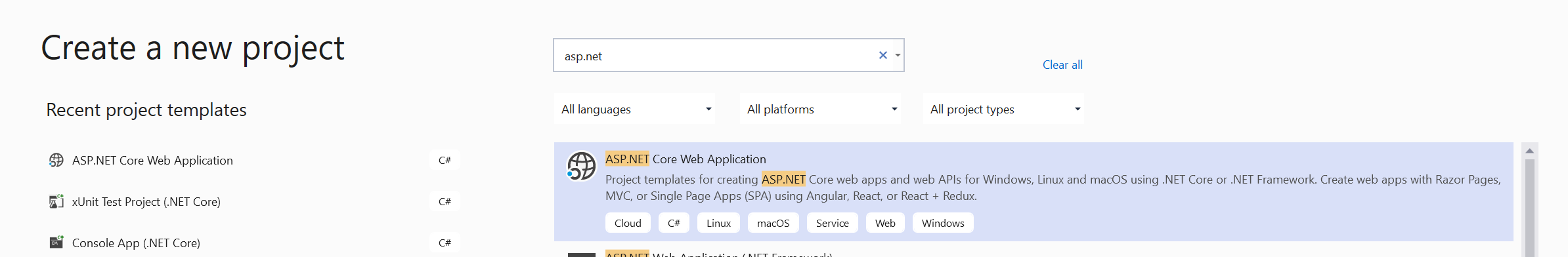
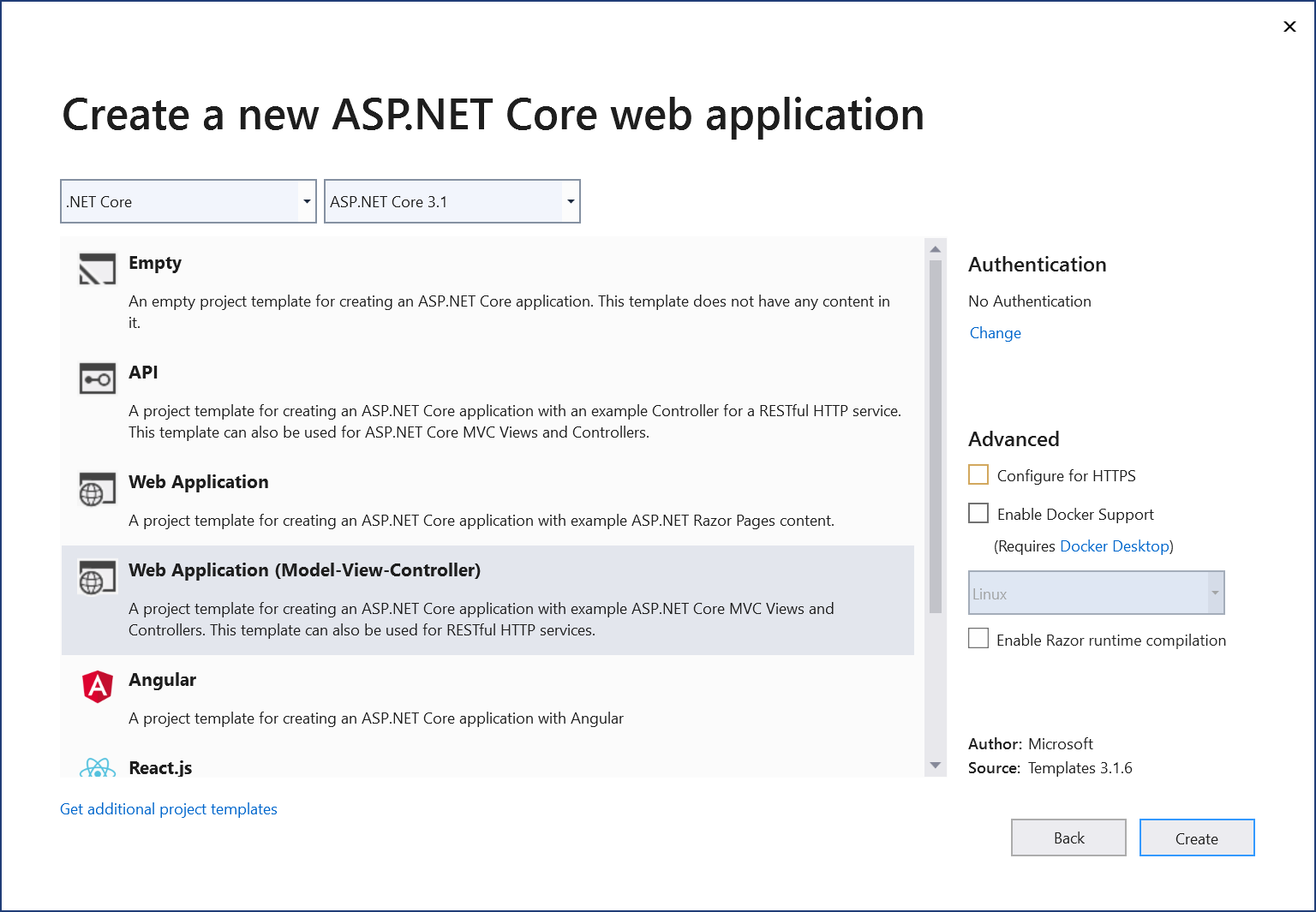
Model-First Web Application in ASP.NET Core MVC

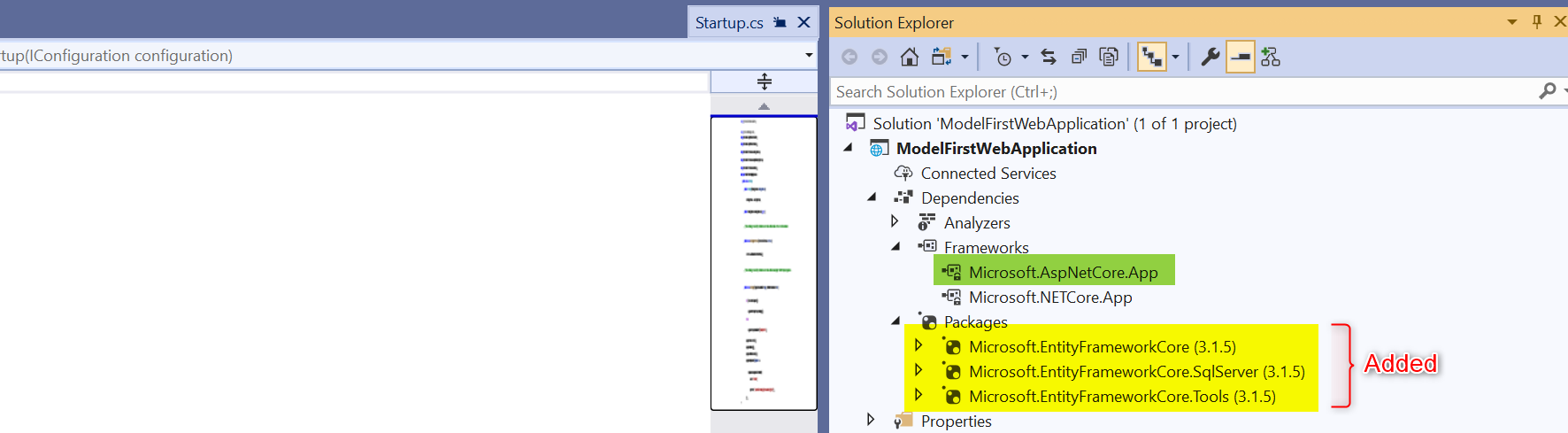
using EntityFrameworkCore



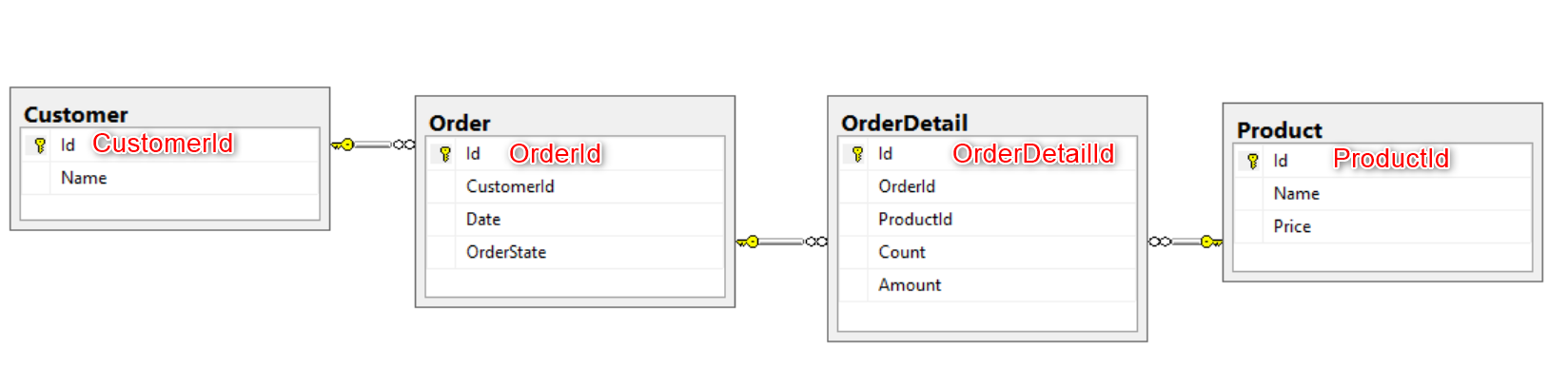


(1) Add EntityFrameworkCore to application (event hough it exists in the following).

It's still best to get the latest version.



(2) We'll base our database on the following database model (n.b. columns nenamed for better understanding)



**Customer.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Threading.Tasks;

namespace ModelFirstWebApplication.Models

{

    [Table("Customer")]

    public class Customer

    {

        [Key]

        public int CustomerId { get; set; }

        [Required]

        public string Name { get; set; }

        public ICollection<Order> Orders { get; set; }

    }

}

**Order.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace ModelFirstWebApplication.Models

{

    [Table("Order")]

    public class Order

    {

        [Key]

        public int OrderId { get; set; }

        [Required]

        [DataType(DataType.Date)]

        public string Date { get; set; }

        [Required]

        public string OrderState { get; set; }

        [Required]

        [ForeignKey("Customer")]

        public int CustomerId { get; set; }

        public Customer Customer { get; set; }

        public ICollection<OrderDetail> OrderDetails { get; set; }

    }

}

**OrderDetails.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace ModelFirstWebApplication.Models

{

    [Table("OrderDetail")]

    public class OrderDetail

    {

        [Key]

        public int OrderDetailId { get; set; }

        [Required]

        public int Count { get; set; }

        [Required]

        [Column(TypeName = "decimal(18,2)")]

        public decimal Amount { get; set; }

        [Required]

        [ForeignKey("Order")]

        public int OrderId { get; set; }

        public Order Order { get; set; }

        [Required]

