JWT Authentication in ASP.NET Core Web API

This document provides a complete, beginner-friendly step-by-step guide to implementing JWT Authentication in ASP.NET Core Web API using Visual Studio. It includes code examples, configuration steps, and Postman testing instructions.

# Step 1: Create a New ASP.NET Core Web API Project

1. Open Visual Studio.

2. Click 'Create a new project'.

3. Select 'ASP.NET Core Web API'.

4. Name the project: JwtAuthDemo and click 'Create'.

5. Choose .NET 6 or .NET 7 as framework.

# Step 2: Install NuGet Package

Go to Tools > NuGet Package Manager > Manage NuGet Packages for Solution.

Install: Microsoft.AspNetCore.Authentication.JwtBearer

# Step 3: Update appsettings.json

Add the following section inside appsettings.json:

"Jwt": {  
 "Key": "ThisIsASecretKeyForJwtToken",  
 "Issuer": "MyAuthServer",  
 "Audience": "MyApiUsers",  
 "DurationInMinutes": 60  
}

# Step 4: Configure JWT in Program.cs

Modify Program.cs as follows:

using Microsoft.AspNetCore.Authentication.JwtBearer;  
using Microsoft.IdentityModel.Tokens;  
using System.Text;  
  
var builder = WebApplication.CreateBuilder(args);  
  
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();  
builder.Services.AddControllers();  
  
var app = builder.Build();  
  
app.UseAuthentication();  
app.UseAuthorization();  
  
app.MapControllers();  
app.Run();

# Step 5: Create LoginModel.cs

Create a folder named Models and add the following class:

namespace JwtAuthDemo.Models  
{  
 public class LoginModel  
 {  
 public string Username { get; set; }  
 public string Password { get; set; }  
 }  
}

# Step 6: Create AuthController.cs

Create a controller named AuthController.cs with the following content:

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 = GenerateJwtToken(model.Username);  
 return Ok(new { Token = token });  
 }  
  
 return Unauthorized("Invalid credentials");  
 }  
  
 private static bool IsValidUser(LoginModel model)  
 {  
 return model.Username == "admin" && model.Password == "password";  
 }  
  
 private static string GenerateJwtToken(string username)  
 {  
 var claims = new[]  
 {  
 new Claim(ClaimTypes.Name, username),  
 new Claim(ClaimTypes.Role, "Admin")  
 };  
  
 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);  
 }  
 }  
}

# Step 7: Create SecureController.cs

Create another controller named SecureController.cs with this code:

using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Mvc;  
  
namespace JwtAuthDemo.Controllers  
{  
 [ApiController]  
 [Route("api/[controller]")]  
 public class SecureController : ControllerBase  
 {  
 [HttpGet("data")]  
 [Authorize]  
 public IActionResult GetSecureData()  
 {  
 return Ok("This is protected data.");  
 }  
 }  
}

# Step 8: Test in Postman

1. Run the project and copy the localhost URL (e.g., https://localhost:7282).

2. In Postman, create a POST request to /api/auth/login with body:

{  
 "username": "admin",  
 "password": "password"  
}

3. Copy the token from the response.

4. Create a GET request to /api/secure/data with header:







