Use JSON Web Tokens to Secure Web API Methods

Paul D. Sheriff

BUSINESS SOLUTIONS ARCHITECT, FAIRWAY TECHNOLOGIES, INC.

www.fairwaytech.com psheriff@fairwaytech.com

Goals

Secure Web API with [Authorize]

Setup JSON Web Tokens

Read JWT settings from configuration

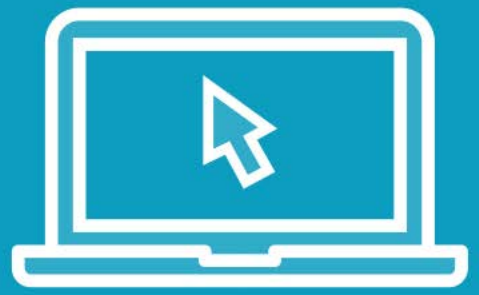
Build bearer token and send to Angular

Secure Get() Method

Secure methods/class using [Authorize] attribute

```
[Route("api/[controller]")]
public class ProductController : BaseApiController
{
  [HttpGet]
  [Authorize]
  public IActionResult Get()
  {
    IActionResult ret = null;
    List<Product> list = new List<Product>();
```

Demo



Secure Web API

Show error

JSON Web Token Overall Steps

Adding JWT

Add JWT package

Add JWT bearer token checking package

Store default JWT settings in configuration file

Register JWT as the authentication service

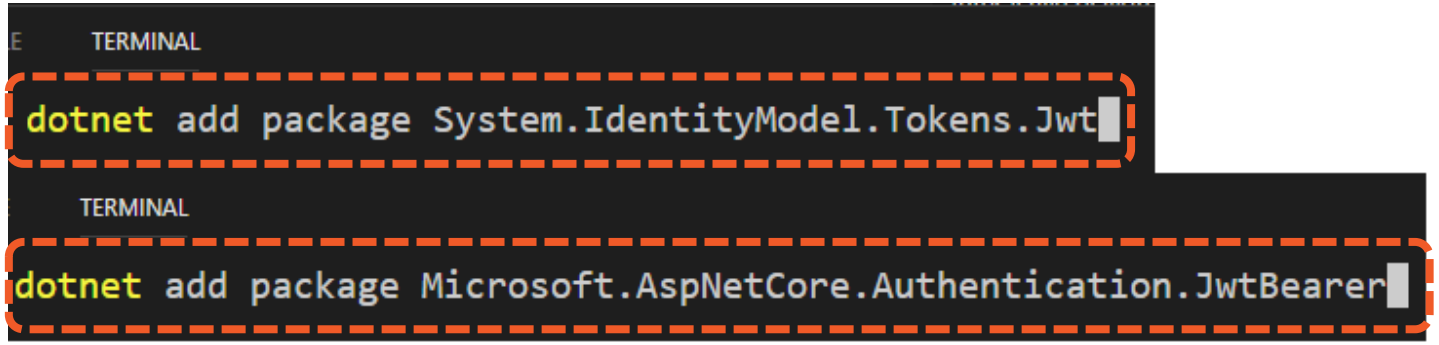
Add bearer token options to validate incoming token

Build JWT token and add to authorization object

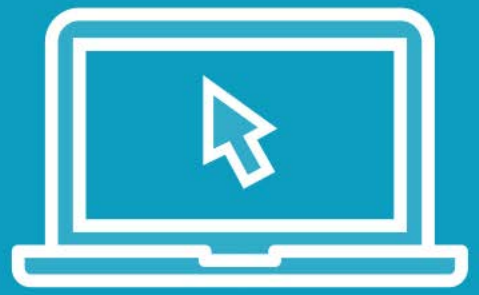
Use .NET Core to add packages

JWT tokens

JWT bearer token



Demo



Add JWT token package

Add JWT bearer token checking package

Configuring JWT

Configuring JSON Web Tokens

A secret key used for hashing data sent to the client

The name of the issuer of the token

The intended audience of the token

How many minutes to allow the token to be valid

Store these items into appsettings.json in Web API project

```
{
  "Logging": {…
  },
  "JwtSettings": {
    "key": "This*Is&A!Long)Key(For%Creating@A$SymmetricKey",
    "issuer": "http://localhost:5000",
    "audience": "PTCUsers",
    "minutesToExpiration": "10"
  }
}
```

Create JwtSettings class

Holds all required data for generating JSON Web Tokens

```
public class JwtSettings {
    public string Key { get; set; }
    public string Issuer { get; set; }
    public string Audience { get; set; }
    public int MinutesToExpiration { get; set; }
}
```

Create GetJwtSettings() method in Startup.cs

Create new instance of JwtSettings

Read data from appsettings.json

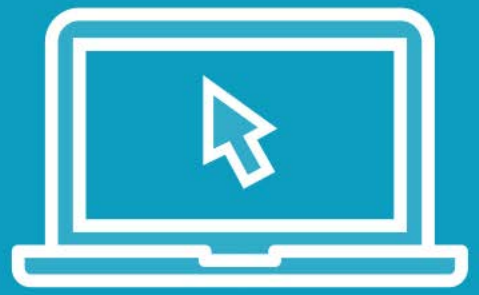
Return settings

```
public JwtSettings GetJwtSettings()
{
    JwtSettings settings = new JwtSettings();

    settings.Key = Configuration["JwtSettings:key"];
    settings.Audience = Configuration["JwtSettings:audience"];
    settings.Issuer = Configuration["JwtSettings:issuer"];
    settings.MinutesToExpiration =
     Convert.ToInt32(
        Configuration["JwtSettings:minutesToExpiration"]);

    return settings;
}
```

Demo



Store JwtSettings into JSON file

Create JwtSettings class

Create method to read settings

Set JWT as Authentication Service

Register JWT as authentication provider in Startup class

Create instance of JwtSettings class

Call method to read from JSON file

Add Authentication options

```
public void ConfigureServices(IServiceCollection services)
{
  // Get JWT Token Settings from JwtSettings.json file
  JwtSettings settings;
  settings = GetJwtSettings();

  // Register Jwt as the Authentication service
  services.AddAuthentication(options =>
  {
    options.DefaultAuthenticateScheme = "JwtBearer";
    options.DefaultChallengeScheme = "JwtBearer";
  })
```

Add bearer token service

Set parameters using *settings* object

```
.AddJwtBearer("JwtBearer", jwtBearerOptions =>
{
    jwtBearerOptions.TokenValidationParameters =
        new TokenValidationParameters
        {
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new SymmetricSecurityKey(
                Encoding.UTF8.GetBytes(settings.Key)),
            ValidateIssuer = true,
            ValidIssuer = settings.Issuer,

            ValidateAudience = true,
            ValidAudience = settings.Audience,

            ValidateLifetime = true,
            ClockSkew = TimeSpan.FromMinutes(
                        settings.MinutesToExpiration)
        };
});
```

Tell Web API project to use the authentication configured

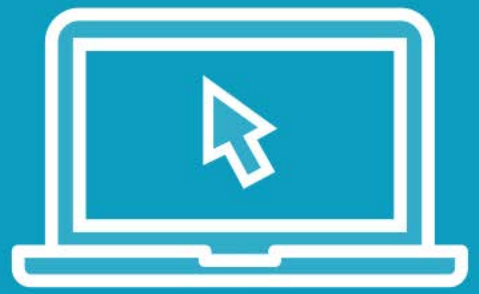
```
public void Configure(IApplicationBuilder app, I
{
  if (env.IsDevelopment())
  {
    app.UseDeveloperExceptionPage();
  }

  app.UseCors(
    options => options.WithOrigins(
      "http://localhost:4200").AllowAnyMethod()
       .AllowAnyHeader()
  );

  app.UseAuthentication();

  app.UseMvc();
}
```

Demo



Setup JWT as authentication method

Add bearer token parameters

Tell Web API to use authentication

Inject JwtSettings Class

Create singleton of JwtSettings class

Allows you to inject settings into the security controller

```
public void ConfigureServices(IServiceCollection services)
{
  // Get JWT Token Settings from JwtSettings.json file
  JwtSettings settings;
  settings = GetJwtSettings();
  // Create singleton of JwtSettings
  services.AddSingleton<JwtSettings>(settings);
```

Inject JwtSettings into security controller

Place into field variable

```
public class SecurityController : Controller
{
    private JwtSettings _settings;
    public SecurityController(JwtSettings settings)
    {
        _settings = settings;
    }
```

Create constructor in security manager

Accept an instance of a JwtSettings class

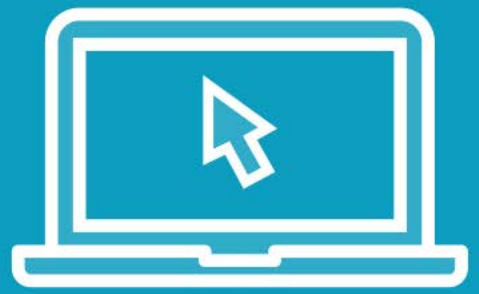
Place into field variable

```
public class SecurityManager
{
  private JwtSettings _settings = null;
  public SecurityManager(JwtSettings settings)
  {
    _settings = settings;
  }
```

Pass settings to security manager within the Login() Web API method

```
[HttpPost("login")]
public IActionResult Login([FromBody]AppUser user)
{
  IActionResult ret = null;
  AppUserAuth auth = new AppUserAuth();
  SecurityManager mgr = new SecurityManager(_settings);
```

Demo



Inject JwtSettings class

Pass JwtSettings to security manager

Build JSON Web Token

Modify SecurityManager.cs

Create BuildJwtToken() method

Create a list of claims

Add subject (sub) and JWT id (jti)

Add claims based on properties

```
protected string BuildJwtToken(AppUserAuth authUser)
{
  SymmetricSecurityKey key = new SymmetricSecurityKey(
    Encoding.UTF8.GetBytes(_settings.Key));

  // Create standard JWT claims
  List<Claim> jwtClaims = new List<Claim>();
  jwtClaims.Add(new Claim(JwtRegisteredClaimNames.Sub,
      authUser.UserName));
  jwtClaims.Add(new Claim(JwtRegisteredClaimNames.Jti,
      Guid.NewGuid().ToString()));

  // Add custom claims
  jwtClaims.Add(new Claim("isAuthenticated",
      authUser.IsAuthenticated.ToString().ToLower()));
  jwtClaims.Add(new Claim("canAccessProducts",
      authUser.CanAccessProducts.ToString().ToLower()));
  jwtClaims.Add(new Claim("canAddProduct",
      authUser.CanAddProduct.ToString().ToLower()));
  jwtClaims.Add(new Claim("canSaveProduct",
      authUser.CanSaveProduct.ToString().ToLower()));
  jwtClaims.Add(new Claim("canAccessCategories",
      authUser.CanAccessCategories.ToString().ToLower()));
  jwtClaims.Add(new Claim("canAddCategory",
      authUser.CanAddCategory.ToString().ToLower()));
```

Create new JWT security token

Add in list of claims

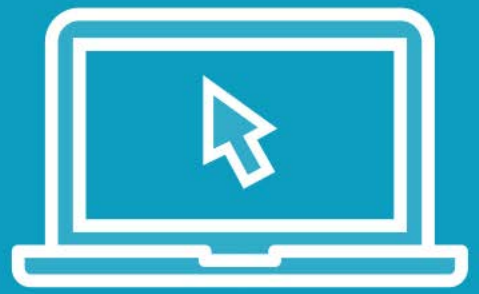
Add in all properties from settings object

Return the new token as a string

```
// Create the JwtSecurityToken object
var token = new JwtSecurityToken(
    issuer: _settings.Issuer,
    audience: _settings.Audience,
    claims: jwtClaims,
    notBefore: DateTime.UtcNow,
    expires: DateTime.UtcNow.AddMinutes(
        _settings.MinutesToExpiration),
    signingCredentials: new SigningCredentials(key,
                SecurityAlgorithms.HmacSha256)
);

// Create a string representation of the Jwt token
return new JwtSecurityTokenHandler().WriteToken(token);
```

Demo



Add method to create token

Summary

Place JWT settings in JSON file

Use a JWT settings class to avoid repetition

For more info on JWT

- Introduction to OAuth2, OpenID Connect and JSON Web Tokens (JWT)

Can use jwt.io to view token info

Coming up in the next module...

How to pass tokens back and forth

Create HTTP interceptor