

2. Code-First Database Migrations

Introduction

In this exercise you will generate the database from the entity data model classes that you have created.

Make sure you have created all the [models](#) and the [relationships](#).

Database Context Class

Database Context Class

The Database context class maps to the database the application is connecting to. It represents the application's session with the database.

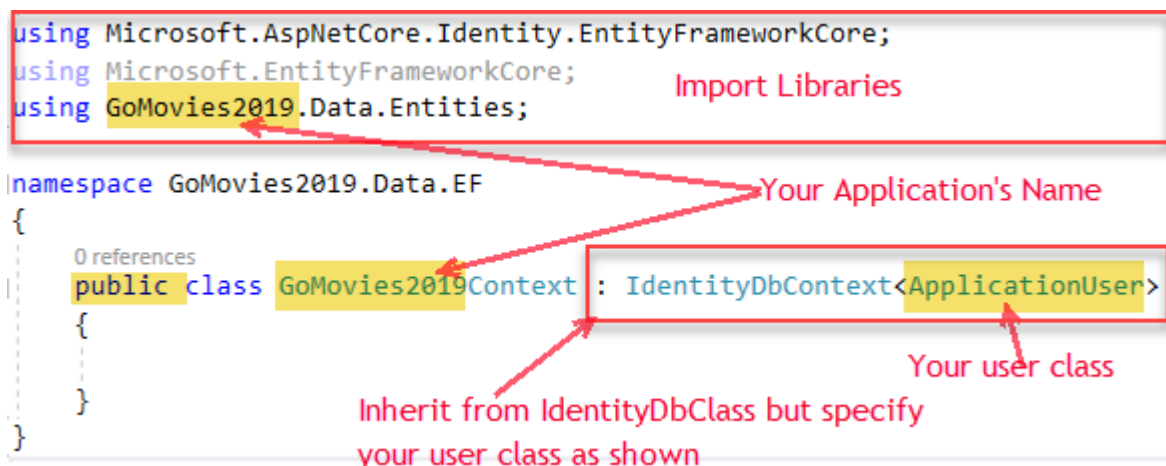
- ♦ Open up your solution in VS
- ♦ Locate the **EF folder** in the **.Data project**
- ♦ Right-click on the **EF folder** and add a new class called **YOUR_APP_NAMEContext**
Replace YOUR_APP_NAME with the name of your application e.g.
GoMovies2019Context.
- ♦ Import the following libraries at the top of the class:
Microsoft.AspNetCore.Identity.EntityFrameworkCore
Microsoft.EntityFrameworkCore
YOUR_APP_NAME.Data.Entities
- ♦ Make the class public and make it inherit from
IdentityDbContext<YOUR_APPLICATION_USER_CLASS_NAME>

Replace YOUR_APPLICATION_USER_CLASS_NAME with the name of your user class in the entities folder and remember that C# is case-sensitive.

See example screenshot below.

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;  
using Microsoft.EntityFrameworkCore;  
using GoMovies2019.Data.Entities;
```

```
namespace GoMovies2019.Data.EF  
{  
    public class GoMovies2019Context : IdentityDbContext<ApplicationUser>  
    {  
    }  
}
```



Database Context Class

Copy and paste the following code into the class

```
public class GoMovies2019Context : IdentityDbContext<ApplicationUser>
{
    // Constructor Methods
    public GoMovies2019Context() { }

    public GoMovies2019Context(DbContextOptions<GoMovies2019Context> options) : base(options)
    {
    }

    // Add the Tables (Database Sets) you want to include in the database
    public DbSet<TicketType> TicketTypes { get; set; }

    // TODO: Add the remaining database sets

    protected override void OnModelCreating(ModelBuilder builder)
    {
        // Specify table names (singular) for entities
        builder.Entity<TicketType>().ToTable("TicketType");

        //TODO: specify table names for other entities

        // Specify Composite Key for Booking Ticket

        base.OnModelCreating(builder);
    }
}
```

Database Context Class: Constructor Method

Edit the constructor method as shown below, replacing the highlighted with the **name of the DatabaseContext class (current class name)**.

