# What is covered:

1. Using .Net Identity
2. Role Management
3. Policy based authorization
4. UserManager<T>
5. SignInManager<T>
6. RoleManager<T>

# Why?

1. Mature and battle hardened, written and tested by Microsoft
2. Comes with a password hasher with 10,000 salt iterations
   1. Our current token system is only doing single iteration
3. Full framework for managing members and roles
4. Provides an Entity Framework schema to create the needed tables
5. Highly customizable

# Not covered

1. Email confirmation
2. Forgot password

# Setting up Entities

## AppUser.cs

Derive from IdentityUser class

Our Id is int so will need to tell the IdentityUser

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using Microsoft.AspNetCore.Identity;

using OAuth2.WebApi.Core.Extensions;

namespace OAuth2.WebApi.Core.Entities;

public class AppUser : IdentityUser<int>

Then remove the following 4 fields since Identity is taking care of these for us

1. Id
2. UserName
3. PasswordHash
4. PasswordSalt : remove this as well since we don’t need it.

Finally add the property UserRoles. AppUserRole is created below.

    /// <summary>

    /// Added due to identity, acting as a join table

    /// </summary>

    public ICollection<AppUserRole> UserRoles { get; set; }

## AppRole.cs

Create a new entity AppRole.cs in folder Core/Entities folder

using Microsoft.AspNetCore.Identity;

namespace OAuth2.WebApi.Core.Entities;

/// <summary>

/// Derive from IdentityRole and make the Id int

/// One to many relationship,

/// each user can have multiple roles and

/// each role can have multiple users

/// </summary>

public class AppRole : IdentityRole<int>

{

    public ICollection<AppUserRole> UserRoles { get; set; }

}

## AppUserRole.cs

Will join AppUser and AppRole

using Microsoft.AspNetCore.Identity;

namespace OAuth2.WebApi.Core.Entities;

/// <summary>

/// Will join AppUser and AppRole

/// Derive from IdentityUserRole

/// </summary>

public class AppUserRole : IdentityUserRole<int>

{

    public AppUser User { get; set; }

    public AppRole Role { get; set; }

}

# Commenting Error Code (Per Above change)

## Business Logic

### UserBusinessLogic.cs

#### Method RegisterAsync

Comment the code throwing error now

        /\*

        //Changes due to identity implementation

        //hash the password using the CryptoExtension. It will give back hash and the Salt

        var passwordHashKey = registerUser.Password.ComputeHashHmacSha512();

        if (passwordHashKey == null)

            throw new ValidationException("Unable to handle provided password");

        //convert to AppUser to register, ID and GUID will be automatically input by EF

        var appUser = new AppUser()

        {

            UserName = registerUser.UserName.ToLower(), //store as lower case always

            PasswordHash = passwordHashKey.Hash,

            PasswordSalt = passwordHashKey.Salt

        };

        \*/

        var appUser = new AppUser()

        {

            UserName = registerUser.UserName.ToLower(), //store as lower case always

        };

#### Method LoginAsync

        /\*

        Changes due to identity implementation

        //password is hashed in db. Hash login password and check against the DB one

        var passwordHashKey = loginInfo.Password.ComputeHashHmacSha512(appUser.PasswordSalt);

        if (passwordHashKey == null)

            throw new UnauthorizedAccessException("Either username or password is wrong");

        //both are byte[]

        if (!passwordHashKey.Hash.AreEqual(appUser.PasswordHash))

            throw new UnauthorizedAccessException("Either username or password is wrong");

        \*/

## Seed.cs

In Core >> DB folder

        //add password to the users, make username lower case and track users

        foreach (var user in users)

        {

            user.UserName = user.UserName.ToLowerInvariant();

            //removed due to Identity

            //user.PasswordHash = hashKey.Hash;

            //user.PasswordSalt = hashKey.Salt;

            //we are only adding tracking to the user, save changes will happen outside of the loop

            context.Users.Add(user);

        }

# Setting up DataContext.cs

## Add Package

Package : Microsoft.AspNetCore.Identity.EntityFrameworkCore

Check [0001 Project Setup.docx](0001%20Project%20Setup.docx) section Adding Packages

## DataContext.cs

In Core >> DB folder

Instead of deriving from Dbcontext, derive from IdentityDbContext

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

using OAuth2.WebApi.Core.Entities;

namespace OAuth2.WebApi.Core.DB;

/// <summary>

/// DataContext Class, add as a service to program.cs

/// </summary>

//Change after identity implementation

//public class DataContext : DbContext

public class DataContext : IdentityDbContext<AppUser,

                                            AppRole,

                                            int,

                                            IdentityUserClaim<int>,

                                            AppUserRole,

                                            IdentityUserLogin<int>,

                                            IdentityRoleClaim<int>,

                                            IdentityUserToken<int>>

Comment out the DbSet for the AppUser as that is not needed any more.

    //Removed after implementing Identity as not needed

    //public DbSet<AppUser> Users { get; set; }

Create new private method to create relationships for AppUser and AppRole

    /// <summary>

    /// User roles due to use of Identity

    /// </summary>

    /// <param name="modelBuilder"></param>

    private void CreateUserRole(ModelBuilder modelBuilder)

    {

        modelBuilder.Entity<AppUser>()

                .HasMany(ur => ur.UserRoles)

                .WithOne(u => u.User)

                .HasForeignKey(ur => ur.UserId)

                .IsRequired()

        ;

        modelBuilder.Entity<AppRole>()

                .HasMany(ur => ur.UserRoles)

                .WithOne(u => u.Role)

                .HasForeignKey(ur => ur.RoleId)

                .IsRequired()

        ;

    }

Add the new private method to OnModelCreating

    protected override void OnModelCreating(ModelBuilder modelBuilder)

    {

        base.OnModelCreating(modelBuilder);

        //AppUser has guid that needs to be autogenerated for insert

        //This isn't working, in AppUser create the guid

        /\*

        modelBuilder.Entity<AppUser>()

        .Property(x => x.Guid)

        .ValueGeneratedOnAdd();

        \*/

        //make DisplayName Unique

        modelBuilder.Entity<AppUser>(entity =>

        {

            entity.HasIndex(e => e.DisplayName).IsUnique();

        });

        //Due to use of Identity

        CreateUserRole(modelBuilder);

    }