**ASP.NET CORE 8.0 WEB API**

1. **WebApi\_Handson**

**Explain RESTful Web Service, Web API & Microservice**

· **RESTful Web Service**: Follows REST architecture using standard HTTP methods for communication (GET, POST, etc.).

· · **Web API**: Framework for building HTTP services. In .NET Core, it allows building RESTful applications using controllers.

· · **Microservice**: Small, independent services performing specific tasks, deployable independently.

**Features of REST Architecture**

· **Stateless**: Each HTTP request from client to server must contain all the information to understand the request.

· **Messages**: Based on HTTP requests/responses using JSON/XML.

· **Not limited to XML**: JSON is default in WebAPI, but XML support can be added.

· **WebService vs WebAPI**:

--> WebService uses SOAP & XML.

--> WebApi uses REST, Supports multiple formats(JSON, XML) and is lightweight.

**HttpRequests & HttpResponse**

· **HttpRequest**: Captures incoming request details (method, headers, body, etc.)

· **HttpResponse**: Sent from server, includes status codes, headers, and content.

**Types of Action Verbs**

1. [HttpGet]: To Retrieve data
2. [HttpPost]: To create data
3. [HttpPut]: To update data
4. [HttpDelete]: To delete data

**Common HttpStatusCodes**

1. 200 Ok
2. 400 BadRequest
3. 401 Unauthorized
4. 500 InternalServerError.

**ValueController.cs**

[ApiController]

[Route("[controller]")]

public class ValuesController : ControllerBase

{

private static List<string> values = new List<string> { "value1", "value2" };

[HttpGet]

public IActionResult Get()

{

return Ok(values);

}

[HttpPost]

public IActionResult Post([FromBody] string value)

{

values.Add(value);

return Ok(values);

}

[HttpPut("{id}")]

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

{

if (id < 0 || id >= values.Count) return BadRequest("Invalid index");

values[id] = value;

return Ok(values);

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

if (id < 0 || id >= values.Count) return BadRequest("Invalid index");

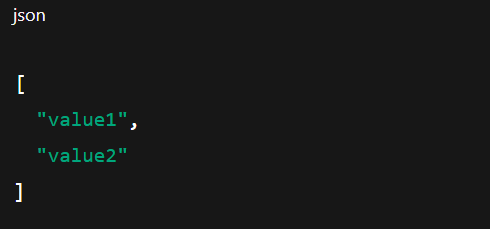
values.RemoveAt(id);

return Ok(values);

}

}

**Output:**

****

1. **WebApi\_Handson**

**Update Startup.cs or Program.cs**

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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")

}

});

});

**Program.cs**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

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

},

License = new OpenApiLicense

{

Name = "License Terms",

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

}

});

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

app.UseAuthorization();

app.MapControllers();

app.Run();

**EmployeeSwaggerAPI.csProj**

<Project Sdk="Microsoft.NET.Sdk.Web">

<PropertyGroup>

<TargetFramework>net6.0</TargetFramework>

<Nullable>enable</Nullable>

<ImplicitUsings>enable</ImplicitUsings>

</PropertyGroup>

<ItemGroup>

<PackageReference Include="Swashbuckle.AspNetCore" Version="6.2.3" />

</ItemGroup>

</Project>

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Http;

using System.Collections.Generic;

namespace EmployeeSwaggerAPI.Controllers

{

[ApiController]

[Route("Emp")]

public class EmployeeController : ControllerBase

{

static List<string> employees = new List<string> { "John", "Jane", "Alice" };

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

public IActionResult GetAll()

{

return Ok(employees);

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public IActionResult Add([FromBody] string name)

{

if (string.IsNullOrWhiteSpace(name)) return BadRequest();

employees.Add(name);

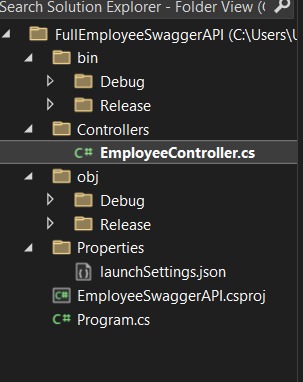
return Ok(employees);

}

}

}

**Output:**

****

1. **WebApi\_Handson.docx**

**Employee.cs**

using System;

using System.Collections.Generic;

namespace EmployeeSwaggerAPI.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; }

}

}

**Department.cs**

namespace EmployeeSwaggerAPI.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Skill.cs**

namespace EmployeeSwaggerAPI.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeSwaggerAPI.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var hasHeader = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token);

if (!hasHeader)

{

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

return;

}

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

{

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

}

}

}

}

**CustomExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc.Filters;

using Microsoft.AspNetCore.Mvc;

using System;

using System.IO;

namespace EmployeeSwaggerAPI.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

string errorMessage = $"[{DateTime.Now}] {context.Exception.Message}\n";

File.AppendAllText("errors.txt", errorMessage);

context.Result = new ObjectResult("An unexpected error occurred.")

