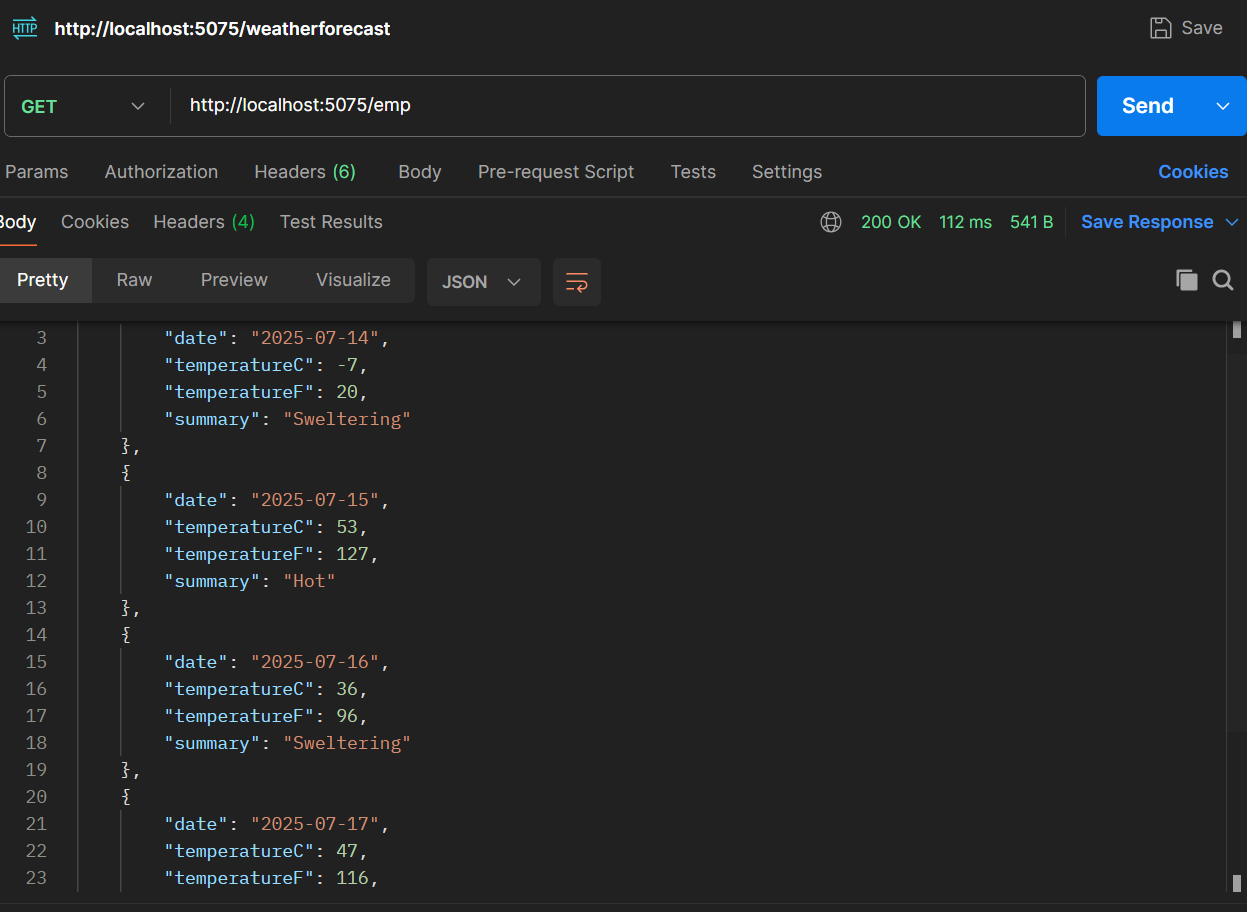
{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");

});

}

Output:



WebApi\_Handson\_3:

**1.Web Api using custom model class**

Code:

using System;

using System.Collections.Generic;

namespace webhandson1.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

--------------------------------------------------------

using System;

using System.Collections.Generic;

namespace webhandson1.Models

{

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public int Salary { get; set; }

public bool Permanent { get; set; }

public Department Department { get; set; }

public List<Skill> Skills { get; set; }

public DateTime DateofBirth { get; set; }

}

}

------------------------------------------------------------

using System;

using System.Collections.Generic;

namespace webhandson1.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

--------------------------------------------------------------

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using webhandson1.Filters;

using webhandson1.Models;

namespace webhandson1.Controllers

{

[ApiController]

