// #r "nuget: System.IdentityModel.Tokens.Jwt, 6.34.0"

// #r "nuget: Microsoft.IdentityModel.Tokens, 6.34.0"

using System;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using Microsoft.IdentityModel.Tokens;

public class Program

{

// ─── JWT settings (mirror appsettings.json) ───

private const string JwtKey = "ThisIsASuperSecretKeyForJwtToken123"; // 36 chars (>= 256 bits)

private const string Issuer = "MyAuthServer";

private const string Audience = "MyApiUsers";

private const int ExpMinutes = 60;

public static void Main()

{

Console.WriteLine("== JWT Authentication / Authorization demo ==\n");

// 1. “Login” – simulate user auth

var username = "alice";

var role = "Admin"; // try changing to "User" and re-run

var jwt = GenerateJwtToken(username, role);

Console.WriteLine(" Generated JWT:\n");

Console.WriteLine(jwt + "\n");

// 2. Simulate hitting a [Authorize] endpoint

Console.WriteLine(" Validating token (like ASP.NET pipeline) …");

var principal = ValidateJwt(jwt);

if (principal == null)

{

Console.WriteLine(" Token invalid or expired.");

return;

}

Console.WriteLine($" Token valid! User = {principal.Identity!.Name}");

// 3. Simulate [Authorize(Roles = "Admin")]

if (principal.IsInRole("Admin"))

Console.WriteLine(" Role check passed – welcome to the admin dashboard!");

else

Console.WriteLine(" Access denied – you are not an Admin.");

}

// Token generation

private static string GenerateJwtToken(string username, string role)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username),

new Claim(ClaimTypes.Role, role)

};

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

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

var token = new JwtSecurityToken(

issuer: Issuer,

audience: Audience,

claims: claims,

expires: DateTime.UtcNow.AddMinutes(ExpMinutes),

signingCredentials: creds);

return new JwtSecurityTokenHandler().WriteToken(token);

}

// Token validation (simulates [Authorize] filter)

private static ClaimsPrincipal? ValidateJwt(string jwt)

{

var parameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = Issuer,

ValidAudience = Audience,

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(JwtKey)),

ClockSkew = TimeSpan.Zero // strict expiry check

};

try

{

var handler = new JwtSecurityTokenHandler();

var principal = handler.ValidateToken(jwt, parameters, out \_);

return principal;

}

catch (SecurityTokenExpiredException)

{

Console.WriteLine(" Token has expired.");

return null;

}

catch (Exception ex)

{

Console.WriteLine($" Validation failed: {ex.Message}");

return null;

}

}

}