{

StatusCode = 500

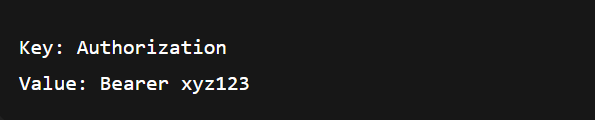
};

}

}

}

**Testing Via Swagger**

****

1. **WebApi\_HandsOn**

**Models/Employees.cs**

using System;

using System.Collections.Generic;

namespace EmployeeAPI.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; }

}

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Controllers/EmployeeController.cs:**

using EmployeeAPI.Models;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

namespace EmployeeAPI.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static List<Employee> employees = new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 60000,

Permanent = true,

Department = new Department { Id = 1, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

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

},

DateOfBirth = new DateTime(1990, 5, 12)

},

new Employee

{

Id = 2,

Name = "Bob",

Salary = 70000,

Permanent = false,

Department = new Department { Id = 2, Name = "IT" },

Skills = new List<Skill>

{

new Skill { Id = 3, Name = "C#" },

new Skill { Id = 4, Name = "SQL" }

},

DateOfBirth = new DateTime(1992, 8, 20)

}

};

[HttpPut("{id}")]

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 properties

existingEmp.Name = updatedEmp.Name;

existingEmp.Salary = updatedEmp.Salary;

existingEmp.Permanent = updatedEmp.Permanent;

existingEmp.Department = updatedEmp.Department;

existingEmp.Skills = updatedEmp.Skills;

existingEmp.DateOfBirth = updatedEmp.DateOfBirth;

return Ok(existingEmp);

}

}

}

**JSON Body Example:**

{

"id": 2,

"name": "Bob Updated",

"salary": 85000,

"permanent": true,

"department": {

"id": 2,

"name": "IT Updated"

},

"skills": [

{ "id": 3, "name": "C# Updated" },

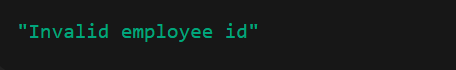
{ "id": 4, "name": "SQL Updated" }

],

"dateOfBirth": "1992-08-20T00:00:00"

}

**Output:**



1. **WebApi\_HandsOn**

**CORS Enablement for Web API Access**

**CORS (Cross-Origin Resource Sharing)** is a security feature implemented by browsers to restrict web pages from making requests to a different domain (origin) than the one that served the web page.

Example: If your front-end app (React/Angular) is running on http://localhost:3000 and your Web API is on http://localhost:5000, requests from the front-end to the API will be blocked by the browser unless CORS is enabled on the server.

**Setup in Program.cs or Startup.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

string securityKey = "mysuperdupersecret";

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

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(x =>

{

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

**AuthController to Generate JWT**

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(1, "Admin");

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(10),

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

1. **WebApi\_HandsOn**

**Producer (Chat Sender)**

using Confluent.Kafka;

using System;

using System.Threading.Tasks;

class ChatProducer

{

public static async Task Main(string[] args)

{

var config = new ProducerConfig { BootstrapServers = "localhost:9092" };

using var producer = new ProducerBuilder<Null, string>(config).Build();

Console.WriteLine("Enter messages to send (type 'exit' to quit):");

while (true)

{

var message = Console.ReadLine();

if (message == "exit") break;

await producer.ProduceAsync("chat-topic", new Message<Null, string> { Value = message });

Console.WriteLine($"Sent: {message}");

}

}

}

### ****Consumer (Chat Receiver)****

using Confluent.Kafka;

using System;

using System.Threading;

class ChatConsumer

{

public static void Main(string[] args)

{

var config = new ConsumerConfig

{

GroupId = "chat-group",

BootstrapServers = "localhost:9092",

AutoOffsetReset = AutoOffsetReset.Earliest

};

using var consumer = new ConsumerBuilder<Ignore, string>(config).Build();

consumer.Subscribe("chat-topic");

Console.WriteLine("Listening for messages (Ctrl+C to exit)...");

CancellationTokenSource cts = new CancellationTokenSource();

Console.CancelKeyPress += (\_, e) => {

e.Cancel = true;

cts.Cancel();

};

try

{

while (true)

{

var cr = consumer.Consume(cts.Token);

Console.WriteLine($"Received: {cr.Message.Value}");

}

}

catch (OperationCanceledException)

{

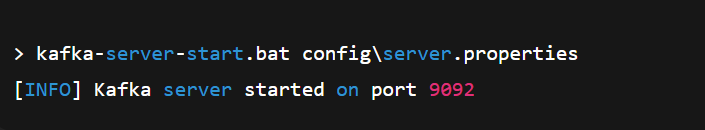
consumer.Close();

}

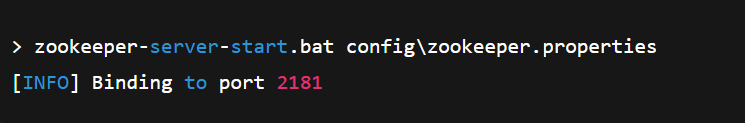
}

}

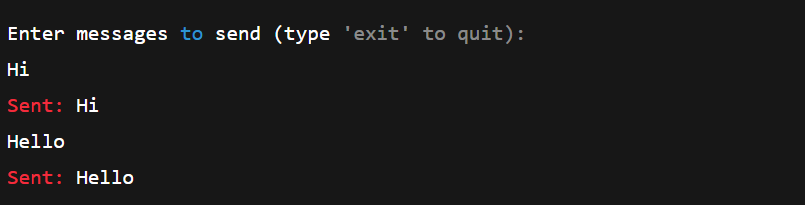
**Kafka Server Running:**



**Zookeeper Running:**



**Chat Producer:**



**Chat Consumer:**

