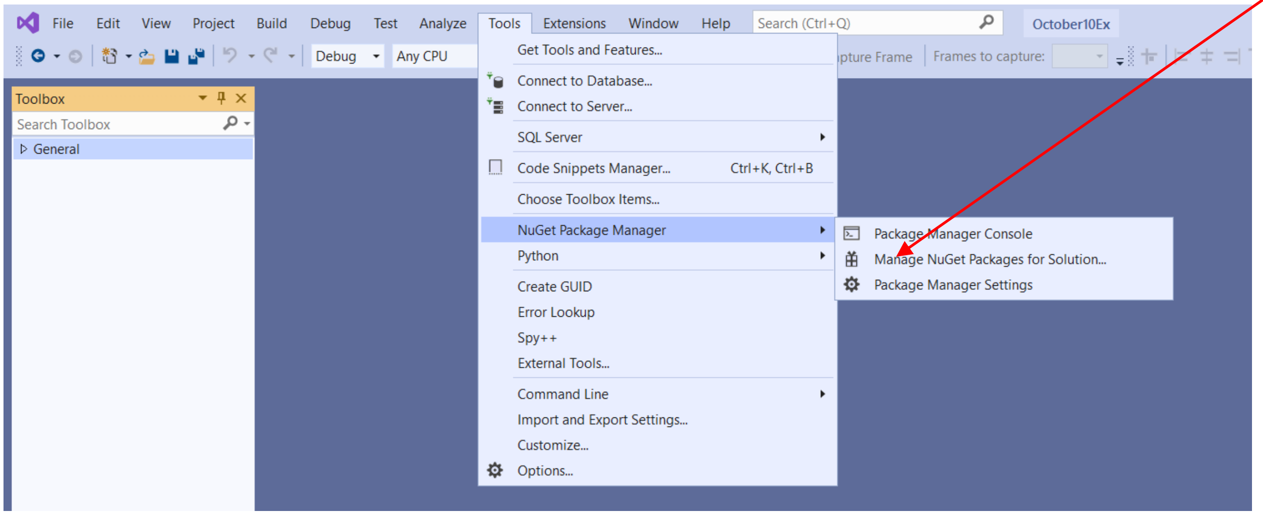
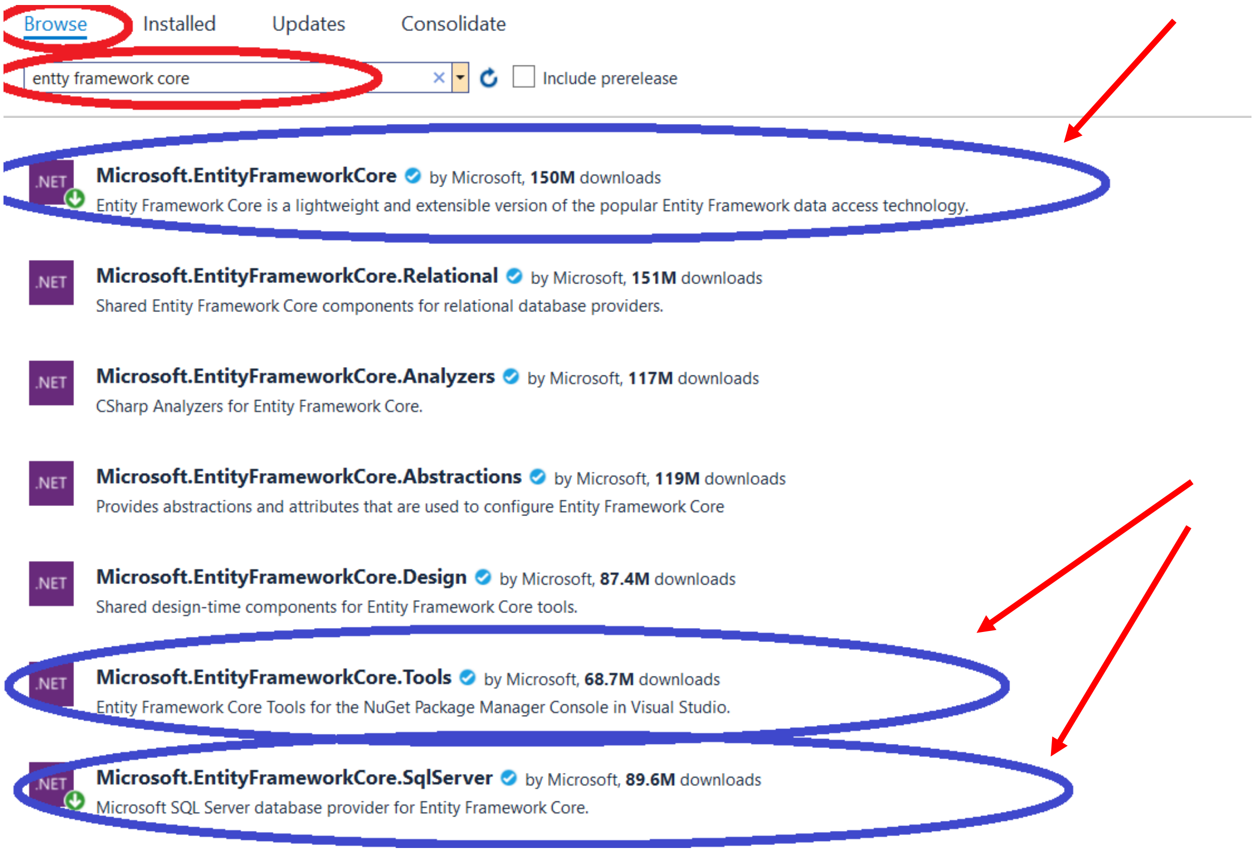
Creating CRUD pages by Reverse Engineering from SQl Server tables

(A) First create your CORE Web application from the template.

Install the following three packages from the NuGet package Manager. See image below on how to open the NuGet package manager.



When you select the NuGet package maager, be sure to click on the Browse tab. If you enter “entitu framework core” in the search bar, the three packages you are to install pop up. See the three packages encircled with blue ovals below



entty framework core

Microsoft.EntityFrameworkCore by Microsoft, **150M** downloads
Entity Framework Core is a lightweight and extensible version of the popular Entity Framework data access technology.

Microsoft.EntityFrameworkCore.Relational by Microsoft, **151M** downloads
Shared Entity Framework Core components for relational database providers.

Microsoft.EntityFrameworkCore.Analyzers by Microsoft, **117M** downloads
CSharp Analyzers for Entity Framework Core.

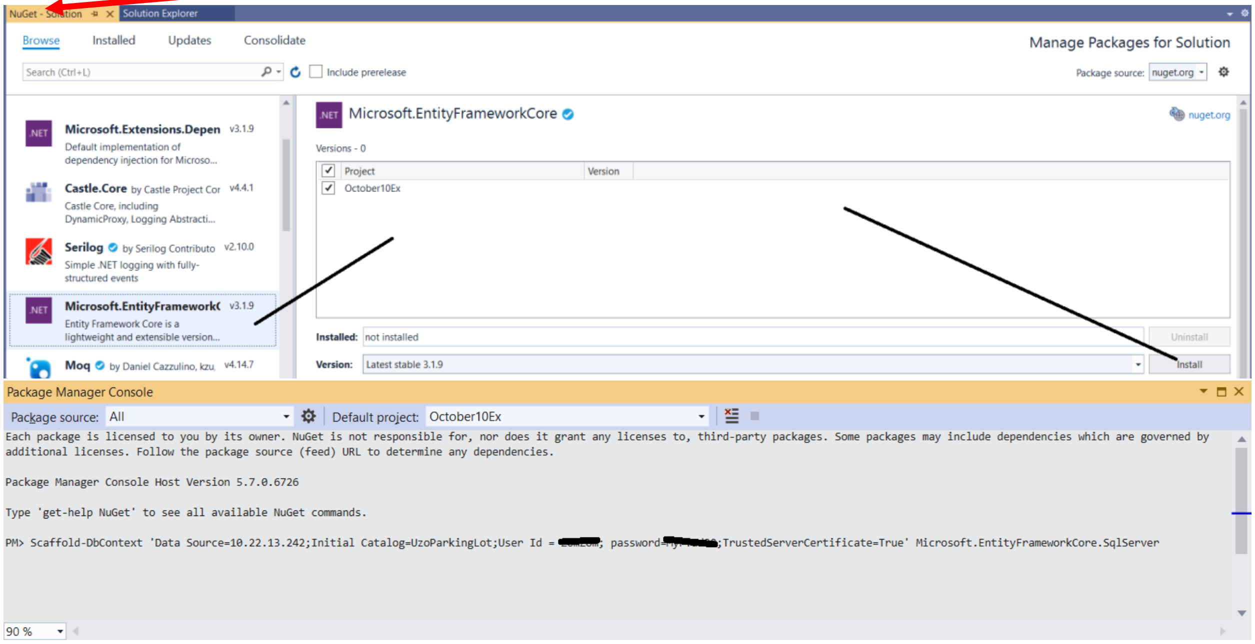
Microsoft.EntityFrameworkCore.Abstractions by Microsoft, **119M** downloads
Provides abstractions and attributes that are used to configure Entity Framework Core

Microsoft.EntityFrameworkCore.Design by Microsoft, **87.4M** downloads
Shared design-time components for Entity Framework Core tools.

Microsoft.EntityFrameworkCore.Tools by Microsoft, **68.7M** downloads
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

Microsoft.EntityFrameworkCore.SqlServer by Microsoft, **89.6M** downloads
Microsoft SQL Server database provider for Entity Framework Core.

One by one, select each package and install. Click on the check boxes as seen below, and click on the install button seen on the bottom right of the screen. To see properly, click on Nuget Tab and select the FLOAT option and stretch the window.


NuGet - Solution
Solution Explorer
Browse
Installed
Updates
Consolidate
Manage Packages for Solution
Search (Ctrl+L)
Include prerelease
Package source: nuget.org
Microsoft.Extensions.Depen v3.1.9
Default implementation of dependency injection for Microso...
Castle.Core by Castle Project Cor v4.4.1
Castle Core, including DynamicProxy, Logging Abstracti...
Serilog by Serilog Contributo v2.10.0
Simple .NET logging with fully-structured events
Microsoft.EntityFramework(v3.1.9
Entity Framework Core is a lightweight and extensible version...
Moq by Daniel Cazzulino, kzu, v4.14.7
Microsoft.EntityFrameworkCore
nuget.org
Versions - 0
Project
Version
Uninstall
Install
Package Manager Console
90 %

☑ Project
☑ October10Ex

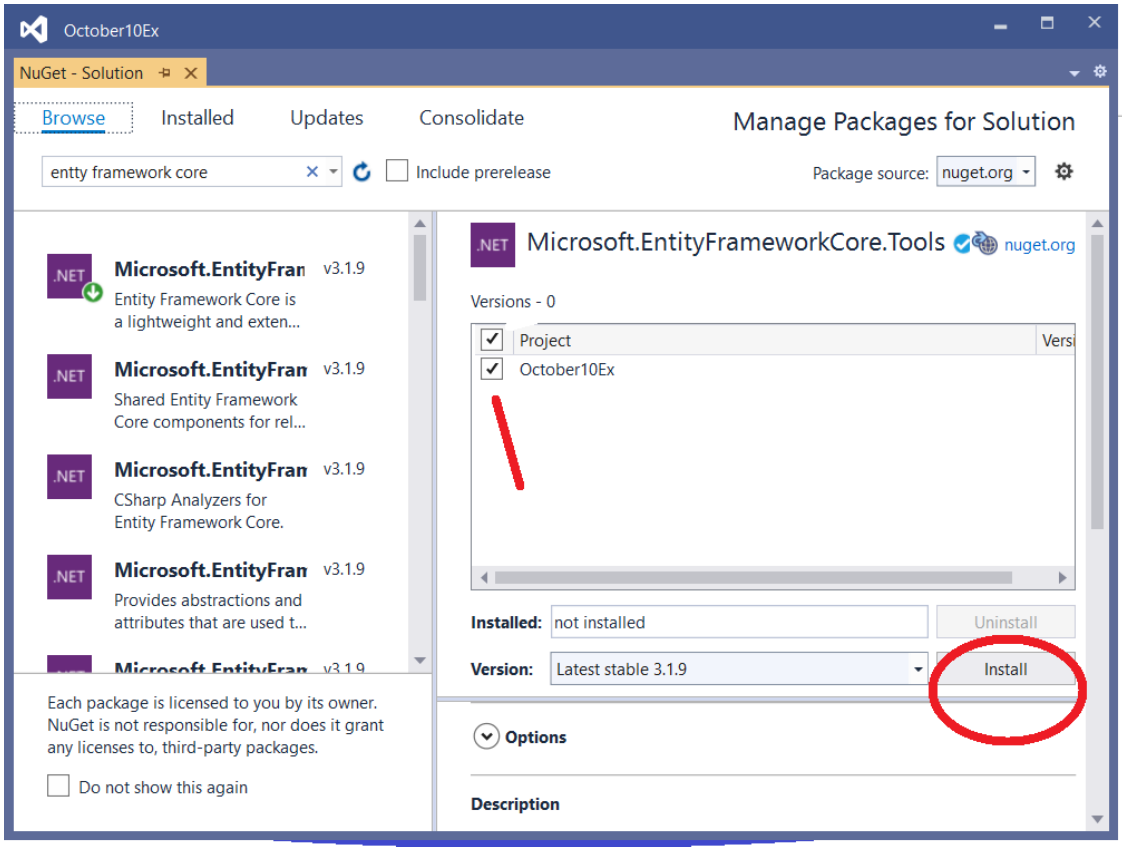
Installed: not installed

Version: Latest stable 3.1.9

Package source: All

Default project: October10Ex

```
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages. Some packages may include dependencies which are governed by
additional licenses. Follow the package source (feed) URL to determine any dependencies.

Package Manager Console Host Version 5.7.0.6726

Type 'get-help NuGet' to see all available NuGet commands.

PM> Scaffold-DbContext 'Data Source=10.22.13.242;Initial Catalog=UzoParkingLot;User Id = [redacted]; password=[redacted];TrustedServerCertificate=True' Microsoft.EntityFrameworkCore.SqlServer
```


October10Ex
NuGet - Solution
Browse
Installed
Updates
Consolidate
Manage Packages for Solution
Include prerelease
Microsoft.EntityFran v3.1.9
Entity Framework Core is a lightweight and exten...
Microsoft.EntityFran v3.1.9
Shared Entity Framework Core components for rel...
Microsoft.EntityFran v3.1.9
CSharp Analyzers for Entity Framework Core.
Microsoft.EntityFran v3.1.9
Provides abstractions and attributes that are used t...
Microsoft.EntityFran v3.1.9
Microsoft.EntityFrameworkCore.Tools
nuget.org
Versions - 0
Project
Versi
Uninstall
Install
Options
Description

