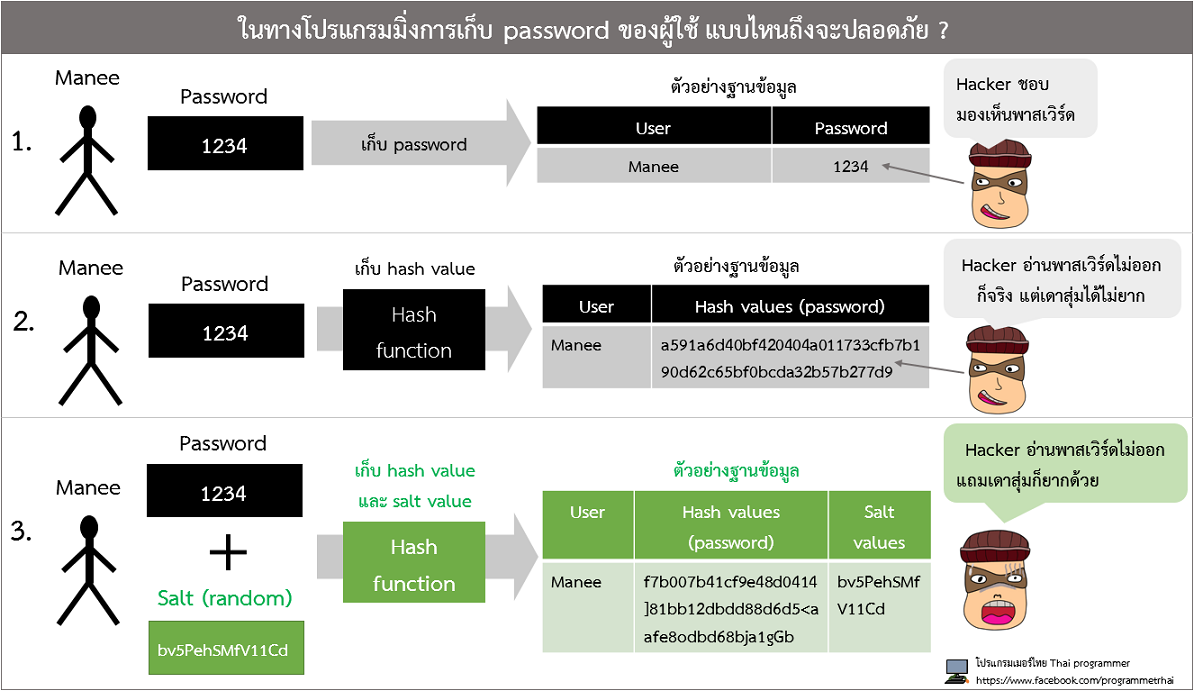
**การยืนยันตัวตน (Authentication)**

**แบบไม่ใช้ Identiy Framework**

**Hash Password** เข้ารหัสผ่านด้วยเทคนิค Hash+Salt เพื่อให้ค่าแฮชไม่ซ้ำกัน

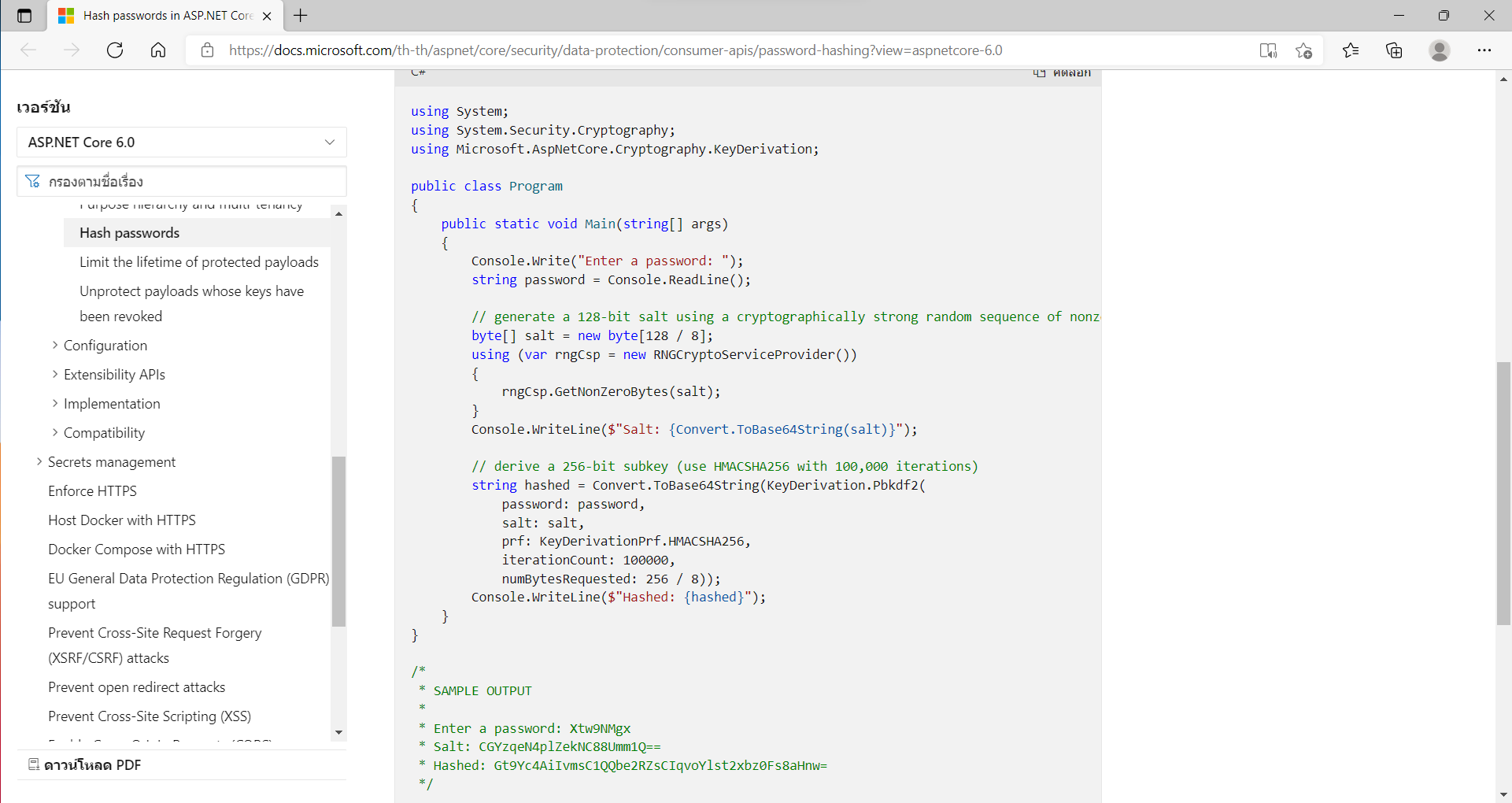
<https://www.patanasongsivilai.com/blog/password-hash-function-salt-values/>

โค้ดการ hash+salt

ค้น hash password.net core ไปดูหลักการ hash ใน help

แบบ 1 <https://docs.microsoft.com/th-th/aspnet/core/security/data-protection/consumer-apis/password-hashing?view=aspnetcore-6.0>

แบบ 2 อันนี้ง่าย Bcrypt <https://jasonwatmore.com/post/2020/07/16/aspnet-core-3-hash-and-verify-passwords-with-bcrypt>

