Hands-On Exercises: Authentication and Authorization in ASP.NET Core Web API Microservices

This document contains 4 hands-on exercises focusing on Authentication and Authorization in ASP.NET Core Web API microservices, with an emphasis on implementing JWT (JSON Web Tokens) authentication. Each exercise includes a scenario, step-by-step instructions, and complete solution code.

# Question 1: Implement JWT Authentication in ASP.NET Core Web API

Scenario:

You are building a microservice that requires secure login. You need to implement JWT- based authentication.

Steps:

1. Create a new ASP.NET Core Web API project.
2. Add a `User` model and a login endpoint.
3. Generate a JWT token upon successful login.
4. Secure an endpoint using `[Authorize]`.

Solution Code:

Install NuGet Packages:

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer appsettings.json:

{

"Jwt": {

"Key": "ThisIsASecretKeyForJwtToken", "Issuer": "MyAuthServer",

"Audience": "MyApiUsers", "DurationInMinutes": 60

}

}

Program.cs: builder.Services.AddAuthentication("Bearer")

.AddJwtBearer("Bearer", options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true, ValidateAudience = true, ValidateLifetime = true, ValidateIssuerSigningKey = true,

ValidIssuer = builder.Configuration["Jwt:Issuer"], ValidAudience = builder.Configuration["Jwt:Audience"], IssuerSigningKey = new

SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]))

};

});

builder.Services.AddAuthorization(); AuthController.cs:

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

public class AuthController : ControllerBase

{

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

if (IsValidUser(model))

{

var token = GenerateJwtToken(model.Username); return Ok(new { Token = token });

}

return Unauthorized();

}

private string GenerateJwtToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

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

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

var token = new JwtSecurityToken( issuer: "MyAuthServer", audience: "MyApiUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(60), signingCredentials: creds);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// JWT Authentication

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = builder.Configuration["Jwt:Issuer"] ?? "MyAuthServer",

ValidAudience = builder.Configuration["Jwt:Audience"] ?? "MyApiUsers",

IssuerSigningKey = new SymmetricSecurityKey(

Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"] ?? "ThisIsASecretKeyForJwtToken"))

};

});

builder.Services.AddAuthorization();

var app = builder.Build();

// Middleware pipeline

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.MapGet("/ping", () => "pong");

app.Run();

***AuthController.cs***

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using JwtAuthDemo.Models;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

if (IsValidUser(model))

{

var token = CreateToken(model.Username);

return Ok(new { Token = token });

}

return Unauthorized();

}

private bool IsValidUser(LoginModel model)

{

            // Replace with actual user validation

            return !string.IsNullOrWhiteSpace(model.Username) && !string.IsNullOrWhiteSpace(model.Password);

}

private string CreateToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

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

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

var token = new JwtSecurityToken(

issuer: "MyAuthServer",

audience: "MyApiUsers",

claims: claims,

expires: DateTime.UtcNow.AddHours(1),

signingCredentials: creds);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

***SecureController.cs***

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class SecureController : ControllerBase

{

[Authorize]

[HttpGet("secret")]

public IActionResult GetSecret()

{

return Ok("This is a protected endpoint!");

}

}

}

***Output:***



