Seeding A Database

Seeding a database is the process in which an initial set of data is provided to a database when it is being installed. This is useful when we want to populate a database with data (and is often an automated process that is executed upon the initial setup of an application.

Adding The SeedData class

In your MVC .NET Application, create a new class (in your Models folder) called SeedData.cs Replace the code with the following:

```
public static void Initialize(IServiceProvider serviceProvider)
{
    using (var context = new DogBreedContext(
        serviceProvider.GetRequiredService<
            DbContextOptions<DogBreedContext>>()))
    {
        // Look for any movies.
        if (context.DogBreeds.Any())
        {
            return;   // DB has been seeded
        }

        context.DogBreeds.AddRange(
            new DogBreed
            {
                Name = "TestBreed",
                Temperament = "Nice",
                EKCGroup = "Test",
                AKCGroup = "Test2",
                FCIGroup = "Test3",
                CountryOfOrigin = "TestCountry",
                Size = "Large",
                AverageLifespan = 1,
                HeightInches = 1,
                WeightPounds =1,
            },

            new DogBreed
            {
                Name = "TestBreed2",
                Temperament = "Naughty",
                EKCGroup = "Test",
                AKCGroup = "Test2",
                FCIGroup = "Test3",
                CountryOfOrigin = "TestCountry",
                Size = "Small",
                AverageLifespan = 2,
                HeightInches = 2,
                WeightPounds = 2,
            }
        ) ;
        context.SaveChanges();
    }
}
```

Make sure you refer to the context you created.

If there are any dogs in the database, then the seeding does not need to take place.

Here you include the new data objects you want to be entered into your database.

This code is provided as a text file so you can copy and paste, but you need to include using directives as required:

```
using Microsoft.
EntityFrameworkCore;

using Microsoft.Extensions.
DependencyInjection;

using YourApplication.Data;

using System;

using System.Linq;.
```

Add The Seed Initialiser

Replace the contents of Program.cs with the following code.

```
public static void Main(string[] args)
{
    var host = CreateHostBuilder(args).Build();

    using (var scope = host.Services.CreateScope())
    {
        var services = scope.ServiceProvider;

        try
        {
            SeedData.Initialize(services);
        }
        catch (Exception ex)
        {
            var logger = services.GetRequiredService<ILogger<Program>>();
            logger.LogError(ex, "An error occurred seeding the DB.");
        }
    }

    host.Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>
    Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
        {
            webBuilder.UseStartup<Startup>();
        });
```

This is also provided as a text file so you can copy and paste.
No changes should be required (just make sure you include the using directives as shown).

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using YourApplication.Models;
using System;
```

Run your application.
Note: The database must be empty for the seeding process to work when the application is run.