Creating ASP.NET ROLES

FILE PROGRAM.CS

REPLACE THIS CODE :

builder.Services.AddIdentity<IdentityUser, IdentityRole>(options => options.SignIn.RequireConfirmedAccount = true)

                .AddEntityFrameworkStores<ApplicationDbContext>()

                .AddDefaultTokenProviders()

                .AddDefaultUI();

Add this in :

using (var scope = app.Services.CreateScope())

{

    var roleManager = scope.ServiceProvider.GetRequiredService<RoleManager<IdentityRole>>();

    var roles = new[] { "Admin", "Manager", "Member" };

    foreach (var role in roles)

    {

        if (!await roleManager.RoleExistsAsync(role))

        {

            await roleManager.CreateAsync(new IdentityRole(role));

        }

    }

}

builder.Services.AddAuthorization(options =>

{

    options.AddPolicy("Admin", policy => policy.RequireRole("Admin"));

    options.AddPolicy("Manager", policy => policy.RequireRole("Manager"));

    options.AddPolicy("Member", policy => policy.RequireRole("Member"));

});

EXAMPLE OF WHAT THE CODE SHOULD LOOK LIKE

using Microsoft.AspNetCore.Identity;

using Microsoft.EntityFrameworkCore;

using ProgIceTask.Data;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection")

?? throw new InvalidOperationException("Connection string 'DefaultConnection' not found.");

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(connectionString));

builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddIdentity<IdentityUser, IdentityRole>(options =>

options.SignIn.RequireConfirmedAccount = true)

.AddEntityFrameworkStores<ApplicationDbContext>()

.AddDefaultTokenProviders()

.AddDefaultUI();

builder.Services.AddAuthorization(options =>

{

options.AddPolicy("Admin", policy => policy.RequireRole("Admin"));

options.AddPolicy("Manager", policy => policy.RequireRole("Manager"));

options.AddPolicy("Member", policy => policy.RequireRole("Member"));

});

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseMigrationsEndPoint();

}

else

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

using (var scope = app.Services.CreateScope())

{

var roleManager = scope.ServiceProvider.GetRequiredService<RoleManager<IdentityRole>>();

var userManager = scope.ServiceProvider.GetRequiredService<UserManager<IdentityUser>>();

var roles = new[] { "Admin", "Manager", "Member" };

foreach (var role in roles)

{

if (!roleManager.RoleExistsAsync(role).GetAwaiter().GetResult())

{

roleManager.CreateAsync(new IdentityRole(role)).GetAwaiter().GetResult();

}

}

// Create an Admin user if it doesn't exist

string adminEmail = "admin@example.com"; // Replace with your admin email

string adminPassword = "Admin@123"; // Replace with a secure password

if (userManager.FindByEmailAsync(adminEmail).GetAwaiter().GetResult() == null)

{

var adminUser = new IdentityUser { UserName = adminEmail, Email = adminEmail };

var result = userManager.CreateAsync(adminUser, adminPassword).GetAwaiter().GetResult();

if (result.Succeeded)

{

userManager.AddToRoleAsync(adminUser, "Admin").GetAwaiter().GetResult();

}

}

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication(); // Ensure authentication middleware is used

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.MapRazorPages();

app.Run();

ISSUES FIXING :

IF YOU GETTING A RUNNIGN ERROR , TRY MIGRATION AND UPDATIN ,

Add-migraiton

If adding is giving you an issue try

Add-migration -context ApplicationDbConext

Then :

Update-database

IF YOU SIGNING IN AND THERES STILL ISSUES WITH THE LOGIN PROCESS TRY THIS

builder.Services.AddIdentity<IdentityUser, IdentityRole>(options =>

options.SignIn.RequireConfirmedAccount = false)

.AddEntityFrameworkStores<ApplicationDbContext>()

.AddDefaultTokenProviders()

.AddDefaultUI();

SET THAT TO FALSE SO YOU DON THAVE TO VERIFY THE ACCOUNT

IF ALL ELSE FAILS, PUT IT INTO CHAT AND ASK WHAT TO DO