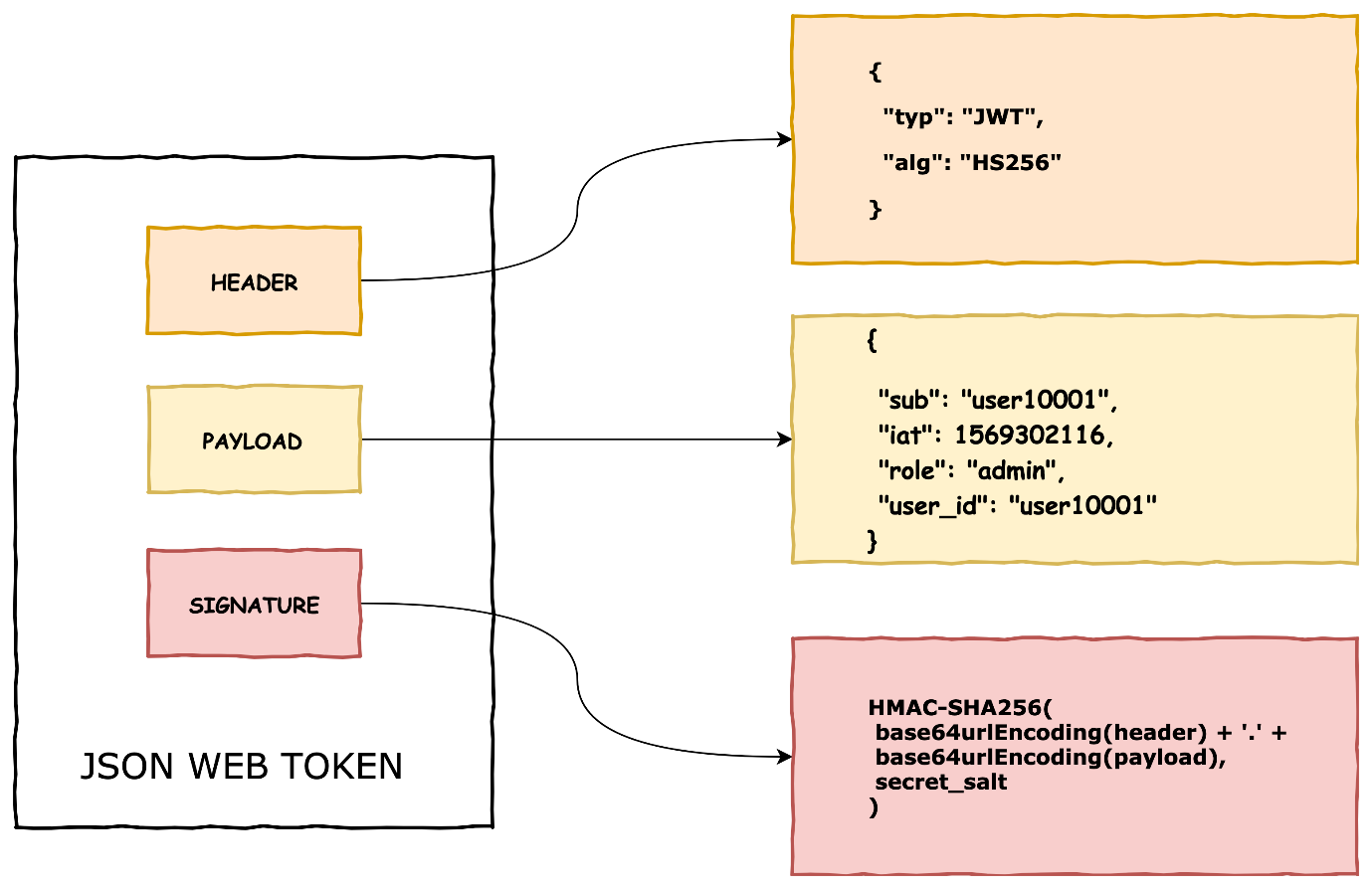


**Token**

Token เป็นรหัสชุดหนึ่งเพื่อระบุตัวตนว่าเป็นใครซึ่งเทียบเท่ากับลายเซ็นต์ เมื่อเข้ารหัสแล้วไม่สามารถเปลี่ยนแปลงแก้ไขได้

JWT (Json Web Token)

JWT เป็นรูปแบบหนึ่งที่ใช้ในการสร้างรหัส token จากข้อมูล JSON Data แล้วทำการเข้ารหัสด้วย Base64Url และถือว่าเป็นมาตรฐานเปิด(RFC 7519) ที่เข้ามาแก้ปัญหาการส่งข้อมูลอย่างปลอดภัยระหว่างกัน โดยถูกออกแบบไว้ว่า จะต้องมีขนาดที่กะทัดรัด (Compact) และเก็บข้อมูลภายในตัว(Self-contained)