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

[AllowAnonymous]

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private static List<Employee> employees = new();

public EmployeeController()

{

if (employees.Count == 0)

{

employees = GetStandardEmployeeList();

}

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "john",

Salary = 6000,

Permanent = true,

DateofBirth = new DateTime(1990, 5, 1),

Department = new Department { Id = 101, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "Python" }

}

}

};

}

[HttpGet("GetStandard")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> GetStandard()

{

throw new Exception("Something went wrong");

//return Ok(employees);

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status200OK)]

public IActionResult Post([FromBody] Employee emp)

{

employees.Add(emp);

return Ok(emp);

}

}

}

--------------------------------------------------------------

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace webhandson1.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var headers = context.HttpContext.Request.Headers;

if (!headers.ContainsKey("Authorization"))

{

context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

return;

}

var token = headers["Authorization"].ToString();

if (!token.Contains("Bearer"))

{

context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

}

}

}

}

---------------------------------------------------------

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

namespace webhandson1.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var error = context.Exception;

var logPath = "logs/errorlog.txt";

Directory.CreateDirectory("logs");

File.AppendAllText(logPath, $"[{DateTime.Now}] {error.Message}\n");

context.Result = new ObjectResult("Internal Server Error")

{

StatusCode = 500

};

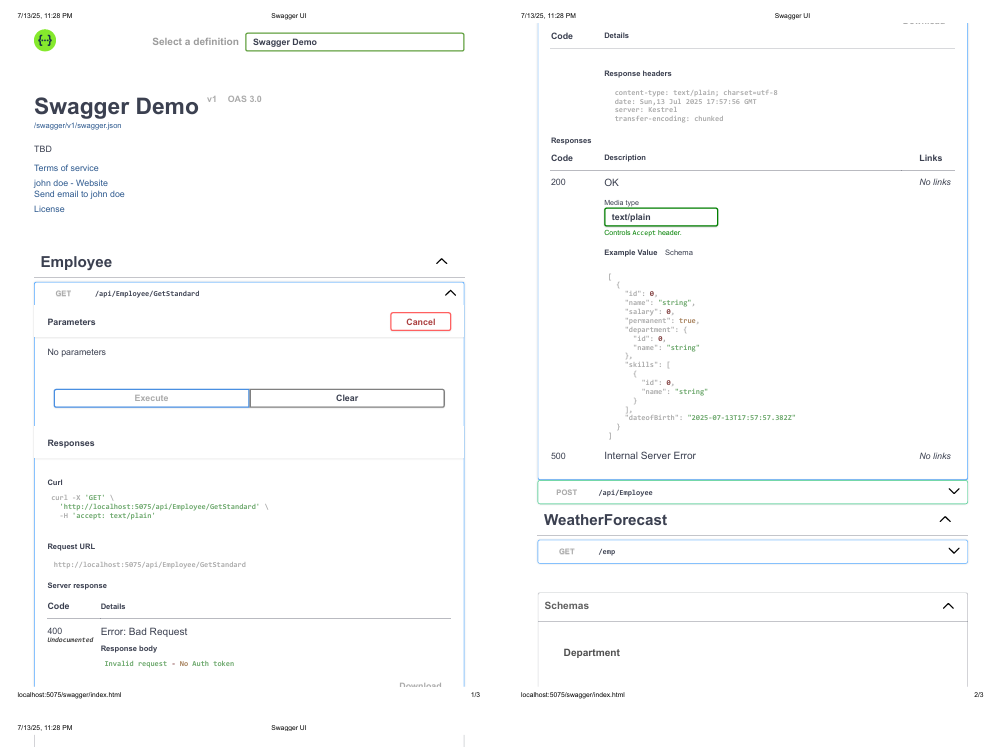
}

}

}

----------------------------------------------------------------------------

Output:



WebApi\_Handson\_4:

1. **Web Api CRUD Operation:**

Code:

-----------------------------------------------------------------------------------------------------

[HttpPut("{id}")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updatedEmp)

{

if (id <= 0)

return BadRequest("Invalid employee id");

var existingEmp = employees.FirstOrDefault(e => e.Id == id);

if (existingEmp == null)

return BadRequest("Invalid employee id");

// Update fields

existingEmp.Name = updatedEmp.Name;

existingEmp.Salary = updatedEmp.Salary;

existingEmp.Permanent = updatedEmp.Permanent;

existingEmp.DateofBirth = updatedEmp.DateofBirth;

