**NET Core JWT Authentication with Built in [Authorize] attaribute using Startup.cs settings**

**1) Application Layer -> Create “Model” Folder and add below classes in that folder**

**a) LoginModel.cs**

public class LoginModel

{

public string UserName { get; set; }

public string Password { get; set; }

}

**b) TokenModel.cs**

public class TokenModel

{

public string AccessToken { get; set; }

public DateTime AccessTokenExpiryTime { get; set; }

public string UserName { get; set; }

public string Name { get; set; }

public string EmailId { get; set; }

}

**c) UserModel.cs**

public class UserModel

{

public int UserId { get; set; }

public string UserName { get; set; }

public string Password { get; set; }

public string Name { get; set; }

public string Email { get; set; }

}

**d) ProductModel.cs**

public class ProductModel

{

public int Id { get; set; }

public string Name { get; set; }

public int Quantity { get; set; }

public int Price { get; set; }

}

**e) AppSettings.cs**

public class AppSettings

{

public string SecretKey { get; set; }

}

===================================================================

**2) Application Layer -> Create “Interface” Folder and add below interface in that folder**

**a) ILoginService.cs**

public interface ILoginService

{

TokenModel GetTokenModel(LoginModel loginModel);

}

**b) IUserService.cs**

public interface IUserService

{

UserModel GetUserDetails(LoginModel loginModel);

Task<UserModel> GetUserDetailsById(Expression<Func<UserModel, bool>>

predicate);

}

**c) IProductService.cs**

public interface IProductService

{

List<ProductModel> GetProductList();

}

=====================================================================

**3) Application Layer -> Create “Services” Folder and add below services in that folder**

**a) LoinService.cs**

public class LoginService : ILoginService

{

private readonly AppSettings \_appSettings;

private readonly IUserService \_userSerice;

public LoginService(IOptions<AppSettings> appSettings, IUserService userSerice)

{

\_appSettings = appSettings.Value;

\_userSerice = userSerice;

}

public TokenModel GetTokenModel(LoginModel loginModel)

{

var user = \_userSerice.GetUserDetails(loginModel);

DateTime accessTokenExpirtTime = DateTime.UtcNow.AddDays(1);

var token = new TokenModel

{

AccessToken = GenerateJwtToken(user),

AccessTokenExpiryTime = accessTokenExpirtTime,

UserName = user.UserName,

Name = user.Name,

EmailId = user.Email

};

return token;

}

private string GenerateJwtToken(UserModel user)

{

var tokenHandler = new JwtSecurityTokenHandler();

var securityKey = Encoding.ASCII.GetBytes(\_appSettings.SecretKey);

var tokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(new Claim[]

{

new Claim("id", user.UserId.ToString()),

new Claim("userName", user.UserName)

}),

Expires = DateTime.UtcNow.AddHours(1),

SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(

securityKey), SecurityAlgorithms.HmacSha256Signature)

};

var token = tokenHandler.CreateToken(tokenDescriptor);

var finaltoken = tokenHandler.WriteToken(token);

return finaltoken;

}

}

**b) UserService.cs**

public class UserService : IUserService

{

public UserModel GetUserDetails(LoginModel loginModel)

{

if (loginModel != null && loginModel.UserName == "admin" &&

loginModel.Password == "abc")

{

UserModel user = new UserModel()

{

UserId = 1,

Name = "Admin",

Email = "admin@gmail.com",

UserName = "admin",

Password = "abc"

};

return user;

}

return null;

}

public async Task<UserModel> GetUserDetailsById(Expression<Func<UserModel,

bool>> predicate)

{

await Task.Delay(100);

UserModel user = new UserModel()

{

UserId = 1,

Name = "Admin",

Email = "admin@gmail.com",

UserName = "admin",

Password = "abc"

};

return user;

}

}

**c) ProductService.cs**

public class ProductService : IProductService

{

public List<ProductModel> GetProductList()

{

List<ProductModel> lst = new List<ProductModel>()

{

new ProductModel { Id=1,Name="Product1",Price=10,Quantity=10},

new ProductModel { Id=2,Name="Product2",Price=20,Quantity=20},

new ProductModel { Id=3,Name="Product3",Price=30,Quantity=30},

new ProductModel { Id=4,Name="Product4",Price=40,Quantity=40}

};

return lst;

}

}

====================================================================

**4) Add below code into appsettings.cs**

"JwtCongifuration": {

"SecretKey": "This is my sample key"

}

===============================================================

**5) Create “Helper” folder in API project and add “JWtMiddleware.cs” in the file**

public class JwtMiddleware

{

private readonly RequestDelegate \_next;

private readonly AppSettings \_appSettings;

public JwtMiddleware(RequestDelegate next, IOptions<AppSettings> appSettings)

{

\_next = next;

\_appSettings = appSettings.Value;

}

public async Task Invoke(HttpContext context, IUserService userService)

{

var token = context.Request.Headers["Authorization"].FirstOrDefault()?.Split(" ").Last();

if (token != null)

await attachAccountToContext(context, userService, token);

await \_next(context);

}

private async Task attachAccountToContext(HttpContext context, IUserService userService, string token)

{

try

{

var tokenHandler = new JwtSecurityTokenHandler();

var securityKey = Encoding.ASCII.GetBytes(\_appSettings.SecretKey);

tokenHandler.ValidateToken(token, new TokenValidationParameters

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(securityKey),

ValidateIssuer = false,

ValidateAudience = false,

// set clockskew to zero so tokens expire exactly at token expiration time (instead of 5 minutes later)

ClockSkew = TimeSpan.Zero

}, out SecurityToken validatedToken);

var jwtToken = (JwtSecurityToken)validatedToken;

var userId = jwtToken.Claims.First(x => x.Type == "id").Value;

if (userId != null)

{

// attach account to context on successful jwt validation

var user = await userService.GetUserDetailsById(x=>x.UserId == int.Parse(userId));

context.Items["User"] = user;

}

}

catch (Exception ex)

{

throw new Exception(ex.ToString());

}

}

}

=======================================================================

**6) Add below code in Startup.cs**

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

var appSettingSections = Configuration.GetSection("JwtCongifuration");

services.Configure<AppSettings>(appSettingSections);

AppSettings appSettings = appSettingSections.Get<AppSettings>();

var securityKey = Encoding.ASCII.GetBytes(appSettings.SecretKey);

services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;

}).AddJwtBearer(x => {

x.RequireHttpsMetadata = false;

x.SaveToken = false;

x.TokenValidationParameters = new Microsoft.IdentityModel.Tokens. TokenValidationParameters

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(securityKey),

ValidateIssuer = false,

ValidateAudience = false,

};

});

services.AddTransient<IUserService, UserService>();

services.AddTransient<ILoginService, LoginService>();

services.AddTransient<IProductService, ProductService>();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

**app.UseAuthentication();**

app.UseRouting();

**app.UseMiddleware<JwtMiddleware>();**

**app.UseAuthorization();**

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

====================================================================

**7) Add below controllers**

**a) AuthorizeController.cs**

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

[ApiController]

public class AuthorizeController : ControllerBase

{

private readonly ILoginService \_loginService;

public AuthorizeController(ILoginService loginService)

{

\_loginService = loginService;

}

[HttpPost("LoginUser")]

public IActionResult LoginUser(LoginModel loginModel)

{

var tokenModel = \_loginService.GetTokenModel(loginModel);

return Ok(tokenModel);

}

}

**b) ProductController.cs**

**[Authorize]**

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

[ApiController]

public class ProductController : ControllerBase

{

private readonly IProductService \_productService;

public ProductController(IProductService productService)

{

\_productService = productService;

}

[HttpGet("GetProductList")]

public IActionResult GetProductList()

{

var result = \_productService.GetProductList();

return Ok(result);

}

}