ส่วนประกอบ JWT

ประกอบด้วย 3 ส่วน ดังนี้

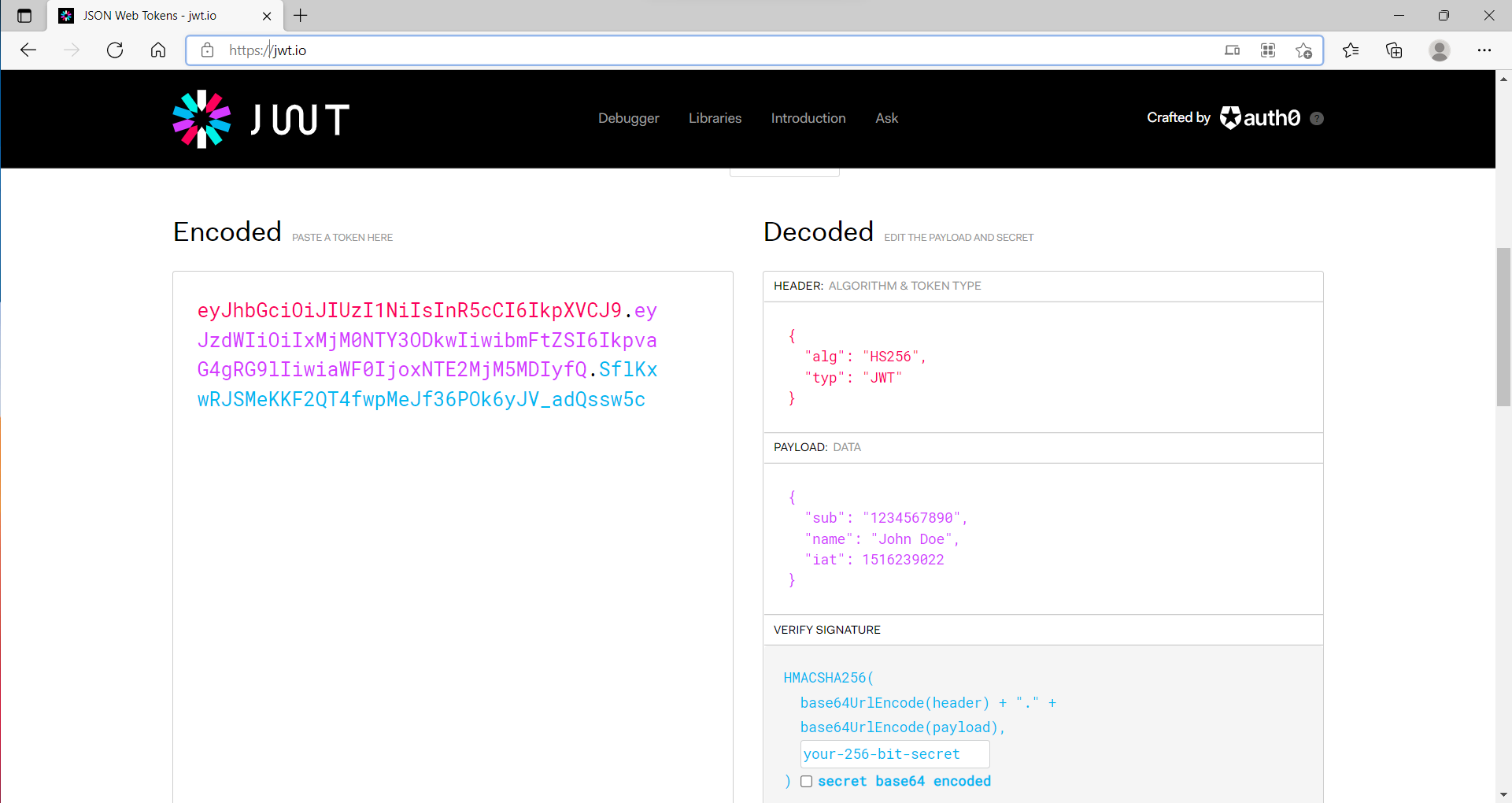
1.Header คือข้อมูล metadata ของ token ซึ่งบอกว่า เป็น type และใช้ algorithm อะไร

2.Body หรือ Payload หรือ Claims คือ ข้อมูลทั้งหมดที่เอาไว้ sign token

3.Signature : ส่วนสำคัญของข้อมูล เป็นการรวมกันของ Header และ Body ใช้ algorithm และ secret key ในการ sign

อ่านเพิ่มเติม <https://devahoy.com/blog/2016/07/understanding-token-and-jwt-create-authentication-with-hapijs/>

<https://jwt.io> เว็ปทดสอบ JWT



ขั้นตอนการใช้ Token

1. Login เข้าระบบไปยังฐานข้อมูล
2. สร้าง Token
3. ใช้ Token โดยไม่ต้องไปดึงข้อมูลมาจากฐานข้อมูลอีก

**การสร้างฐานข้อมูลแบบ Code First**

รูปแบบคำสั่งการสร้างตารางในฐานข้อมูล ef migration

<https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli>

Add a migration

dotnet ef migrations add InitialCreate

Create/Upate database and schema (latest migration)

dotnet ef database update

OR The following updates your database to a given migration:

dotnet ef database update AddNewTables

\*สีแดงคือชื่อ Migration ที่ต้องการกำหนด หรือจัดการ

ตัวอย่าง

การสร้างตารางในฐานข้อมูล

dotnet ef migrations add InitialCreate เริ่มสร้างกำหนดชื่อ migration ว่า InitialCreate

dotnet ef migrations remove ลบการ migration ตัวล่าสุดออก

dotnet ef database update การอัพเดทฐานข้อมูลด้วยตัว migration ล่าสุด

การสร้างตารางโดยระบุชื่อ context

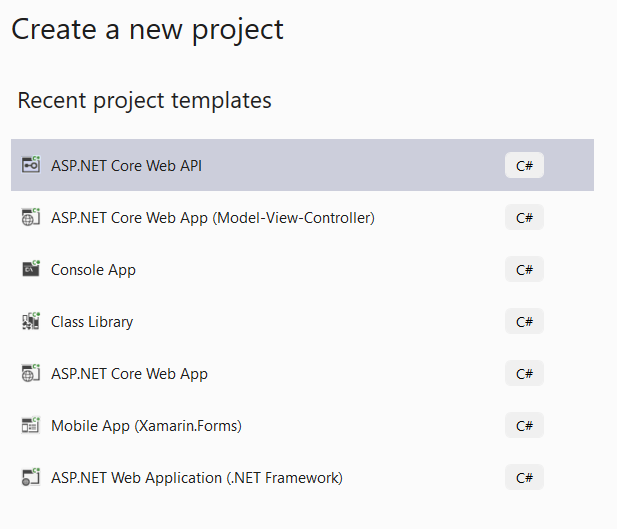
dotnet ef migrations add CreateIdentitySchema --context ชื่อContext

dotnet ef database update --context ชื่อContext

dotnet ef migrations remove -c IdentityContext

\*หมายเหตุ การระบุ --context กรณีมี dbcontext มากกว่า 1 ตัว

สร้างโปรเจค Web API เพื่อให้ง่ายต่อการทดสอบ ชื่อ JwtAuthen



ติดตั้ง Nugets : Entity Framework (แพคเกตสำหรับสร้างฐานข้อมูล)

Microsoft.EntityFrameworkCore

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.Design

ติดตั้ง Nugets สำหรับเข้ารหัส(HashPassword)