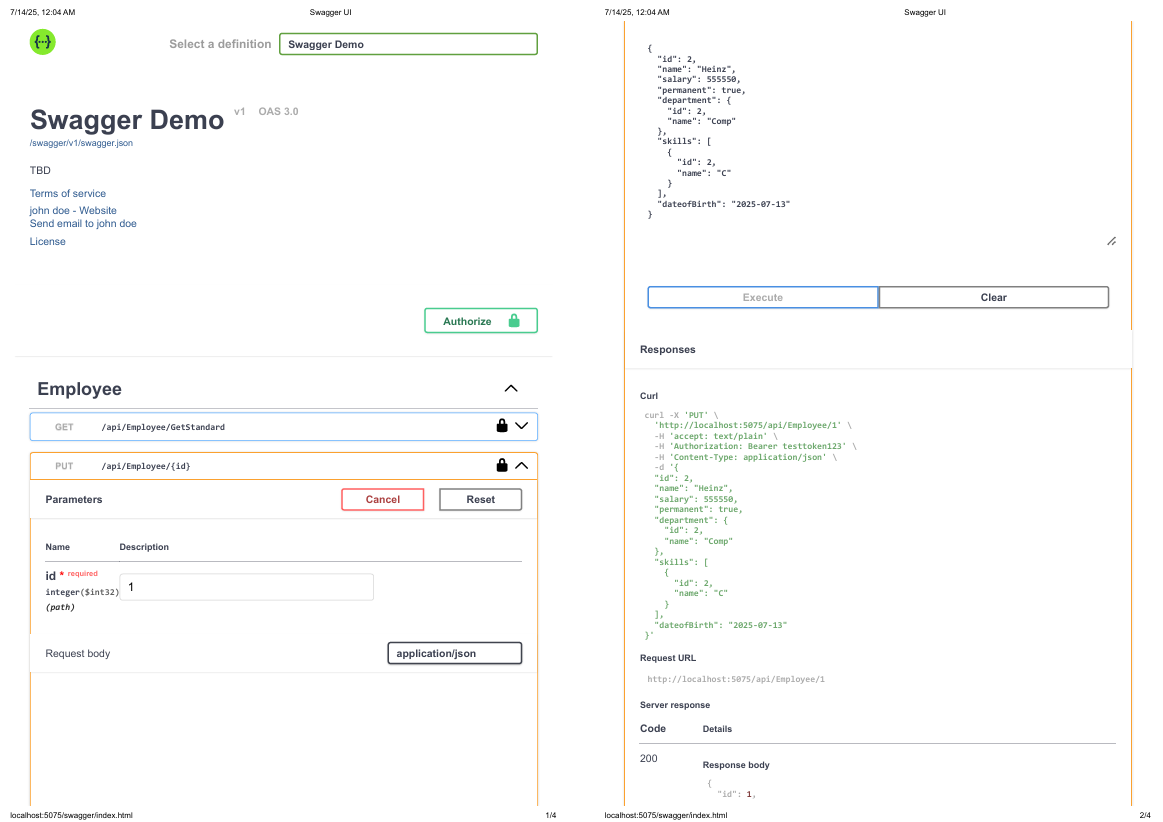
existingEmp.Department = updatedEmp.Department;

existingEmp.Skills = updatedEmp.Skills;

return Ok(existingEmp);

}

----------------------------------------------------------------------------------------------------------

Output:

WebApi\_Handson\_5:

1. **Json WebToken:**

**Code:**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using Microsoft.OpenApi.Models;

using webhandson1.Filters;

var builder = WebApplication.CreateBuilder(args);

var securityKey = "mysuperdupersecret";

var symmetricKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));

// Add services to the container.

builder.Services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(x =>

{

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricKey

};

});

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

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://example.com/terms")

},

License=new Microsoft.OpenApi.Models.OpenApiLicense

{

Name="License",

Url=new Uri("https://example.com/terms")

}

});

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

In = ParameterLocation.Header,

Description = "Please enter a valid token (e.g., Bearer mytoken123)",

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");

});

}

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

---------------------------------------------------------------------------------------------------------

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace webhandson1.Controllers

{

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(1, "Admin"); // Set role here

return Ok(new { token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecret"));

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

var claims = new List<Claim>

{

new Claim(ClaimTypes.Role, userRole),

new Claim("UserId", userId.ToString())

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(2), // ⏱ Set to 2 minutes for expiration test

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

------------------------------------------------------------------------------------------------

Ouptut: