# Assignment 2 Guide: Creating the Database

Select ***Tools* ⇨ *Command Line* ⇨ *Developer PowerShell***

* When you open the Developer PowerShell wait for the prompt to appear
* You must change to the Project folder (i.e with ***Startup.cs*** and ***Program.cs***) to add the packages
  + ***dir*** will list the files/folders
  + ***cd*** will change directory
  + ***cd..*** will change up one directory level

If you are not in the ***KoalaBeach*** Solution folder (it should start where you left off last time)

**cd KoalaBeach**

Do a ***dir*** to check if you are in the ***project*** folder (may be different for your project). If not

**cd KoalaBeach**

First add ***Entity Framework Core*** to the project (copy then just right click in the PowerShell window)

**dotnet add package Microsoft.EntityFrameworkCore.Design --version 3.1.1**

**dotnet add package Microsoft.EntityFrameworkCore.SqlServer --version 3.1.1**

If you do not end with the prompt displayed press Enter

Then add the ***Entity Framework Core Tools*** so we can create, edit and update the database

**dotnet tool uninstall --global dotnet-ef**

**dotnet tool install --global dotnet-ef --version 3.1.1**

Add the connection string to **appsetting.json**

"AllowedHosts": "\*"

**"ConnectionStrings": {**

**" KoalaBeachConnection": "Server=(localdb)\\MSSQLLocalDB;Database=KoalaBeach;MultipleActiveResultSets=true"**

**}**

Add the class file ***StoreDbContext.cs*** to the ***Models*** folder

**using Microsoft.EntityFrameworkCore;**

**namespace KoalaBeach.Models {**

**public class StoreDbContext : DbContext {**

**public StoreDbContext(DbContextOptions<StoreDbContext> options)**

**: base(options) { }**

**public DbSet<Product> Products { get; set; }**

**}**

**}**

Register the db context in ***Startup.cs***

// copy this line only if not already present

using Microsoft.Extensions.Configuration;

**using Microsoft.EntityFrameworkCore;**

**using KoalaBeach.Models;**

namespace KoalaBeach {

    public class Startup {

        // copy this only if not already present

public Startup(IConfiguration config) {

            Configuration = config;

        }

        // copy this only if not already present

public IConfiguration Configuration { get; set; }

        public void ConfigureServices(IServiceCollection services) {

services.AddControllersWithViews();

**services.AddDbContext<StoreDbContext>(opts => {**

**opts.UseSqlServer(**

**Configuration["ConnectionStrings: KoalaBeachConnection"]);**

**});**

        }

Add the class file ***Product.cs*** to the ***Models*** folder

**using System;**

**using System.ComponentModel.DataAnnotations.Schema;**

**namespace KoalaBeach.Models {**

**public class Product {**

**public long ProductID { get; set; }**

**public string Name { get; set; }**

**public string Description { get; set; }**

**[Column(TypeName = "decimal(8, 2)")]**

**public decimal Price { get; set; }**

**public string Category { get; set; }**

**public string SubCategory { get; set; }**

**public String Image { get; set; }**

**public bool Sale { get; set; } = false;**

**}**

**}**

Add the class file ***IStoreRepository.cs*** to the ***Models*** folder

**using System.Linq;**

**namespace KoalaBeach.Models {**

**public interface IStoreRepository {**

**IQueryable<Product> Products { get; }**

**// void SaveProduct(Product p);**

**// void CreateProduct(Product p);**

**// void DeleteProduct(Product p);**

**}**

**}**

Add the class file ***EFStoreRepository.cs*** to the ***Models*** folder

**using System.Linq;**

**namespace KoalaBeach.Models {**

**public class EFStoreRepository : IStoreRepository {**

**private StoreDbContext context;**

**public EFStoreRepository(StoreDbContext ctx) {**

**context = ctx;**

**}**

**public IQueryable<Product> Products => context.Products;**

**}**

**}**

Modify ***Startup.cs***

public class Startup {

...

public void ConfigureServices(IServiceCollection services) {

services.AddControllersWithViews();

services.AddDbContext<StoreDbContext>(opts =>

opts.UseSqlServer(Configuration[“ConnectionStrings: KoalaBeachConnection"]));

**services.AddScoped<IStoreRepository, EFStoreRepository>();**

}

## Migrate the database

Note: You MUST save all files before the migration

* Open the Developer PowerShell and change to the project folder
* Run the EF migration command to create the migration file

**dotnet ef migrations add Initial**

* Check that the Migration folder was created in the project
* Check the initial migration file was created and the database Products table code is in the Up method
* If not, then you did not save the files before doing the migration (save and repeat migration)

## Create the database

* In the PowerShell run the command below to create the database from the migration file
* Note that if you do not run this it will be done by the seed file when first run
* Better to do it here you can check the response from the command to see if it was successful

**dotnet ef database update**

## Check the database has been created

* Open ***View => SQL Server Object Explorer***
* Expand **SQL** ***Server (localdb)MSSQLLocalDB*** … and then expand ***Databases***
* If KoalaBeach is not there right click on ***SQL Server (localdb)MSSQLLocalDB*** …
* Select ***Refresh***

Sometimes the SQL Server Object Explorer tab will be there but there are no databases or connections

* Close the tab and go to View => SQL Server Object Explorer
* Or run the app (do not worry of the app fails)

## Now add the SeedData

Add the class file ***SeedData.cs*** to the ***Models*** folder

**using System.Linq;**

**using Microsoft.AspNetCore.Builder;**

**using Microsoft.Extensions.DependencyInjection;**

**using Microsoft.EntityFrameworkCore;**

**namespace KoalaBeach.Models**

**{**

**public static class SeedData**

**{**

**public static void EnsurePopulated(IApplicationBuilder app)**

**{**

**StoreDbContext context = app.ApplicationServices**

**.CreateScope().ServiceProvider.GetRequiredService<StoreDbContext>();**

**if (context.Database.GetPendingMigrations().Any())**

**{**

**context.Database.Migrate();**

**}**

**if (!context.Products.Any())**

**{**

**context.Products.AddRange(**

**new Product**

**{**

**Name = "T Shirt",**

**Description = "Groovy T Shirt",**

**Category = "Shirt",**

**SubCategory = "Men",**

**Price = 15**

**},**

**new Product**

**{**

**Name = "Single Top",**

**Description = "Singlet style top",**

**Category = "Shirt",**

**SubCategory = "Men",**

**Price = 20**

**}**

**);**

**context.SaveChanges();**

**}**

**}**

**}**

**}**

In ***Startup.cs*** run seed data in the configuration

public class Startup

{

...

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

...

app.UseEndpoints(endpoints => {

endpoints.MapDefaultControllerRoute();

});

**SeedData.EnsurePopulated(app);**

}

Then run the web site Note that the database server is only running while the app is running

* If you get a login or connection error it may simply be due to the database not being created

See the ***Deleting the Database.doc*** for details of how to use SQL Server Explorer to check, edit and delete the database

## Create a page to test the database

Add the Razor View file ***Test.cshtml*** to the ***Views*** folder

**@model IQueryable<Product>**

**@foreach (var p in Model)**

**{**

**<div>**

**<h3>@p.Name</h3>**

**@p.Description**

**<h4>@p.Price.ToString("c")</h4>**

**</div>**

**}**

Update ***HomeController.cs*** to inject the repository and call the test page

public class HomeController : Controller

{

**private IStoreRepository repository;**

**public HomeController(IStoreRepository repo){**

**repository = repo;**

**}**

**public IActionResult Test()**

**=> View(repository.Products);**

Then run the web site again to test the products are displayed

* Use ***https://localhost:[your port]/home/test***

