***// each of these import declarations should be commented –***

***// - what are we importing from each module?***

***// what do each of the imported functions do, in one phrase?***

***//***

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Hosting;

using Microsoft.EntityFrameworkCore;

using CPH\_IVT.Data;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using CPH\_IVT.Models;

using CPH\_IVT.Services.MongoDB.Repository;

using CPH\_IVT.Services.MongoDB.Init;

using CPH\_IVT.Services.MongoDB.Context;

namespace CPH\_IVT

{

/// <summary>

/// Provides configuration for CPH-IVT project.

/// ***so, what all does this configuration operation entail?***

/// Boilerplate provided by ASP.NET Core.

/// </summary>

public class Startup

{

/// <summary>

/// Parameterized constructor.

/// ***again, so what do we need to know about what’s going on?***

/// </summary>

/// <param name="configuration"><see cref="IConfiguration"/></param>

// ***all params should be documented inline, rather than just referenced***

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

/// <summary>

/// <see cref="IConfiguration"/>

/// ***more pseudo-documentation; this is not useful***

/// ***I assume that the configuration dataset is read-only, once defined – hence, just the getter;***

/// ***but is that so? A note to this effect would be helpful***

/// </summary>

public IConfiguration Configuration { get; }

/// <summary>

/// Adds services to the DI and/or IOC containers at runtime.

/// ***more pseudo-documentation; this is not useful***

/// ***what is a DI container? An IOC container?***

/// ***what is an EntityFrameworkStore?***

/// ***where is ApplicationDbContext defined?***

/// </summary>

/// <param name="services"><see cref="IServiceCollection"/></param>

// ***all params should be documented inline, rather than just referenced***

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<ApplicationDbContext>(options => options.UseSqlServer(Configuration.GetConnectionString("SQLConnection")));

services.AddDefaultIdentity<IdentityUser>(

options => options.SignIn.RequireConfirmedAccount = true).AddEntityFrameworkStores<ApplicationDbContext>();

services.AddMvc();

services.Configure<HealthDatabaseSettings>(

options =>

{

options.ConnectionString = Configuration.GetSection("HealthDatabaseSettings:ConnectionString").Value;

options.DatabaseName = Configuration.GetSection("HealthDatabaseSettings:DatabaseName").Value;

});

// ***what are Transients?***

// Transients for all contexts

//

services.AddTransient<ICensusDivisionContext, CensusDivisionContext>();

services.AddTransient<ICensusRegionContext, CensusRegionContext>();

services.AddTransient<ICountyContext, CountyContext>();

services.AddTransient<ICustomRegionContext, CustomRegionContext>();

services.AddTransient<IHealthIndicatorContext, HealthIndicatorContext>();

services.AddTransient<IStateContext, StateContext>();

// Transients for all repositories

//

services.AddTransient<ICensusDivisionRepository, CensusDivisionRepository>();

services.AddTransient<ICensusRegionRepository, CensusRegionRepository>();

services.AddTransient<ICountyRepository, CountyRepository>();

services.AddTransient<ICustomRegionRepository, CustomRegionRepository>();

services.AddTransient<IHealthIndicatorRepository, HealthIndicatorRepository>();

services.AddTransient<IStateRepository, StateRepository>();

// ***what is this scoping doing, exactly?***

// Scoped for database initialization

services.AddScoped<InitializerAsync>();

// ***from where are the controllers with views added? Does this have anything to do with the “Transients”?***

services.AddControllersWithViews();

// ***what is a Razor Page? From where are these pages added?***

services.AddRazorPages();

}

/// <summary>

/// Configures the HTTP request pipeline at runtime.

/// </summary>

/// <param name="app"><see cref="IApplicationBuilder"/></param>

/// <param name="env"><see cref="IWebHostBuilder"/></param>

// ***odd – why is an IApplicationBuilder param named app, not appBuilder?***

// ***all params should be documented inline, rather than just referenced***

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

// ***what do these “use” methods do, exactly?***

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseDatabaseErrorPage();

}

else

{

// ***is “/Home/Error” a well-known URI?***

// ***what is HSTS??***

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.

// TODO: Remove the above comment before deployment.

app.UseHsts();

}

// ***what exactly is HttpsRedirection? Is that Http --> Https?***

app.UseHttpsRedirection();

// ***what are these “static files”?***

app.UseStaticFiles();

// ***where is routing established, exactly? Is there a routes table somewhere in this collection of codes?***

app.UseRouting();

// ***what datasets do authentication and authorization key off***

app.UseAuthentication();

app.UseAuthorization();

// ***what are these statements doing?***

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute( name: "default", pattern: "{controller=Home}/{action=Index}/{id?}");

endpoints.MapRazorPages();

});

}

}

}