WEEK 4

ASP.NET Core 8.0 Web API

1. WebApi\_Handson

ValuesController.cs

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace WebApiDemo.Controllers

{

    [ApiController]

    [Route("[controller]")]

    public class ValuesController : ControllerBase

    {

        private static List<string> values = new List<string> { "Value1", "Value2" };

        [HttpGet]

        public ActionResult<List<string>> Get()

        {

            return Ok(values);

        }

        [HttpPost]

        public ActionResult Post([FromBody] string value)

        {

            values.Add(value);

            return Ok("Added Successfully");

        }

        [HttpPut("{index}")]

        public ActionResult Put(int index, [FromBody] string newValue)

        {

            if (index >= values.Count)

                return BadRequest("Index out of range");

            values[index] = newValue;

            return Ok("Updated Successfully");

        }

        [HttpDelete("{index}")]

        public ActionResult Delete(int index)

        {

            if (index >= values.Count)

                return BadRequest("Index out of range");

            values.RemoveAt(index);

            return Ok("Deleted Successfully");

        }

    }

}

1. WebApi\_Handson

Employee.cs

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; } }

EmployeeController.cs

[ApiController]

[Route("[controller]")]

public class EmployeeController : ControllerBase

{

    [HttpGet]

    [ProducesResponseType(StatusCodes.Status200OK)]

    public ActionResult<List<Employee>> Get() => Ok(GetEmployees());

    private List<Employee> GetEmployees()

    {

        return new List<Employee>

        {

            new Employee

            {

                Id = 1,

                Name = "John",

                Salary = 50000,

                Permanent = true,

                DateOfBirth = DateTime.Now.AddYears(-30),

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

                Skills = new List<Skill>

                {

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

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

                }

            }

        };

    }

}

CustomAuthFilter.cs

public class CustomAuthFilter : ActionFilterAttribute

{

    public override void OnActionExecuting(ActionExecutingContext context)

    {

        if (!context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token))

        {

            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");

        }

    }

}

1. WebApi\_Handson

EmployeeController.cs

[HttpPut]

public ActionResult<Employee> UpdateEmployee([FromBody] Employee emp)

{

    if (emp.Id <= 0)

        return BadRequest("Invalid employee id");

    var list = GetEmployees();

    var existing = list.FirstOrDefault(e => e.Id == emp.Id);

    if (existing == null)

        return BadRequest("Invalid employee id");

    existing.Name = emp.Name;

    return Ok(existing);

}

1. WebApi\_Handson

Program.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var key = "mysuperdupersecret";

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

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(x =>

{

    x.TokenValidationParameters = new TokenValidationParameters

    {

        ValidateIssuer = true,

        ValidateAudience = true,

        ValidateLifetime = true,

        ValidateIssuerSigningKey = true,

        ValidIssuer = "mySystem",

        ValidAudience = "myUsers",

        IssuerSigningKey = securityKey

    };

});

AuthController.cs

[ApiController]

[Route("[controller]")]

public class AuthController : ControllerBase

{

    [AllowAnonymous]

    [HttpGet("token")]

    public IActionResult GetToken()

    {

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

        return Ok(token);

    }

    private string GenerateJSONWebToken(int userId, string userRole)

    {

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

        var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

        var claims = new[] {

            new Claim(ClaimTypes.Role, userRole),

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

        };

        var token = new JwtSecurityToken("mySystem", "myUsers", claims, expires: DateTime.Now.AddMinutes(2), signingCredentials: creds);

        return new JwtSecurityTokenHandler().WriteToken(token);

    }

}

1. WebApi\_Handson

Publisher

using Confluent.Kafka;

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

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

while (true)

{

    var msg = Console.ReadLine();

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

}

Consumer

using Confluent.Kafka;

var config = new ConsumerConfig

{

    BootstrapServers = "localhost:9092",

    GroupId = "chat-group",

    AutoOffsetReset = AutoOffsetReset.Earliest

};

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

consumer.Subscribe("chat-topic");

while (true)

{

    var cr = consumer.Consume();

    Console.WriteLine(cr.Message.Value);

}