        [ForeignKey("Product")]

        public int ProductId { get; set; }

        public Product Product { get; set; }

    }

}

**Product.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace ModelFirstWebApplication.Models

{

    [Table("Product")]

    public class Product

    {

        [Key]

        public int ProductId { get; set; }

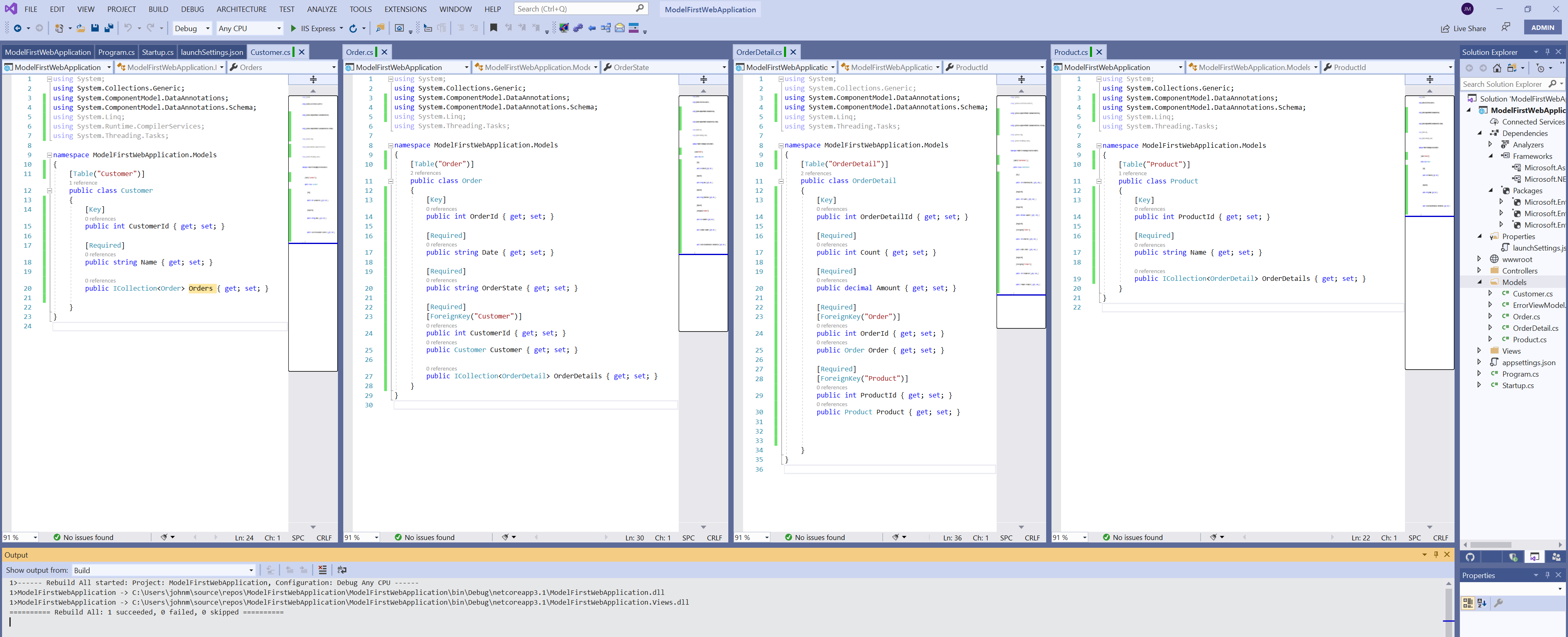
        [Required]

        public string Name { get; set; }

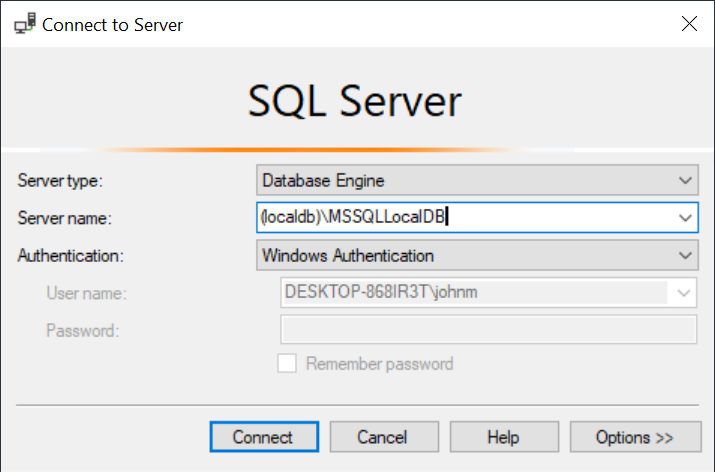
        public ICollection<OrderDetail> OrderDetails { get; set; }