entty framework core

Package source: nuget.org

☑ Project
☑ October10Ex

Installed: not installed

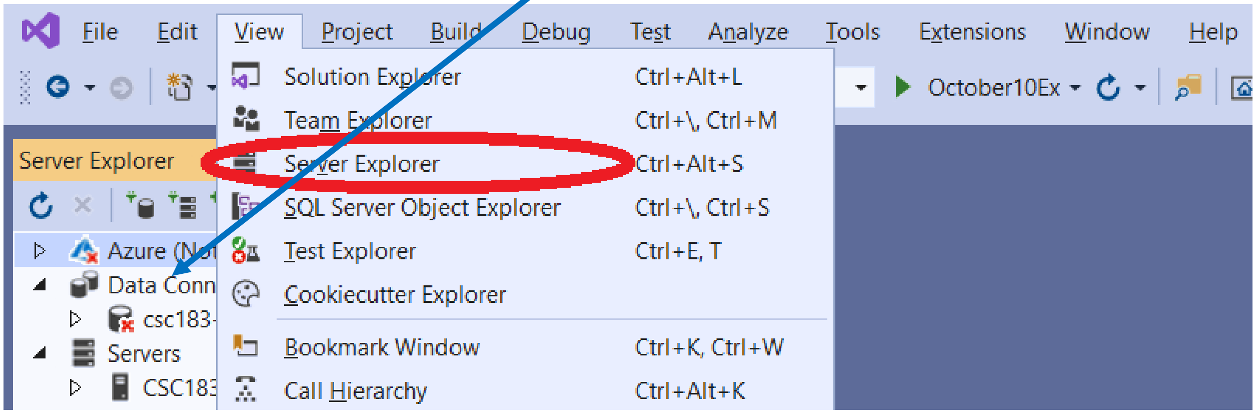
Version: Latest stable 3.1.9

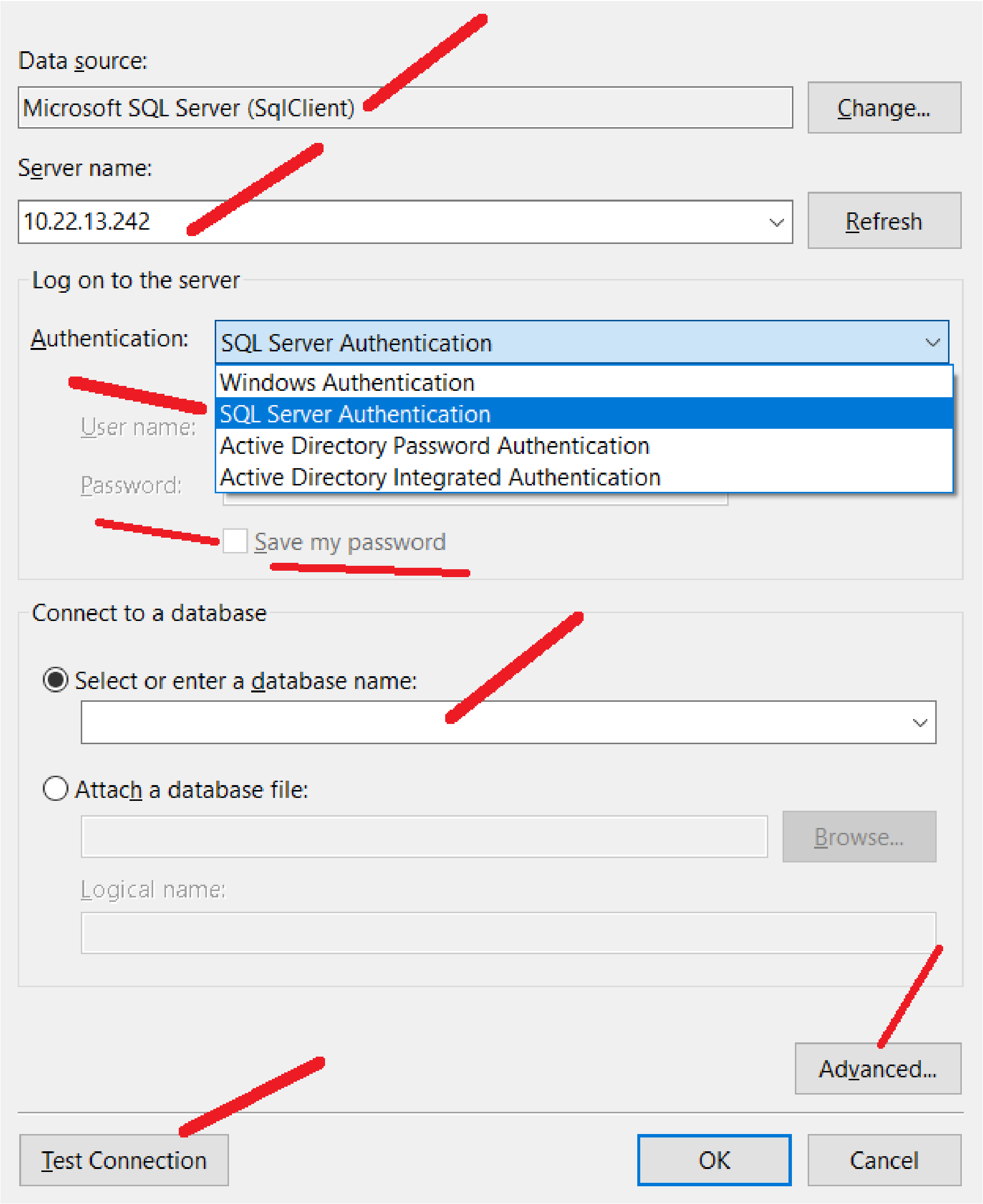
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

□ Do not show this again

(B) **Set up the database connection**

Select the Server Explorer, that will reveal the “Data Connection”. Right click on Data Connection, and click “Add Connection”


File
Edit
View
Project
Build
Debug
Test
Analyze
Tools
Extensions
Window
Help
Solution Explorer Ctrl+Alt+L
Team Explorer Ctrl+\, Ctrl+M
Server Explorer Ctrl+Alt+S
SQL Server Object Explorer Ctrl+\, Ctrl+S
Test Explorer Ctrl+E, T
Cookiecutter Explorer
Bookmark Window Ctrl+K, Ctrl+W
Call Hierarchy Ctrl+Alt+K
October10Ex
Server Explorer
Azure (Not
Data Conn
csc183
Servers
CSC183


Data source: Microsoft SQL Server (SqlClient)

Server name: 10.22.13.242

Log on to the server

Authentication: SQL Server Authentication

□ Save my password

Connect to a database

● Select or enter a database name:

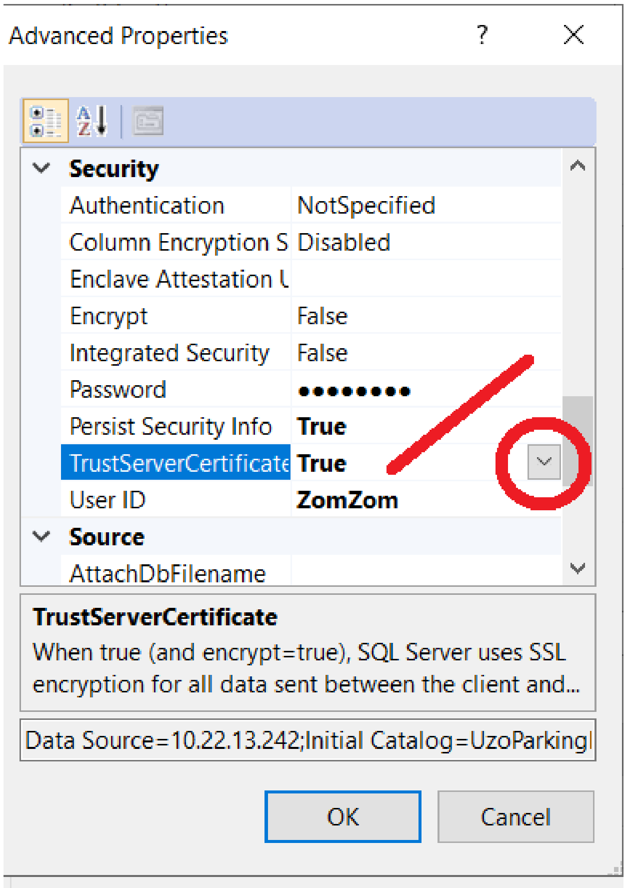
○ Attach a database file:

- Select Microsoft SQL Server SQL Client
- Enter Server IP address of 10.22.13.242
- Select SQL Server Authentication
- CLICK SAVE MY PASSWORD
- Click Select database name
- Enter the database name you created in SQLSERVER

To make sure you are on the right track, click on Test Connection.

If it succeeded then

- Click ADVANCED and see the image on the next page


Advanced Properties

Security

Authentication	NotSpecified
Column Encryption S	Disabled
Enclave Attestation U	
Encrypt	False
Integrated Security	False
Password	●●●●●●●●
Persist Security Info	True
TrustServerCertificate	True
User ID	ZomZom

Source

AttachDbFilename	

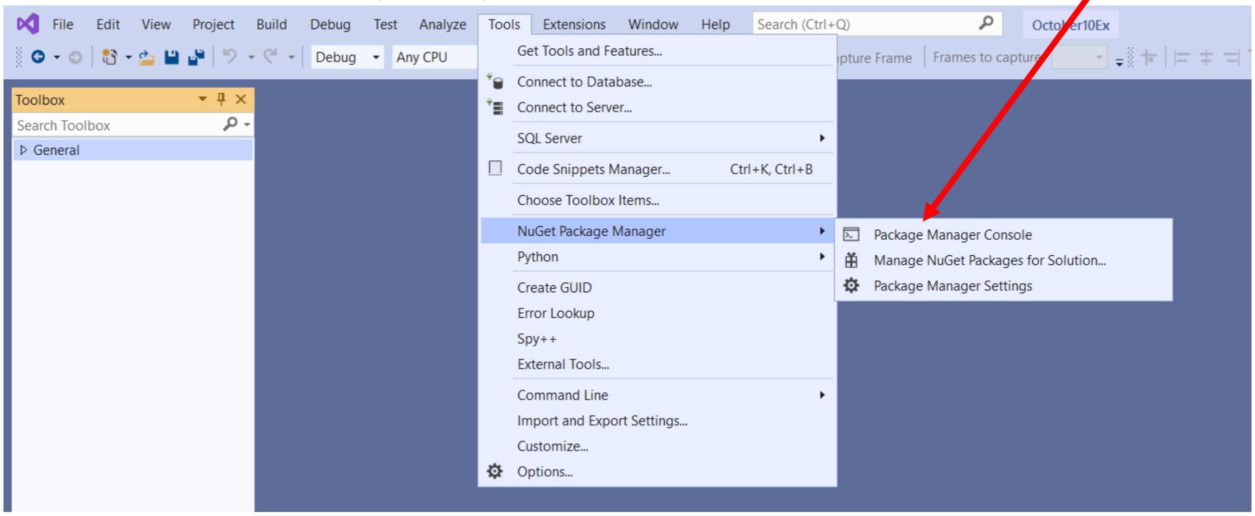
TrustServerCertificate
When true (and encrypt=true), SQL Server uses SSL encryption for all data sent between the client and...

Data Source=10.22.13.242;Initial Catalog=UzoParkingl

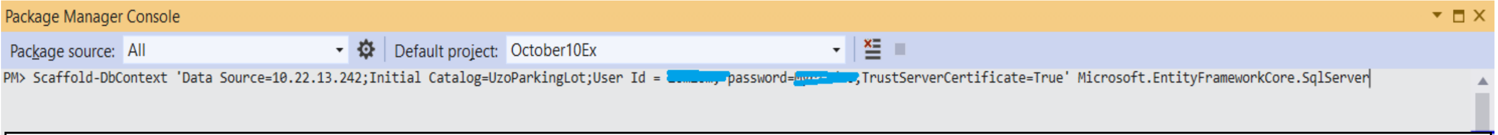
- TrustServerCertificate, be sure to Select TRUE
- Select OK

(C) Scaffold the Database

Select Tools, then NuGet Package Manager, then PACKAGE MANAGER CONSOLE



In the package manager console window, shown below, do the following:

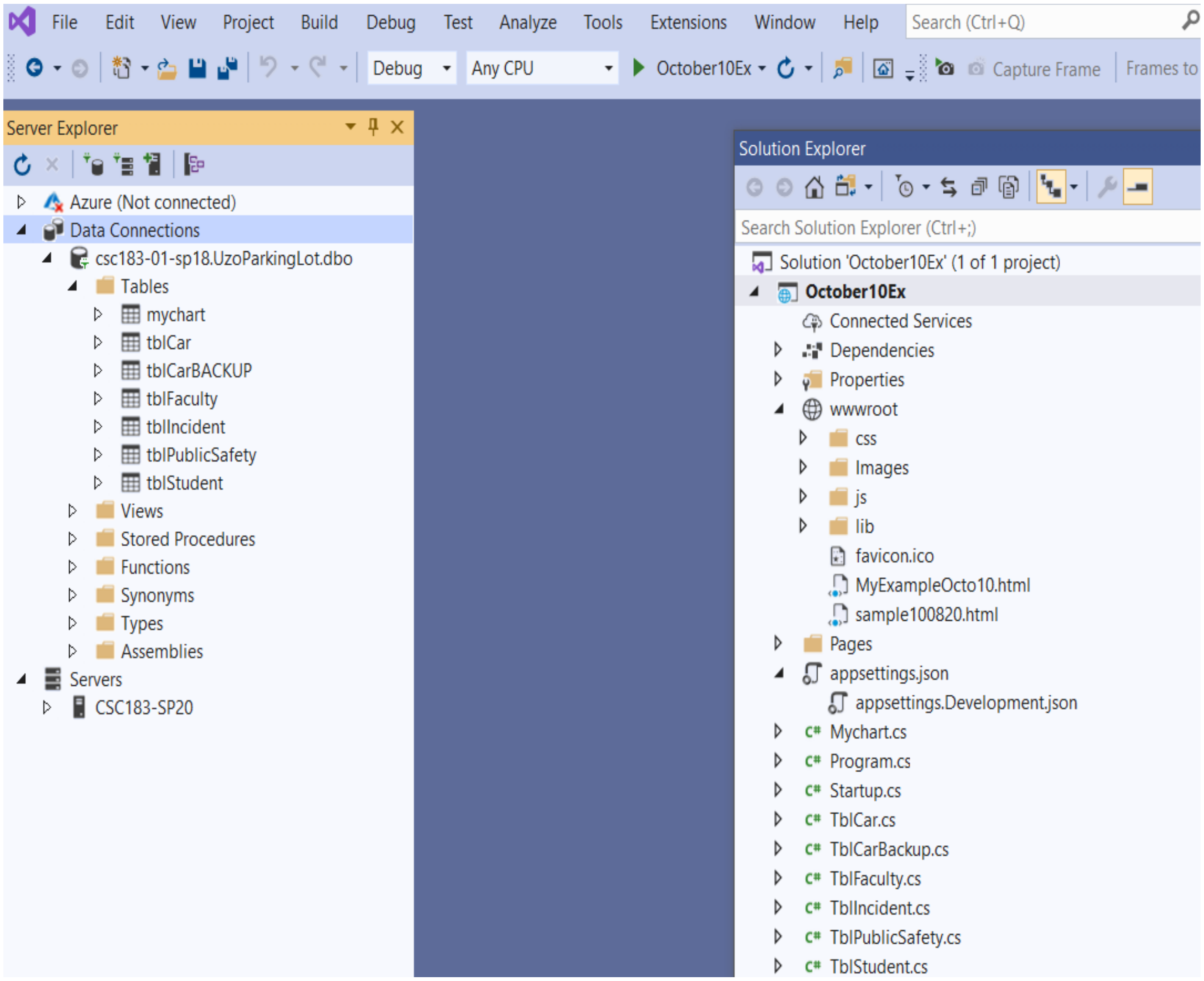


```
PM> Scaffold-DbContext 'Data Source=10.22.13.242;Initial Catalog=UzoParkingLot;User Id = [redacted]password=[redacted],TrustServerCertificate=True' Microsoft.EntityFrameworkCore.SqlServer
```

Paste the following command in the console window, put **your database name** (the one you made in SQL SERVER), your **user id**, and **password** in the shaded areas below. Do exactly as shown:

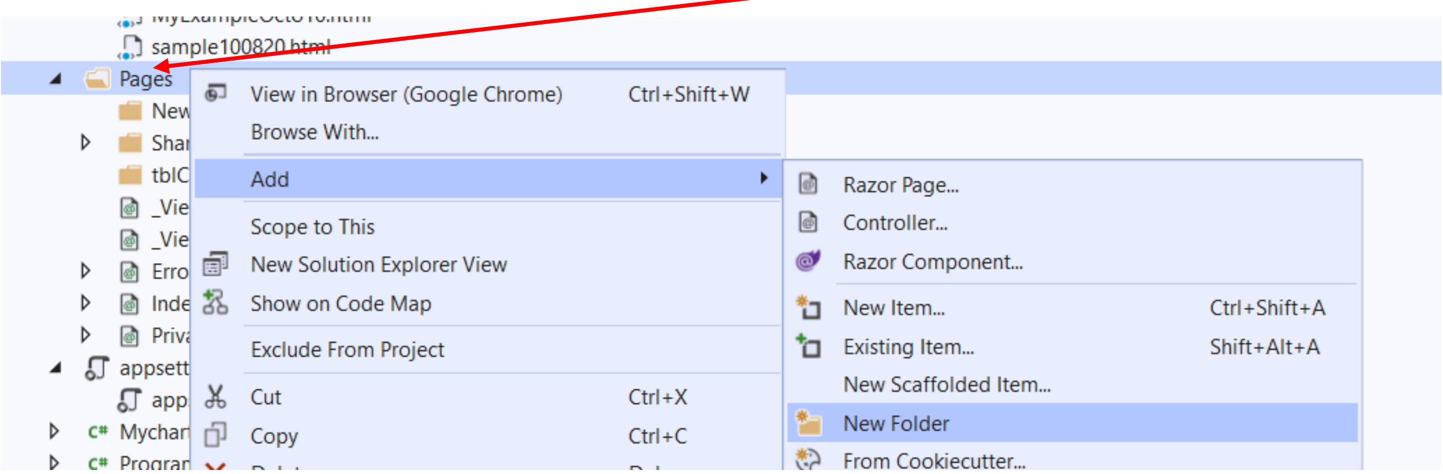
Scaffold-DbContext 'Data Source=10.22.13.242; Initial Catalog= yourDataBase ; User Id = yourUserId ; password= yourPwd ; TrustServerCertificate=True' Microsoft.EntityFrameworkCore.SqlServer

After running the above command,   A C# (.cs) page will be made for each table in your specified database. Your screen should look like the image below:

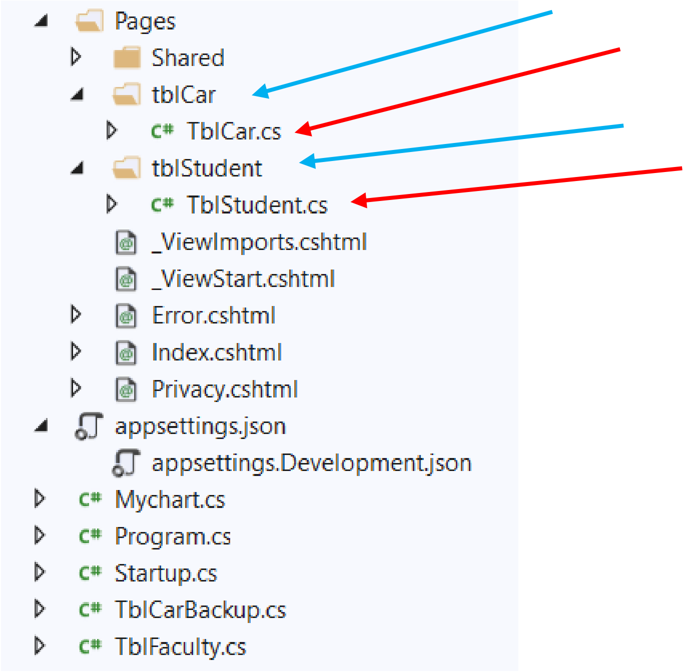


(D) Reengineering the Database and Make the CRUD Pages

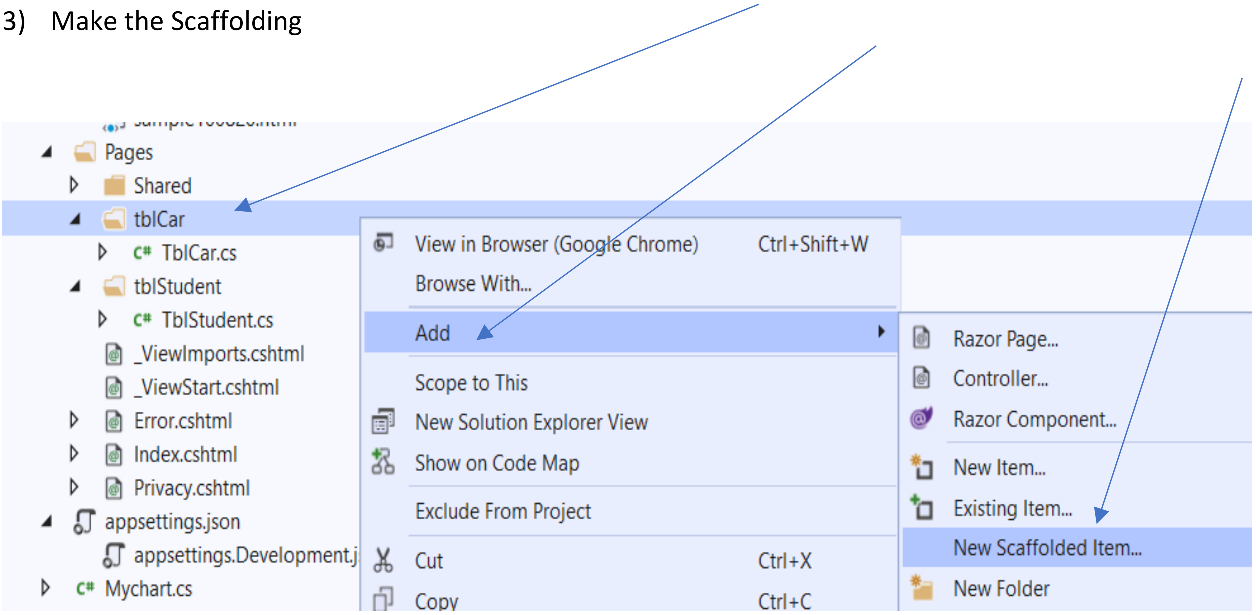
1) Start by creating a folder for each table you would like to scaffold. Be sure to select Pages before right clicking and Adding new folder.



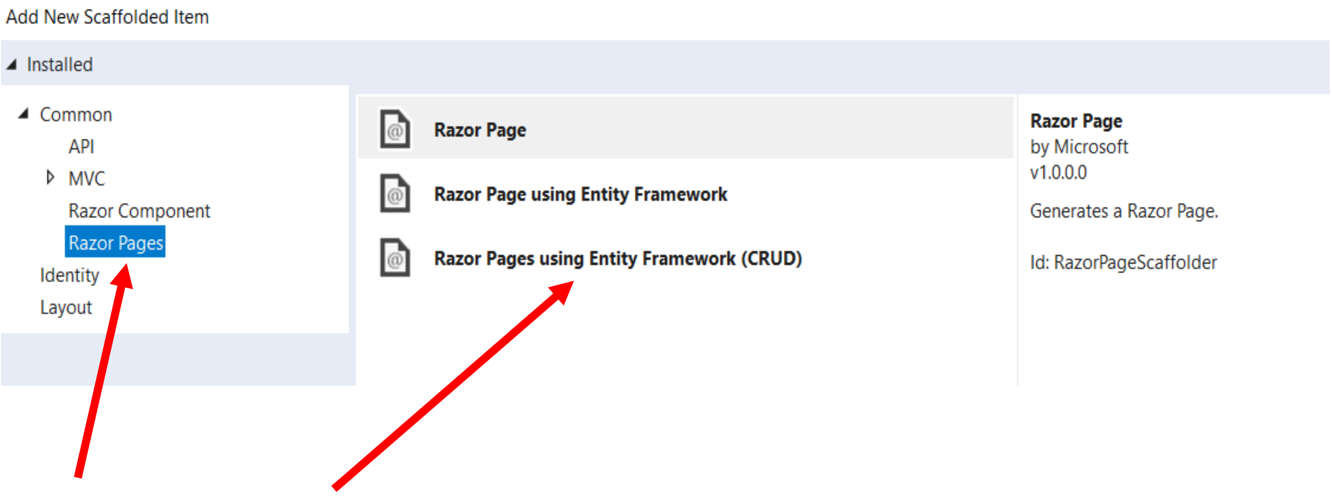
2) Move the controller file for that table into the appropriate folder. Just drag the respective .cs file into the folder.



3) Make the Scaffolding



4) Create the Scaffolded CRUD pages



Add New Scaffolded Item

◢ Installed

◢ Common
- API
- ▷ MVC
- Razor Component
- Razor Pages
- Identity
- Layout

Razor Page

Razor Page using Entity Framework

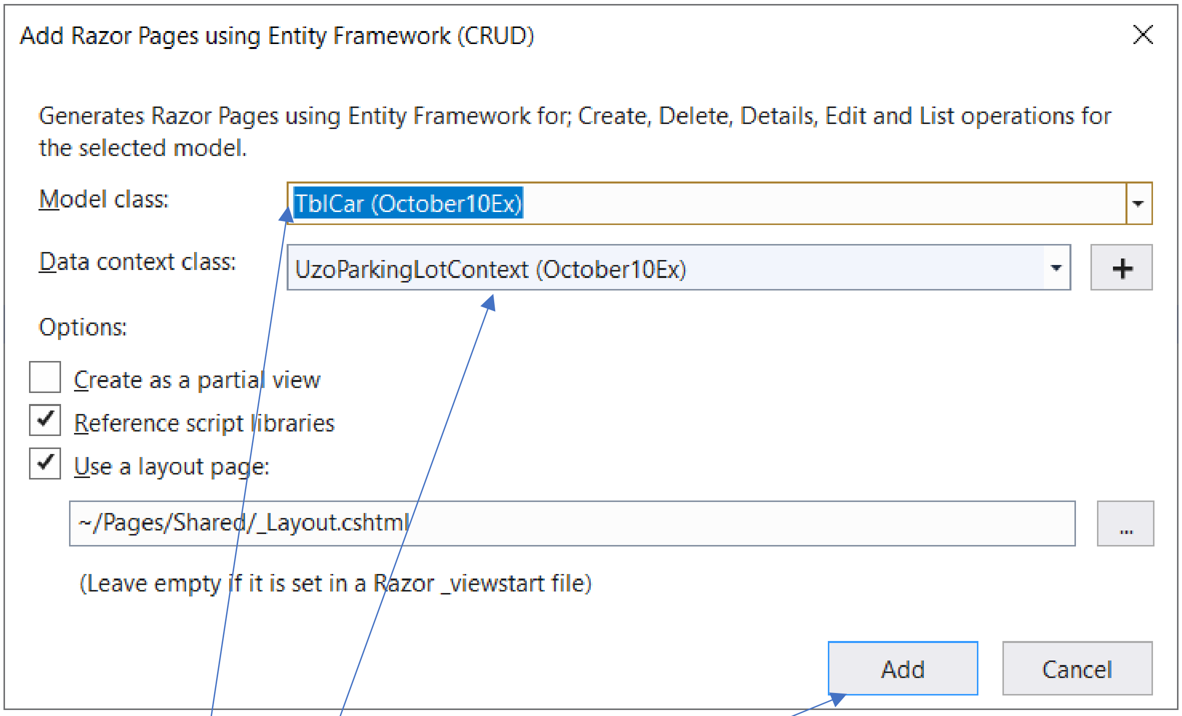
Razor Pages using Entity Framework (CRUD)

Razor Page
by Microsoft
v1.0.0.0

Generates a Razor Page.

Id: RazorPageScaffolder

5) After selecting Razor Pages using Entity Framework CRUD, click the ADD button at the bottom right of the screen.



Add Razor Pages using Entity Framework (CRUD)

Generates Razor Pages using Entity Framework for; Create, Delete, Details, Edit and List operations for the selected model.

Model class: TblCar (October10Ex)

Data context class: UzoParkingLotContext (October10Ex) +

Options:

□ Create as a partial view

☑ Reference script libraries

☑ Use a layout page:

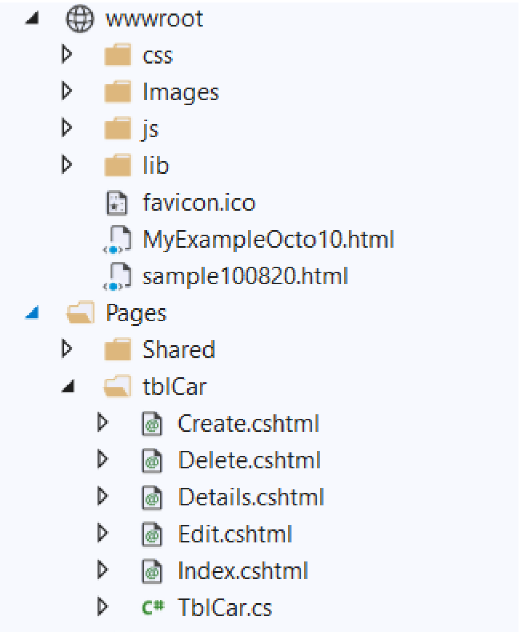
~/Pages/Shared/_Layout.cshtml ...

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

6) Select which entity (model class) you wish to create CRUD pages on. Your database context which should already be selected will be listed. Then click ADD

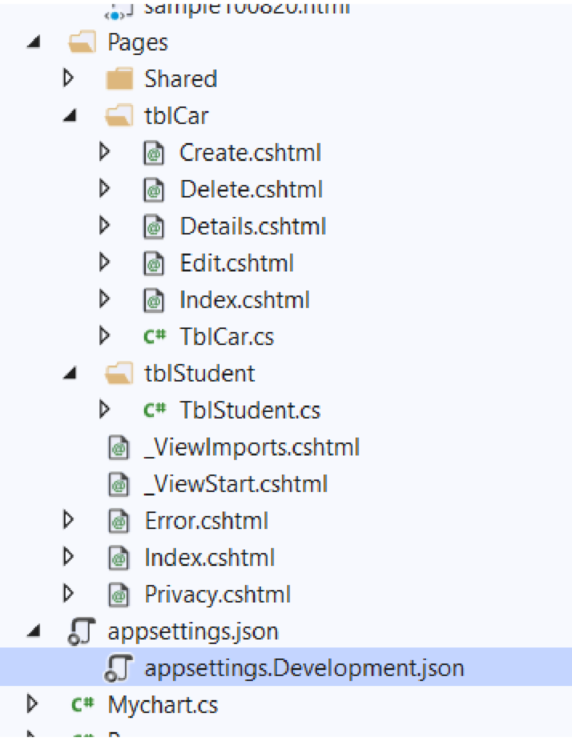
Your screen will look like the following:



```
◢ wwwroot
   ▷ css
   ▷ Images
   ▷ js
   ▷ lib
     favicon.ico
     MyExampleOcto10.html
     sample100820.html
◢ Pages
   ▷ Shared
   ◢ tblCar
      ▷ Create.cshtml
      ▷ Delete.cshtml
      ▷ Details.cshtml
      ▷ Edit.cshtml
      ▷ Index.cshtml
      ▷ C# TblCar.cs
```

E) Establish application settings in the appsettings.json file.

1) In your **appsettings.development.json** file seen below



```
     sample100820.html
◢ Pages
   ▷ Shared
   ◢ tblCar
      ▷ Create.cshtml
      ▷ Delete.cshtml
      ▷ Details.cshtml
      ▷ Edit.cshtml
      ▷ Index.cshtml
      ▷ C# TblCar.cs
   ◢ tblStudent
      ▷ C# TblStudent.cs
     _ViewImports.cshtml
     _ViewStart.cshtml
   ▷ Error.cshtml
   ▷ Index.cshtml
   ▷ Privacy.cshtml
◢ appsettings.json
     appsettings.Development.json
▷ C# Mychart.cs
```

Enter the following as a new json object in the file, replacing your database name, user id, and password as indicated by your sqlserver credentials:

For example "parkingDBConnection"
Include the quotes

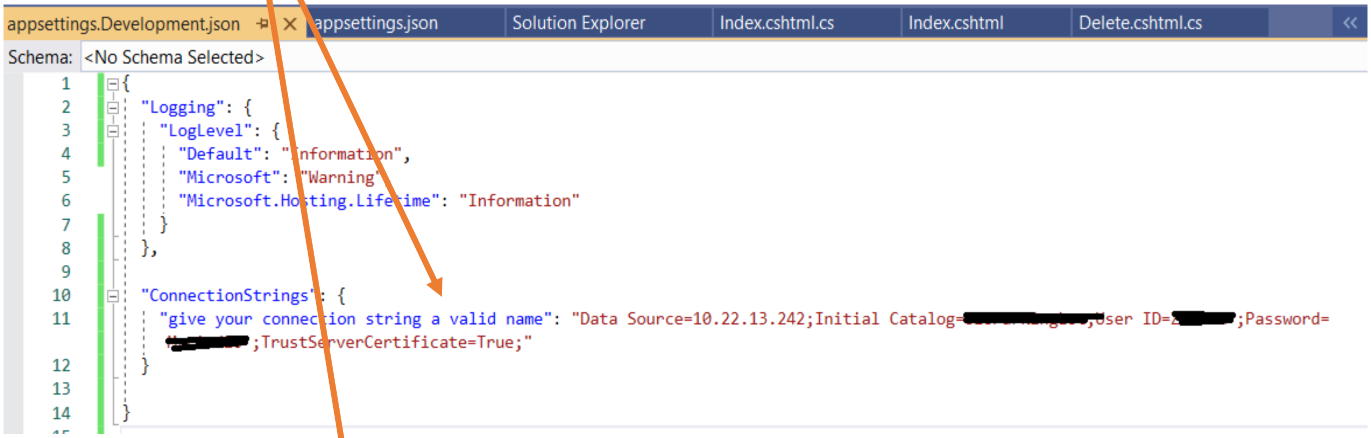
"ConnectionStrings": {

"give your connection string a valid name": "Data Source=10.22.13.242; Initial Catalog =yourDatabase ; User ID = yourUSerId ; Password = yourPassword ; TrustServerCertificate = True ;"

}

Be sure to replace the Initial Catalog value with your database name, the user id and password with your sqlserver credentials. Also replace "give your connection string a name" with the name of your choice e.g. "parkingDBConnection". Keep the SAME connection string name for any prompt requiring a connection to your database from this web application.

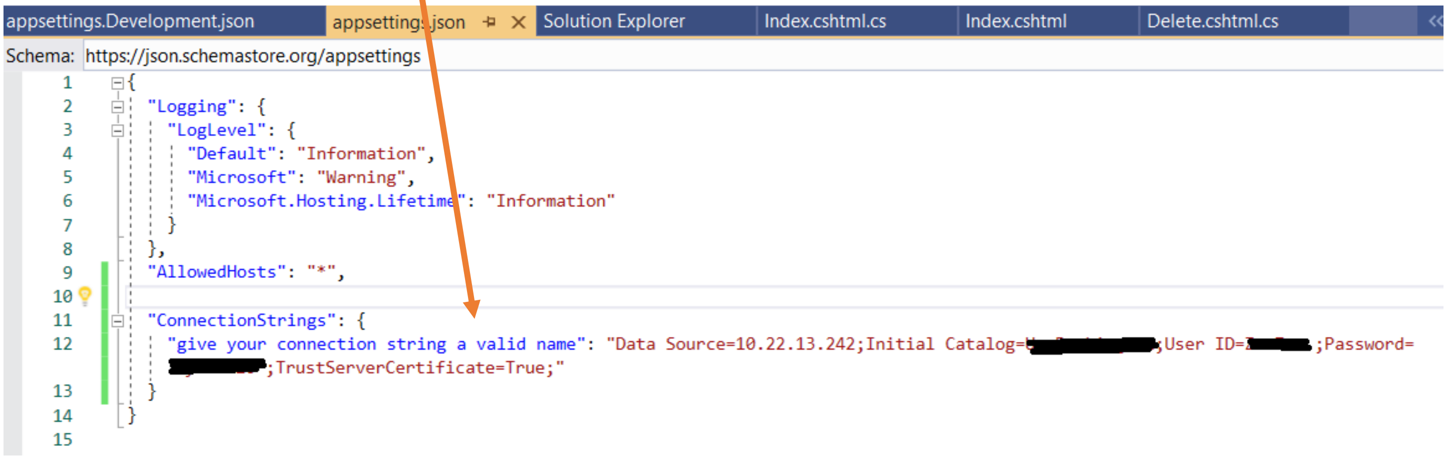
Note the curly braces used and the comma separating the previous object from the connection string object



Schema: <No Schema Selected>

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },

  "ConnectionStrings": {
    "give your connection string a valid name": "Data Source=10.22.13.242;Initial Catalog=[redacted]ser ID=[redacted];Password=
    [redacted];TrustServerCertificate=True;"
  }

}
```

2) **Do the same thing for the appsettings.json file, slightly different from the development .json file. Note the comma after allowed hosts, and the balancing of the curly braces.**

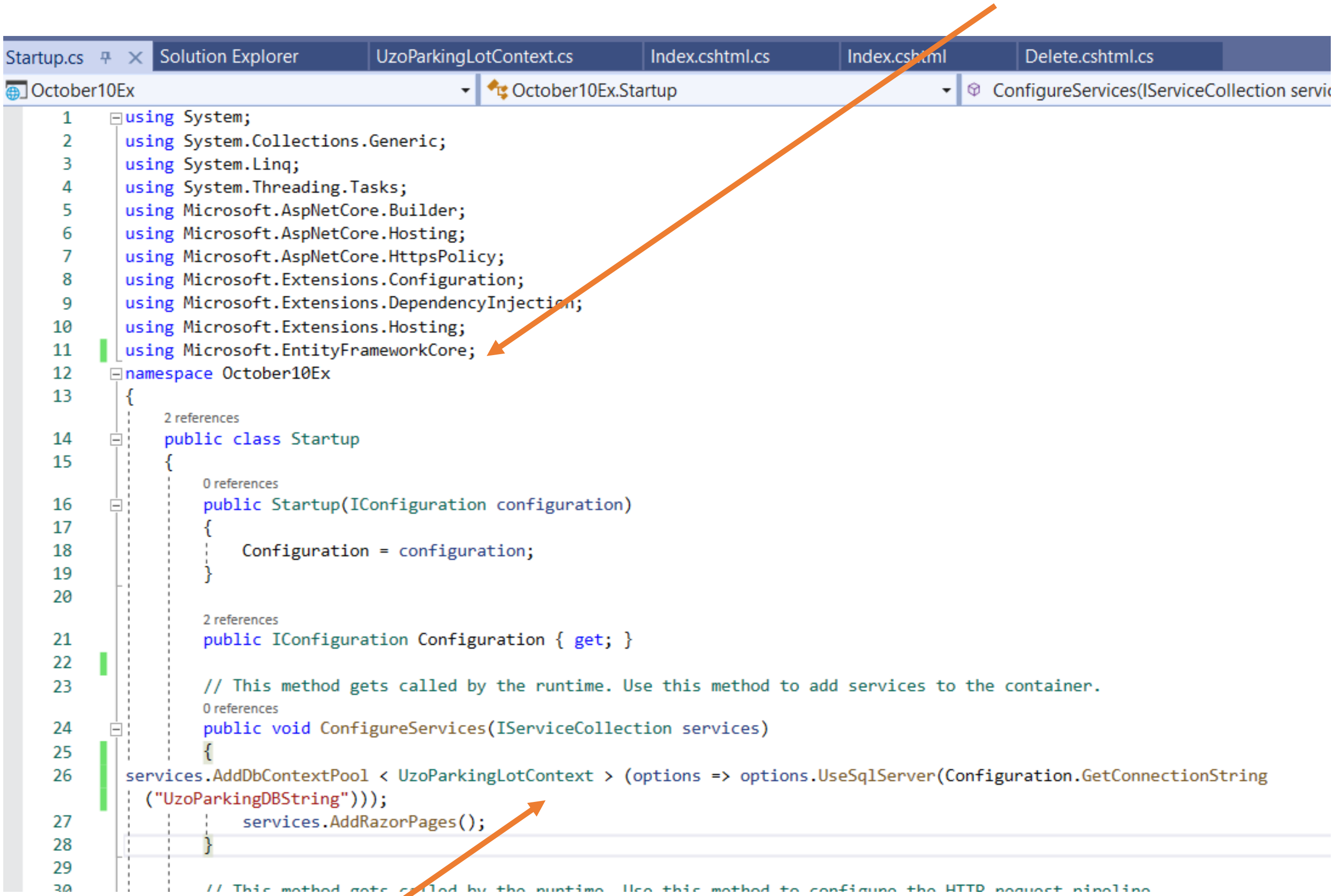


Schema: https://json.schemastore.org/appsettings

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",

  "ConnectionStrings": {
    "give your connection string a valid name": "Data Source=10.22.13.242;Initial Catalog=[redacted];User ID=[redacted];Password=
    [redacted];TrustServerCertificate=True;"
  }
}
```