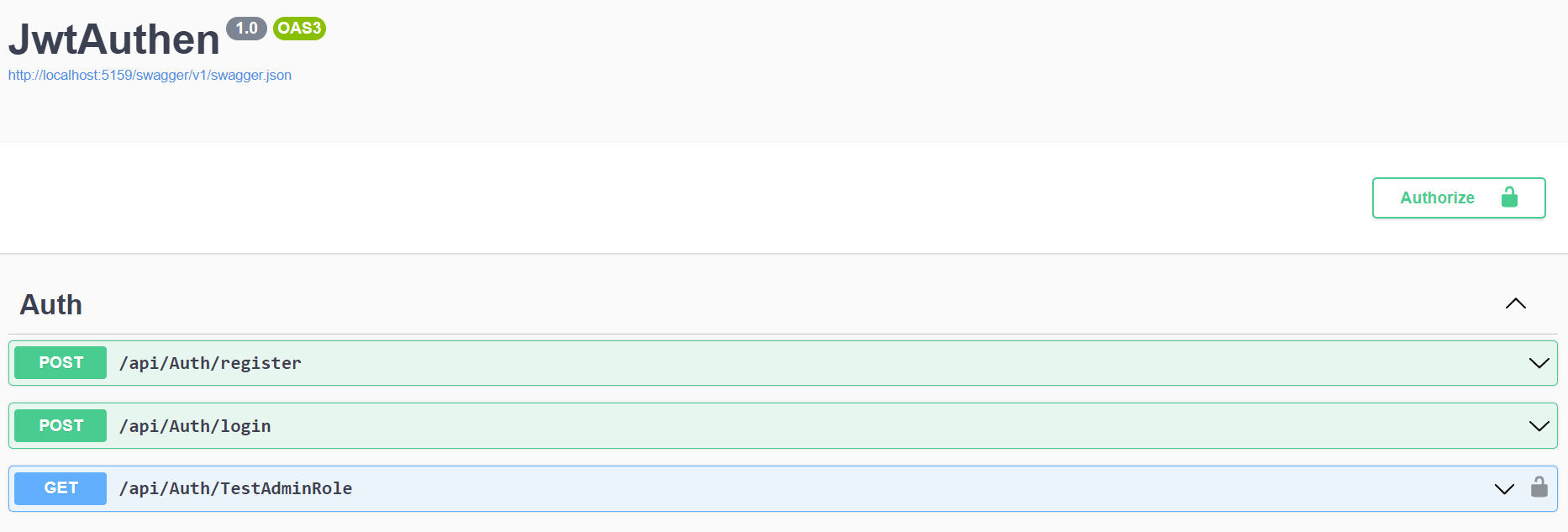
BCrypt.Net-Next

ติดตั้ง Nugets สำหรับจัดการ Token (authenticationScheme)

Microsoft.AspNetCore.Authentication.JwtBearer

ติดตั้ง Nugets สำหรับจัดการ Swaggle (Authorize)

Swashbuckle.AspNetCore.Filters

ผลลัพธ์ที่ต้องการ Register, Login , TestAdminRold

สร้าง Models

Models\User.cs

public class User

{

public string Username { get; set; } = string.Empty;

public string PasswordHash { get; set; } = string.Empty;

}

Models\UserDto.cs

public class UserDto

{

public required string Username { get; set; }

public required string Password { get; set; }

}

appsettings.json

{

"AppSettings": {

"Token": "my top secret key"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

Program.cs

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using Swashbuckle.AspNetCore.Filters;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

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

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

{

options.AddSecurityDefinition("oauth2", new OpenApiSecurityScheme

{

In = ParameterLocation.Header,

Name = "Authorization",

Type = SecuritySchemeType.ApiKey

});

options.OperationFilter<SecurityRequirementsOperationFilter>();

});

builder.Services.AddAuthentication().AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuerSigningKey = true,

ValidateAudience = false,

ValidateIssuer = false,

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(

builder.Configuration.GetSection("AppSettings:Token").Value!))

};

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseAuthorization();

app.MapControllers();

app.Run();

AuthController.cs

using JwtAuthen.Models;

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 JwtAuthen.Controllers

{

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

[ApiController]

public class AuthController : ControllerBase

{

public static User user = new User();

private readonly IConfiguration \_configuration;

public AuthController(IConfiguration configuration)

{

\_configuration = configuration;

}

[HttpPost("register")]

public ActionResult<User> Register(UserDto request)

{

string passwordHash

= BCrypt.Net.BCrypt.HashPassword(request.Password);

user.Username = request.Username;

user.PasswordHash = passwordHash;

return Ok(user);

}

[HttpPost("login")]

public ActionResult<User> Login(UserDto request)

{

if (user.Username != request.Username)

{

return BadRequest("User not found.");

}

if (!BCrypt.Net.BCrypt.Verify(request.Password, user.PasswordHash))

{

return BadRequest("Wrong password.");

}

string token = CreateToken(user);

return Ok(token);

}

private string CreateToken(User user)

{

List<Claim> claims = new List<Claim> {

new Claim(ClaimTypes.Name, user.Username),

new Claim(ClaimTypes.Role, "Admin"),

new Claim(ClaimTypes.Role, "User"),

};

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(

\_configuration.GetSection("AppSettings:Token").Value!));

var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha512Signature);

var token = new JwtSecurityToken(

claims: claims,

expires: DateTime.Now.AddDays(1),

signingCredentials: creds

);

var jwt = new JwtSecurityTokenHandler().WriteToken(token);

return jwt;

}

[HttpGet("TestAdminRole"), Authorize(Roles = "Admin")]

public IActionResult test()

{

return Ok("Authorize Success");

}

}

}