```
public class GoMovies2019Context : IdentityDbContext<ApplicationUser>
{
    // Constructor Method
    0 references | 0 exceptions
    public GoMovies2019Context() { }

    1 reference | 0 exceptions
    public GoMovies2019Context(DbContextOptions<GoMovies2019Context> options) : base(options)
    {
        The highlighted names should be the same (remember C# is case-sensitive)
    }
}
```

Database Context Class: Add the Database Sets (Tables)

Add the Database Sets (Tables)

The database sets represent each of the data entity models (i.e. tables) that you want to include in the Database Context (i.e. database).

- ♦ I have shown the first database set (TicketTypes) for you. **Add the remaining database sets** for the following entities to include in the database.
- ♦ **Note the following** when adding the database sets

You do not need to add the ApplicationUser class - .NET will automatically add all the classes needed for user accounts for you.

The convention is to specify the table property names in plural form e.g. **TicketTypes**.

```
public class GoMovies2019Context : IdentityDbContext<ApplicationUser>
{
    0 references | 0 exceptions
    public GoMovies2019Context(DbContextOptions<GoMovies2019Context> options) : base(options)
    {
        // Add the Tables (Database Sets) you want to include in the database
        0 references | 0 exceptions
        public DbSet<TicketType> TicketTypes { get; set; } ← Example
        // TODO: Add the remaining database sets ← Plural
        ← Add the other DbSet here
```

Entity	Property Name (note the property name is plural)
TicketType	TicketTypes
Film	Films
FilmShowing	FilmShowings
Booking	Bookings
BookingTicket	BookingTickets

Database Context Class: OnModelCreating Method Singular Table Names

OnModelCreating Method: Singular Table Names

The OnModelCreating method is used to specify any additional settings when the database is being created.

- While the convention is to use plural names for database set properties, the convention is to use singular names for database tables.

You use the syntax below to specify the table name for each of the entities. If you want to use different names (e.g. tblFilm), you can also do this here.

- Add the OnModelCreating method as shown and complete the table names for the following entities (I have shown the first for you).

TicketType	◦
Film	◦
FilmShowing	◦
Booking	◦
BookingTicket	◦

```
protected override void OnModelCreating(ModelBuilder builder)
{
    // Specify table names (singular) for entities
    builder.Entity<TicketType>().ToTable("TicketType"); ← Example
    //TODO: specify table names for other entities
    ← Add others here
    // Specify Composite Key for Booking Ticket
    base.OnModelCreating(builder);
}
```

Database Context Class: OnModelCreating Method Composite

OnModelCreating Method: Composite Keys

If you have any composite keys (Primary Keys made up of more than one column), you need to specify these in the OnModelCreating method.

We have one in the BookingTicket table (PK consists of BookingId and TicketTypeId).

Add this as shown below in the **OnModelCreating method**:

```
protected override void OnModelCreating(ModelBuilder builder)
{
    // Specify table names (singular) for entities
    builder.Entity<TicketType>().ToTable("TicketType");

    // ... Other code not shown ...

    // Specify Composite Key for Booking Ticket
    builder.Entity<BookingTicket>().HasKey(t => new { t.BookingId, t.TicketTypeId });

    base.OnModelCreating(builder);
}
```

The Connection String

The Connection String

The connection string is how the application knows which database to connect to and it is stored in the appsettings.json file in the web project.

- ♦ Open up the appsettings.json file.
- ♦ By default the database name is quite long, **simplify this to a name of your choice and make note of the name**. For example, I have simplified mine to GoMovies2019

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=GoMovies2019;Trusted_Connection=True;MultipleActiveResultSets=true"
  },
}
```

- ♦ By default, VS will create the database file in the users folder. This, however, makes it difficult if your application is on a USB and you use different PCs e.g. at home and at Uni.
- ♦ We are going to make it create the database in the application's solution folder.
- ♦ **Add the following as shown** in the screenshot below replacing your chosen database name too.

AttachDBFilename=|DataDirectory|\\GoMovies2019.mdf;

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=GoMovies2019;AttachDBFilename=|DataDirectory|\\GoMovies2019.mdf;Trusted_Connection=True"
  },
}
```

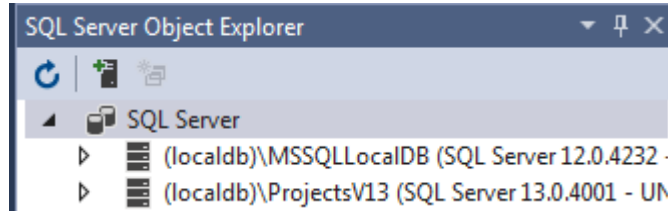
Your chosen database name

Creating the Database

You are going to use a local database stored in the **Data folder in the .Data project**

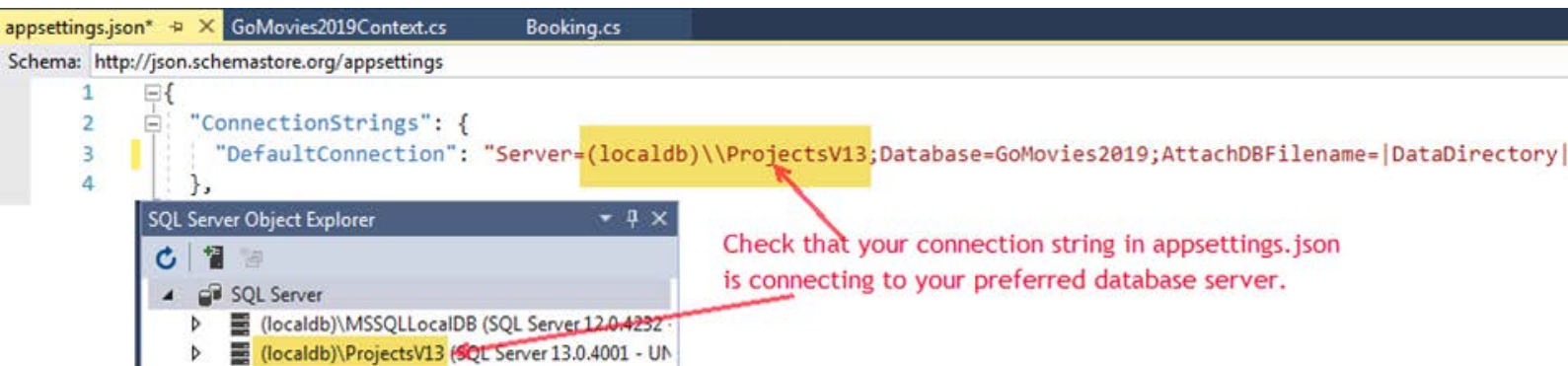
Creating the Databases

- In Visual Studio, from the menu go to View ? SQL Server Object Explorer.
- This will connect to the database engine on your computer.



- If you have multiple SQL Servers installed (you can skip to next slide if you have just one), choose the one to use (preferably the latest version). In the screenshot above, there are two so I am choosing (localdb)ProjectsV13.

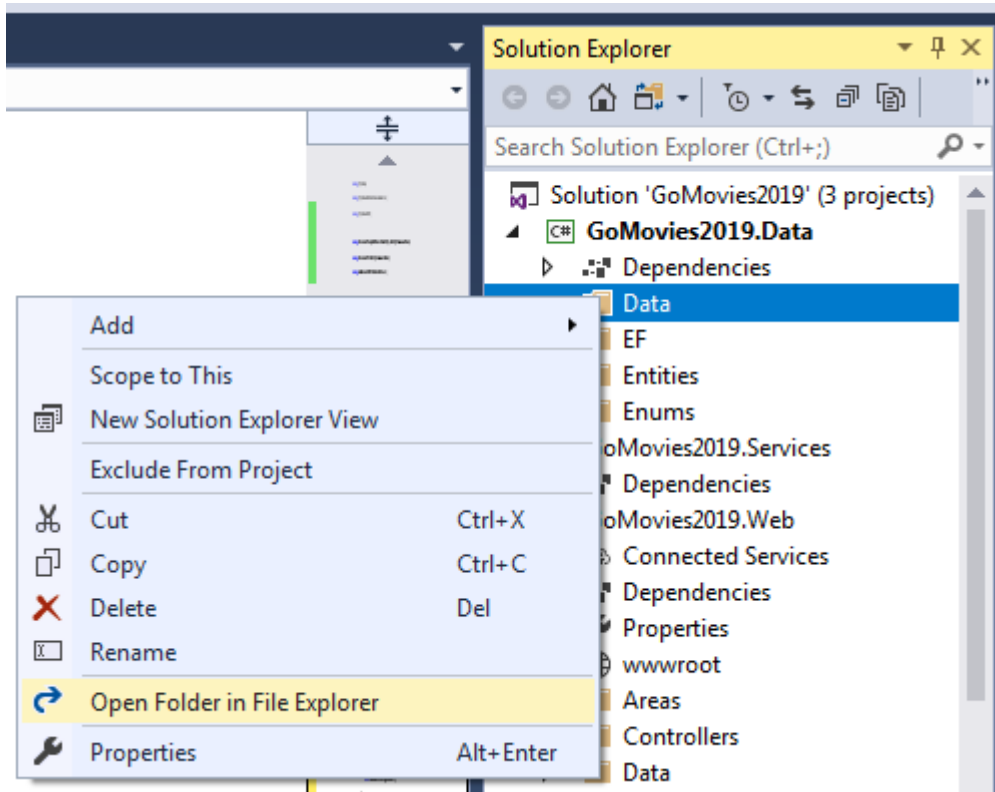
Note: You will need to update the connectionstring in appsettings too. **Use double-backslash \\ for the backslash.**



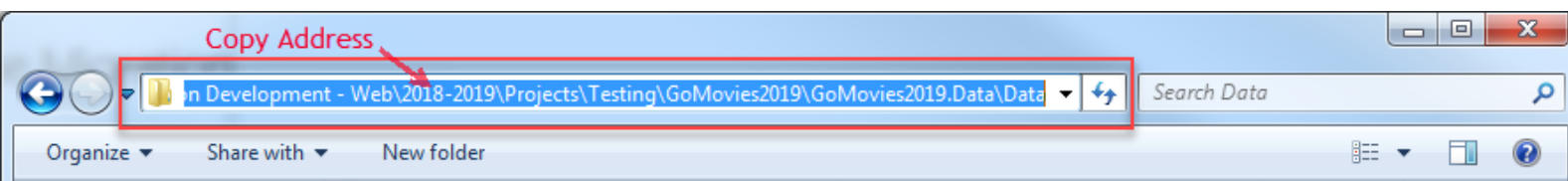
Creating the Database

You are going to use a local database stored in the **Data folder in the .Data project**

Right-click on the **Data folder in the .Data project** and select Open Folder in File Explorer.



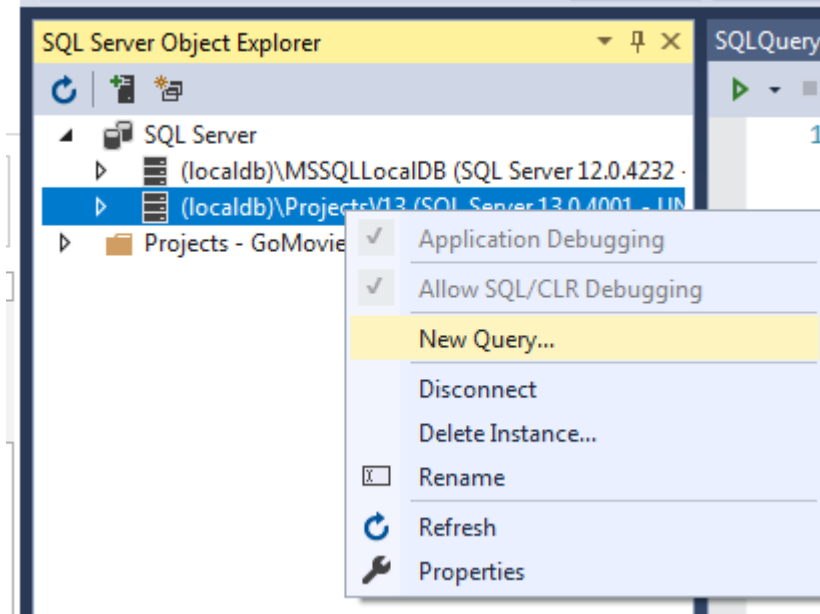
This should open a file explorer window, copy the address as shown



- ♦ Go back to VS
- ♦ In SQL Server Object Explorer Window, right-click on the database and click New Query...
- ♦ **Temporarily paste** the copied address in the window as a comment - you will use it shortly.

Creating the Database

You are going to use a local database stored in the **Data** folder in the **.Data** project



```
SQLQuery1.sql *  appsettings.json*  GoMovies2019Context.cs  Booking.cs
master
1  -- E:\UWTSD\Modules\L5 - Application Development - Web\2018-2019\Projects\Testing\GoMovies2019\GoMovies2019.Data\Data
2
```

Creating the Database

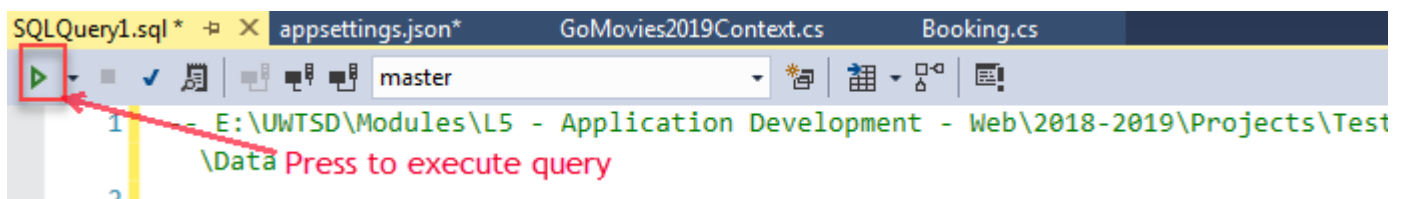
Copy and paste the following command into the new query window and edit the database name and file location to your preferred name as shown:

```
create database GoMovies2019 on (name='GoMovies2019',  
filename='E:\...\GoMovies2019\GoMovies2019.Data\Data\GoMovies2019.mdf');
```

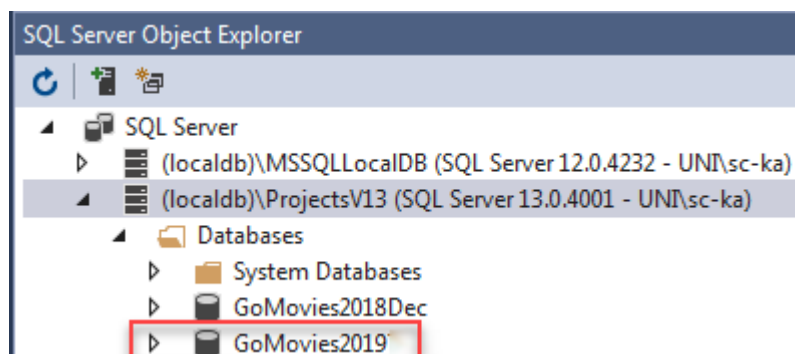


The diagram shows the SQL command with red arrows pointing to specific parts: 'GoMovies2019' is labeled 'Database Name', and the file path 'E:\UWTSD\Projects\Testing\GoMovies2019\GoMovies2019.Data\Data\GoMovies2019.mdf' is labeled 'File Location'.

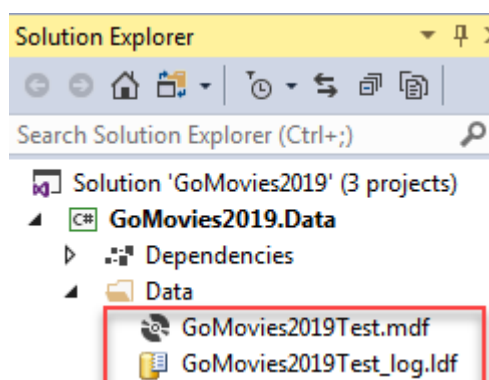
Press the little green arrow to execute the query.



You can expand the database window to see the newly created database (no tables yet).



You can also view the files in the solution explorer window



Boilerplate Code: Context Factory

- Open up your solution in VS
- Locate the **EF folder** in the **.Data project**
- Right-click on the **EF folder** and add a new class called **YOUR_APP_NAMEContextFactory**
- Import the following libraries

```
Microsoft.EntityFrameworkCore;  
Microsoft.EntityFrameworkCore.Design;
```

- Make the class **public** and make it inherit from **IDesignTimeDbContextFactory<YOUR_APPLICATION_USER_CLASS_NAME>** as shown
- don't worry about the error, you will fix this next.

```
using Microsoft.EntityFrameworkCore;  
using Microsoft.EntityFrameworkCore.Design;
```

```
namespace GoMovies2019.Data.EF  
{  
    public class GoMovies2019ContextFactory : IDesignTimeDbContextFactory  
    <GoMovies2019Context>  
    {  
    }  
}
```

```
using Microsoft.EntityFrameworkCore;  
using Microsoft.EntityFrameworkCore.Design;
```

← Import libraries

```
namespace GoMovies2019.Data.EF
```

```
{  
    0 references  
    public class GoMovies2019ContextFactory : IDesignTimeDbContextFactory<GoMovies2019Context>  
    {  
    }  
}
```

Your Database Context Class i.e.
the other class in the EF folder

Boilerplate Code: Context Factory

- ♦ **Copy and paste** the following code into the ContextFactory class

```
public class GoMovies2019ContextFactory : IDesignTimeDbContextFactory<GoMovies2019Context>
{
    private const string connStr = "...ADD YOUR CONNECTION STRING HERE...";

    public GoMovies2019Context CreateDbContext(string[] args)
    {
        var builder = new DbContextOptionsBuilder<GoMovies2019Context>();
        builder.UseSqlServer(connStr,
            b => b.MigrationsAssembly("GoMovies2019.Data"));

        return new GoMovies2019Context(builder.Options);
    }
}
```

Boilerplate Code: Context Factory

Make the following changes to the code copied in the previous step:

- ♦ Open your **appsettings.json** file and copy the connection string
- ♦ Edit the following line as shown in the screenshots

```
private const string connStr = "...ADD YOUR CONNECTION STRING HERE...";
```

```
public class GoMovies2019ContextFactory : IDesignTimeDbContextFactory<GoMovies2019Context>
```

```
{
```

```
    private const string connStr = "Server=(localdb)\n    \ProjectsV13;Database=GoMovies2019Test;AttachDBFilename=|DataDirectory|\n    \GoMovies2019Test.mdf;Trusted_Connection=True;MultipleActiveResultSets=true";
```

Copy connection string and delete the AttachDBfilename key/value option as highlighted

- ♦ Update the rest of the code (the highlighted options shown below) to reflect your application i.e.

```
public GoMovies2019Context CreateDbContext(string[] args)
```

```
{
```

```
    var builder = new DbContextOptionsBuilder<GoMovies2019Context>();
```

```
    builder.UseSqlServer(connStr,
```

```
        b => b.MigrationsAssembly("GoMovies2019.Data"));
```

Name of your .Data Project

```
    return new GoMovies2019Context(builder.Options);
```

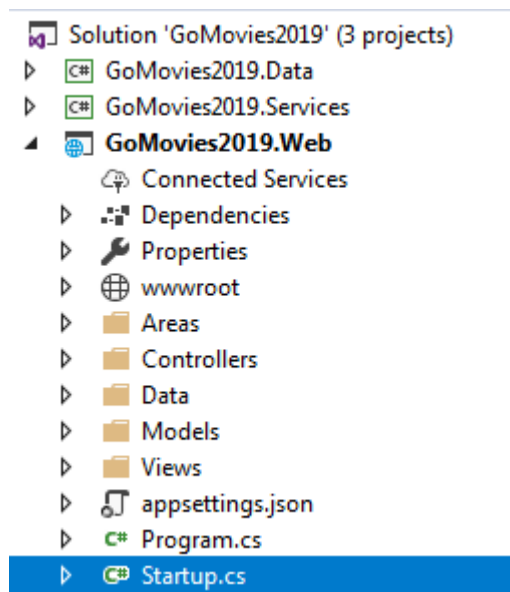
```
}
```

Name of your Database Context Class

Updating Startup

Updating StartUp

Locate the **startup.cs** file in the Web project and open it.



Updating Startup

Import the following libraries (replacing `YOUR_APP_NAME` as before)

- ♦ `YOUR_APP_NAME.Data.EF`
- ♦ `YOUR_APP_NAME.Data.Entities`

```
using GoMovies2019.Data.EF;  
using GoMovies2019.Data.Entities;
```

Locate the **Startup method** and copy and paste the following highlighted code and parameter (lines 1 and 5-7) - see screenshot for explanation.

The code is creating a configuration setting that stores the location of the data folder in your solution.

```
public Startup(IConfiguration configuration, IHostingEnvironment env)  
{  
    Configuration = configuration;  
  
    // Create data directory setting which points to the Data folder in t  
    he YOUR_APP.Data\Data.  
    string dirDataProject = env.ContentRootPath.Replace(".Web", ".Data",  
StringComparison.CurrentCultureIgnoreCase);  
    AppDomain.CurrentDomain.SetData("DataDirectory", System.IO.Path.Combi  
ne(dirDataProject, "Data"));  
}
```

```
public class Startup  
{  
    0 references | 0 exceptions  
    public Startup(IConfiguration configuration, IHostingEnvironment env)  
    {  
        Configuration = configuration;  
  
        // Create data directory setting which points to the Data folder in the YOUR_APP.Data\Data.  
        string dirDataProject = env.ContentRootPath.Replace(".Web", ".Data", StringComparison.CurrentCultureIgnoreCase);  
        AppDomain.CurrentDomain.SetData("DataDirectory", System.IO.Path.Combine(dirDataProject, "Data"));  
    }  
}
```

Line 1: Current folder is GoMovies2019.Web
Replace Web with Data = GoMovies2019.Data

Line 2: Append Data to the path to get data folder:
GoMovies2019.Data\Data

Updating Startup

Locate the `ConfigureServices` method and copy and paste the highlighted code (lines 11-16) as shown below

```
// This method gets called by the runtime. Use this method to add service
s to the container.
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<CookiePolicyOptions>(options =>
    {
        // This lambda determines whether user consent for non-
essential cookies is needed for a given request.
        options.CheckConsentNeeded = context => true;
        options.MinimumSameSitePolicy = SameSiteMode.None;
    });

    // Replace the data directory in the connection string
    string defaultConnStr = Configuration.GetConnectionString("DefaultCon
nection");
    if (defaultConnStr.Contains("|DataDirectory|"))
    {
        defaultConnStr = defaultConnStr.Replace("|DataDirectory|", AppDom
ain.CurrentDomain.GetData("DataDirectory").ToString());
    }

    // ... Other code not shown
}
```

Updating Startup

Still in the ConfigureServices method, locate the line that says
services.AddDbContext<ApplicationDbContext>(options =>

Replace with the following highlighted code (see before and after screenshots):

BEFORE

```
services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(
        Configuration.GetConnectionString("DefaultConnection")));
services.AddDefaultIdentity<IdentityUser>()
    .AddDefaultUI(UIFramework.Bootstrap4)
    .AddEntityFrameworkStores<ApplicationDbContext>();
```

AFTER

