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

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace MyFirstWebAPI.Controllers

{

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

[ApiController]

public class ValuesController : ControllerBase

{

// Complete GET method

[HttpGet]

public IEnumerable<string> Get()

{

return new string[] { "value1", "value2" };

}

}

}

A screenshot of a computer

AI-generated content may be incorrect.

Exercise 2: **Web Api using .Net core with Swagger**

1.var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

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://www.example.com")

},

License = new Microsoft.OpenApi.Models.OpenApiLicense

{

Name = "License Terms",

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

}

});

});

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.UseAuthorization();

app.MapControllers();

app.Run();

A screenshot of a computer

AI-generated content may be incorrect.

2.using Microsoft.AspNetCore.Mvc;

namespace MyfirstWebApi.Controllers

{

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

[ApiController]

public class EmployeeController : ControllerBase

{

[HttpGet]

public IActionResult GetEmployees()

{

var employees = new List<object>

{

new { Id = 1, Name = "John", Role = "Developer" },

new { Id = 2, Name = "Jane", Role = "Tester" },

new { Id = 3, Name = "Bob", Role = "Manager" }

};

return Ok(employees);

}

}

}

A screenshot of a computer program

AI-generated content may be incorrect.

3.using Microsoft.AspNetCore.Mvc;

namespace MyfirstWebApi.Controllers

{

[Route("api/emp")]

[ApiController]

public class EmployeeController : ControllerBase

{

[HttpGet]

public IActionResult GetEmployees()

{

var employees = new List<object>

{

new { Id = 1, Name = "John", Role = "Developer" },

new { Id = 2, Name = "Jane", Role = "Tester" },

new { Id = 3, Name = "Bob", Role = "Manager" }

};

return Ok(employees);

}

}

}

A screenshot of a computer

AI-generated content may be incorrect.

Exercise 3: **Web Api using custom model class**

//Program.cs

using EmployeeAPI.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

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

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

//Employee.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 SkillName { get; set; }

}

}

//EmployeeController.cs

using Microsoft.AspNetCore.Mvc;

using EmployeeAPI.Models;

using EmployeeAPI.Filters;

namespace EmployeeAPI.Controllers

{

[ApiController]

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

[CustomAuthFilter] // Authorization Filter

public class EmployeeController : ControllerBase

{

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

public EmployeeController()

{

if (!employeeList.Any())

employeeList = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

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

Skills = new List<Skill> {

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

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

},

DateOfBirth = new DateTime(1990, 5, 15)

}

};

}

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), 200)]

[ProducesResponseType(500)]

public ActionResult<List<Employee>> Get()

{

throw new Exception("Simulated server error");

// return Ok(employeeList);

}

[HttpGet("standard")]

public ActionResult<Employee> GetStandard()

{

return Ok(GetStandardEmployeeList().First());

}

[HttpPost]

public IActionResult Post(Employee emp)

{

employeeList.Add(emp);

return CreatedAtAction(nameof(Get), new { id = emp.Id }, emp);

}

[HttpPut("{id}")]

public IActionResult Put(int id, Employee emp)

{

var existing = employeeList.FirstOrDefault(e => e.Id == id);

if (existing == null)

return NotFound();

existing.Name = emp.Name;

existing.Salary = emp.Salary;

existing.Permanent = emp.Permanent;

existing.Department = emp.Department;

existing.Skills = emp.Skills;

existing.DateOfBirth = emp.DateOfBirth;

return NoContent();

}

}

}

//CustomAuthFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeAPI.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");

}

}

}

}

//CustomExceptionFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

namespace EmployeeAPI.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

File.AppendAllText("logs.txt", $"[{DateTime.Now}] {exception.Message}\n");

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

{

StatusCode = 500

};

}

}

}

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

Exercise 4: **Web Api CRUD operation**

//EmployeeController.cs

using Microsoft.AspNetCore.Mvc;

using EmployeeAPI.Models;

using EmployeeAPI.Filters;

namespace EmployeeAPI.Controllers

{

[ApiController]

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

[CustomAuthFilter] // Authorization Filter

public class EmployeeController : ControllerBase

{

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

public EmployeeController()

{

if (!employeeList.Any())

employeeList = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

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

Skills = new List<Skill> {

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

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

},

DateOfBirth = new DateTime(1990, 5, 15)

}

};

}

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), 200)]

[ProducesResponseType(500)]

public ActionResult<List<Employee>> Get()

{

//throw new Exception("Simulated server error");

return Ok(employeeList);

}

[HttpGet("standard")]

public ActionResult<Employee> GetStandard()

{

return Ok(GetStandardEmployeeList().First());

}

[HttpPost]

public IActionResult Post(Employee emp)

{

employeeList.Add(emp);

return CreatedAtAction(nameof(Get), new { id = emp.Id }, emp);

}

[HttpPut("{id}")]

public ActionResult<Employee> Put(int id, [FromBody] Employee emp)

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

var existing = employeeList.FirstOrDefault(e => e.Id == id);

if (existing == null)

{

return BadRequest("Invalid employee id");

}

// Update fields

existing.Name = emp.Name;

existing.Salary = emp.Salary;

existing.Permanent = emp.Permanent;

existing.Department = emp.Department;

existing.Skills = emp.Skills;

existing.DateOfBirth = emp.DateOfBirth;

return Ok(existing);

}

}

}

A screenshot of a computer program

AI-generated content may be incorrect.

Exercise 5: **JsonWebToken**

//program.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

string securityKey = "mysuperdupersecretkey@345678901234567890"; // Must match

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

// Add services

builder.Services.AddControllers();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new() { Title = "JWTAuthDemo", Version = "v1" });

// Enable JWT token input

c.AddSecurityDefinition("Bearer", new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Name = "Authorization",

Type = Microsoft.OpenApi.Models.SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = Microsoft.OpenApi.Models.ParameterLocation.Header,

Description = "Enter 'Bearer' [space] and then your valid token.\n\nExample: Bearer eyJhbGciOiJIUzI1NiIsInR..."

});

c.AddSecurityRequirement(new Microsoft.OpenApi.Models.OpenApiSecurityRequirement

{

{

new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Reference = new Microsoft.OpenApi.Models.OpenApiReference

{

Type = Microsoft.OpenApi.Models.ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<string>()

}

});

});

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

};

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(); // Optional: add custom endpoint inside here

}

// Add middleware

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

//AuthController.cs

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 JwtAuthDemo.Controllers

{

[ApiController]

[Route("[controller]")]

[AllowAnonymous]

public class AuthController : ControllerBase

{

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkey@345678901234567890"));

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), // Change to 2 to test expiry

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

[HttpGet("login")]

public IActionResult Login()

{

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

return Ok(new { token });

}

}

}

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.