3) Adjust **Startup.cs** file you will need the name of the context.cs page created in (C) and the name of your connection string created in (E1).   At the top of the page add the following:

Add this →>   **using Microsoft.EntityFrameworkCore;**



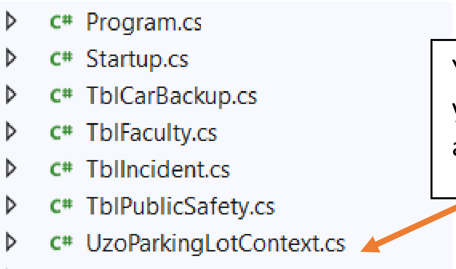
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.EntityFrameworkCore;
namespace October10Ex
{
    2 references
    public class Startup
    {
        0 references
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        2 references
        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        0 references
        public void ConfigureServices(IServiceCollection services)
        {
services.AddDbContextPool < UzoParkingLotContext > (options => options.UseSqlServer(Configuration.GetConnectionString
  ("UzoParkingDBString")));
            services.AddRazorPages();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
```

Replace the contents of the public void ConfigureServices function with your specified connection string, instead of leaving your user id and password exposed.

Put the following statement ***Before*** the services.AddRazorPAges() statement

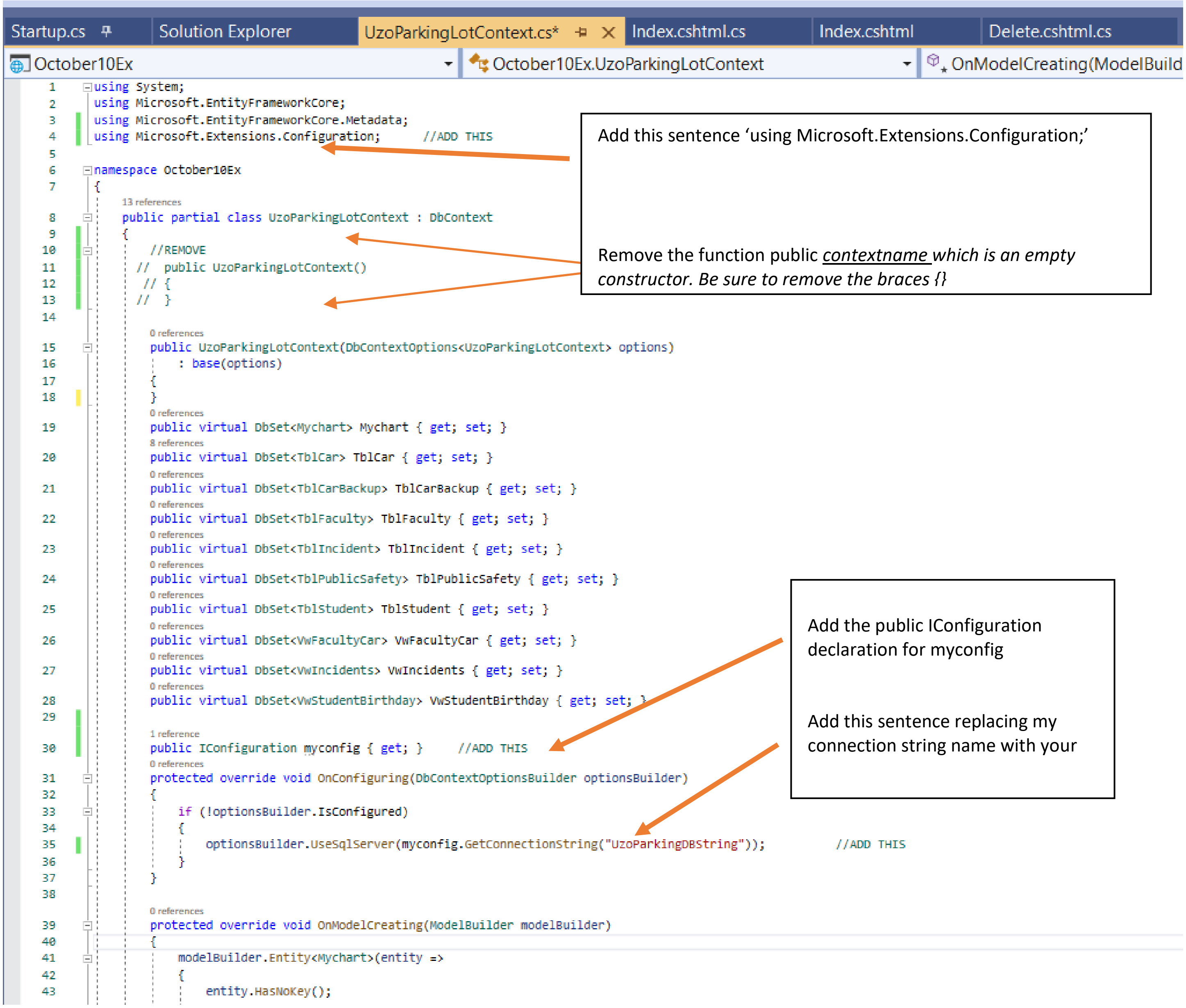
services.AddDbContextPool< *replace with the name of your context file* >(options => options.UseSqlServer(myconfig.GetConnectionString("*name of your connection string*")));

F) Modify contents of context file to utilize connection string as opposed to exposing user id and password



- Program.cs
- Startup.cs
- TblCarBackup.cs
- TblFaculty.cs
- TblIncident.cs
- TblPublicSafety.cs
- UzoParkingLotContext.cs

Your context file will have a name that has your databasename with the word Context and file extension of .cs



```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.Extensions.Configuration;      //ADD THIS

namespace October10Ex
{
    13 references
    public partial class UzoParkingLotContext : DbContext
    {
        //REMOVE
      //  public UzoParkingLotContext()
       // {
      //  }

        0 references
        public UzoParkingLotContext(DbContextOptions<UzoParkingLotContext> options)
            : base(options)
        {
        }
        0 references
        public virtual DbSet<Mychart> Mychart { get; set; }
        8 references
        public virtual DbSet<TblCar> TblCar { get; set; }
        0 references
        public virtual DbSet<TblCarBackup> TblCarBackup { get; set; }
        0 references
        public virtual DbSet<TblFaculty> TblFaculty { get; set; }
        0 references
        public virtual DbSet<TblIncident> TblIncident { get; set; }
        0 references
        public virtual DbSet<TblPublicSafety> TblPublicSafety { get; set; }
        0 references
        public virtual DbSet<TblStudent> TblStudent { get; set; }
        0 references
        public virtual DbSet<VwFacultyCar> VwFacultyCar { get; set; }
        0 references
        public virtual DbSet<VwIncidents> VwIncidents { get; set; }
        0 references
        public virtual DbSet<VwStudentBirthday> VwStudentBirthday { get; set; }

        1 reference
        public IConfiguration myconfig { get; }     //ADD THIS
        0 references
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
                optionsBuilder.UseSqlServer(myconfig.GetConnectionString("UzoParkingDBString"));         //ADD THIS
            }
        }

        0 references
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Mychart>(entity =>
            {
                entity.HasNoKey();
```

Add this sentence ‘using Microsoft.Extensions.Configuration;’

Remove the function public <u>*contextname*</u> *which is an empty constructor. Be sure to remove the braces {}*

Add the public IConfiguration declaration for myconfig

Add this sentence replacing my connection string name with your

Save everything by clicking on double blue diskette icon on menu bar.

Test your CRUD pages by right-clicking on the index.cshtml of the table you scaffolded.and select View in Browser