```
services.AddDbContext<GoMovies2019Context>(options =>
    options.UseSqlServer(defaultConnStr));
services.AddIdentity<ApplicationUser, IdentityRole>()
    .AddDefaultUI(UIFramework.Bootstrap4)
    .AddEntityFrameworkStores<GoMovies2019Context>();
```

No quotes as it is using the variable created earlier

Your user class

Your database context class in the EF folder

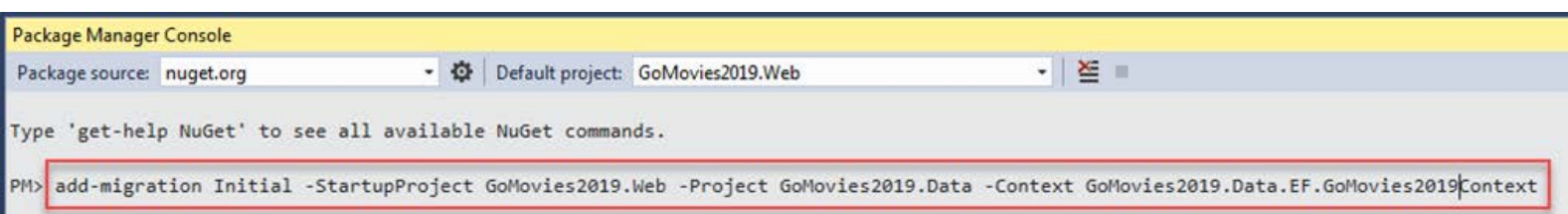
Generating the Database

Finally, you can generate the database!!

You use **migrations** to generate the database from your code and each migration is given a name. The first one is usually called **Initial**.

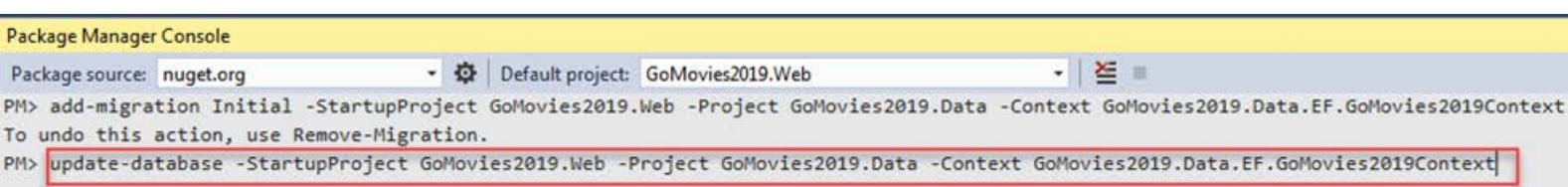
- ♦ Go to Tools ? NuGet Package Manager ? Package Manager Console
- ♦ Copy and paste in the following code at the command prompt replacing **YOUR_APP_NAME** as before (see example screenshot). Remember C# is case-sensitive.
- ♦ Press enter key to run the command

```
add-migration Initial -StartupProject YOUR_APP_NAME.Web -Project YOUR_APP_NAME.Data  
-Context YOUR_APP_NAME.Data.EF.YOUR_APP_NAMEContext
```



- ♦ You should get the message: To undo this action, use Remove-Migration.
- ♦ Now, run the following command to send the changes to the database replacing **YOUR_APP_NAME** as before (see example screenshot). Remember C# is case-sensitive.
- ♦ Press enter key to run the command

```
update-database -StartupProject YOUR_APP_NAME.Web -Project YOUR_APP_NAME.Data  
-Context YOUR_APP_NAME.Data.EF.YOUR_APP_NAMEContext
```



- ♦ Hopefully you get the message Applying Migration... Done
- ♦ Check to see if your database tables have been generated.

- ▲  (localdb)\ProjectsV13 (SQL Server 13.0.4001 - UNI\sc-ka)
 - ▲  Databases
 - ▷  System Databases
 - ▷  GoMovies2018Dec
 - ▲  GoMovies2019
 - ▲  Tables
 - ▷  System Tables
 - ▷  External Tables
 - ▷  dbo.__EFMigrationsHistory
 - ▷  dbo.AspNetRoleClaims
 - ▷  dbo.AspNetRoles
 - ▷  dbo.AspNetUserClaims
 - ▷  dbo.AspNetUserLogins
 - ▷  dbo.AspNetUserRoles
 - ▷  dbo.AspNetUsers
 - ▷  dbo.AspNetUserTokens
 - ▷  dbo.Booking
 - ▷  dbo.BookingTicket
 - ▷  dbo.Film
 - ▷  dbo.FilmShowing
 - ▷  dbo.TicketType