I'll guide you through creating the ASP.NET Core Web API with JWT Authentication and Product Controller using Visual Studio 2022. Here's how to do it step-by-step:

---

1. Create new ASP.NET Core Web API Project using Visual Studio 2022

2. Add NuGet Packages

Install the following packages Package Manager Console:

```
Install-Package Microsoft.AspNetCore.Authentication.JwtBearer   -Version 8.0.2
Install-Package System.IdentityModel.Tokens.Jwt     -Version 8.0.2
```

3. Update appsettings.json

In Solution Explorer, double-click `appsettings.json`, add "Jwt" section as follows:

```
{
  "Jwt": {
    "Key": "YourSecretKeyHere1234567890",
    "Issuer": "YourIssuer",
    "Audience": "YourAudience"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

4. Add Models

1. Right-click project > Add > New Folder > Name it "Models"
2. Right-click Models folder > Add > Class

Product.cs:

```
namespace WebApplication22.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
```

```
        public decimal Price { get; set; }
    }
}
```

User.cs:

```
namespace WebApplication22.Models
{
    public class UserModel
    {
        public string Username { get; set; }
        public string Password { get; set; }
    }
}
```

5. Add Controllers

1. Right-click Controllers folder > Add > Controller
2. Select "API Controller - Empty"

AuthController.cs:

```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

// Additional Namespaces need to import
using Microsoft.AspNetCore.Identity;
using Microsoft.Extensions.Configuration;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using Microsoft.IdentityModel.Tokens;
using System.Text;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Authorization;   // for AllowAnonymous
using WebApplication22.Models;

namespace WebApplication22.Controllers
{
    [AllowAnonymous]
    [Route("api/[controller]")]
    [ApiController]
    public class AuthenticateController : ControllerBase
    {

        public List<UserModel> usersList = null;
        public AuthenticateController()
```

```
        {
            usersList = new List<UserModel>()
            {
                new UserModel() { UserName = "Admin", Password = "Admin123" },
                new UserModel() { UserName = "Scott", Password = "Scott123"}
            };
        }

        [HttpPost]
        public IActionResult Login(UserModel requestUser)
        {
            UserModel userObj = usersList.Where(x => x.UserName ==
requestUser.UserName && x.Password == requestUser.Password).FirstOrDefault();

            if (userObj != null)
            {
                string tokenStr = GenerateJSONWebToken(userObj);
                return Ok(new { token = tokenStr });
            }
            else
            {
                return BadRequest("Invalid user id or password");
            }
        }

        private string GenerateJSONWebToken(UserModel userObj)
        {
            SymmetricSecurityKey securityKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecret"));
            SigningCredentials credentials = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);


            List<Claim> authClaims = new List<Claim>
            {
                    new
Claim(ClaimTypes.NameIdentifier,Convert.ToString(userObj.UserId)),
                    new Claim(ClaimTypes.Name, userObj.UserName),
                    new Claim(JwtRegisteredClaimNames.Jti,
Guid.NewGuid().ToString()),  // (JWT ID) Claim
                    new Claim(ClaimTypes.Role, userObj.Role)
             };

            JwtSecurityToken token = new JwtSecurityToken(
                        issuer: "mySystem",
                        audience: "myUsers",
                        claims: authClaims,
                        expires: DateTime.Now.AddMinutes(10),
                        signingCredentials: credentials);

            string tokenString = new JwtSecurityTokenHandler().WriteToken(token);

            return tokenString;
        }
```

```
    }
}
```

ProductController.cs:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using WebApplication22.Models;

namespace WebApplication22.Controllers
{

    [Route("api/[controller]")]
    [ApiController]
    [Authorize]
    // [Authorize(Roles = "Admin")]
    public class ProductsController : ControllerBase
    {
        private readonly ProductDbContext _context;

        public ProductsController(ProductDbContext context)
        {
            _context = context;
        }

        // GET: api/products
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Product>>> GetProducts()
        {
            return await _context.Products.ToListAsync();
        }

        // GET: api/products/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Product>> GetProduct(int id)
        {
            var product = await _context.Products.FindAsync(id);
            if (product == null) return NotFound();
            return product;
        }

        // POST: api/products
        [HttpPost]
        public async Task<ActionResult<Product>> PostProduct(Product product)
        {
            _context.Products.Add(product);
            await _context.SaveChangesAsync();
            return CreatedAtAction(nameof(GetProduct), new { id = product.Id },
product);
        }
```

```
        // PUT: api/products/5
        [HttpPut("{id}")]
        public async Task<IActionResult> PutProduct(int id, Product product)
        {
            if (id != product.Id) return BadRequest();

            _context.Entry(product).State = EntityState.Modified;
            await _context.SaveChangesAsync();
            return NoContent();
        }

        // DELETE: api/products/5
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteProduct(int id)
        {
            var product = await _context.Products.FindAsync(id);
            if (product == null) return NotFound();

            _context.Products.Remove(product);
            await _context.SaveChangesAsync();
            return NoContent();
        }
    }
}
```

6. Update Program.cs

```
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;
using JwtProductApi.Services;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container
builder.Services.AddControllers();
builder.Services.AddScoped<IJwtService, JwtService>();

// Configure 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"],
            ValidAudience = builder.Configuration["Jwt:Audience"],
```

```
            IssuerSigningKey = new SymmetricSecurityKey(
                Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]))
        };
    });

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();

// Add the below statements to enable security
app.UseAuthentication();
app.UseAuthorization();

app.MapControllers();

app.Run();
```

7. Run and Test

1. Press F5 to run the project
2. The Swagger UI should open in your browser
3. Test the endpoints:
    - First, use POST `/api/auth/login` with `{"username": "test", "password": "password"}`
    - Copy the token from the response
    - Click "Authorize" in Swagger UI and enter "Bearer {token}"
    - Test the Product endpoints

Note: Use Swagger / Postman to test the end ponts

Visual Studio 2022 Tips

- Use Ctrl+. for quick fixes and adding using statements
- Right-click > "Go To Definition" to navigate code
- Use the built-in debugger (F5) with breakpoints (F9)
- Solution Explorer shows all files and allows easy navigation