    }

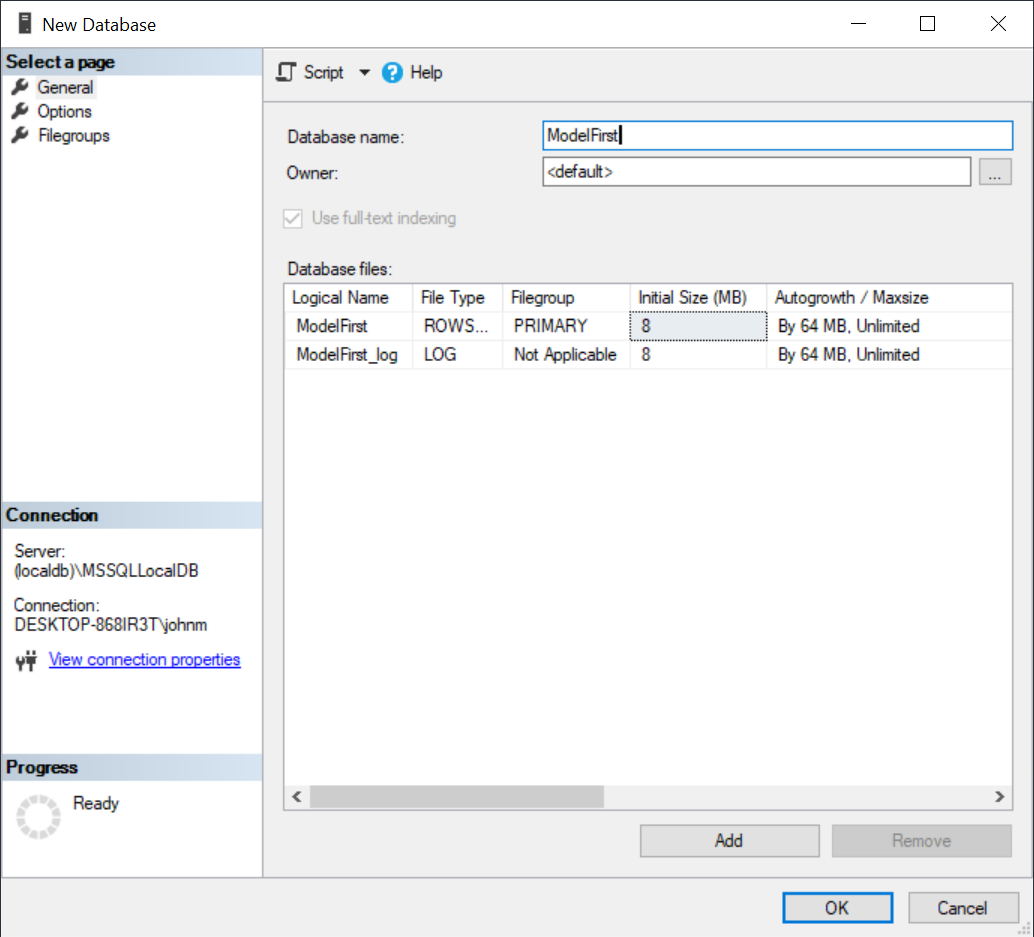
}



(3) Create the SQL Server database (just the database - no tables, these will come from Model First approach in ASP.NET Core MVC and Entity Framework Core). Use SSMS (Sql Server Management Studio) to create the database:



Create a new database



(4) Create the ConnectionString in appsettings.json

{

  "Logging": {

    "LogLevel": {

      "Default": "Information",

      "Microsoft": "Warning",

      "Microsoft.Hosting.Lifetime": "Information"

    }

  },

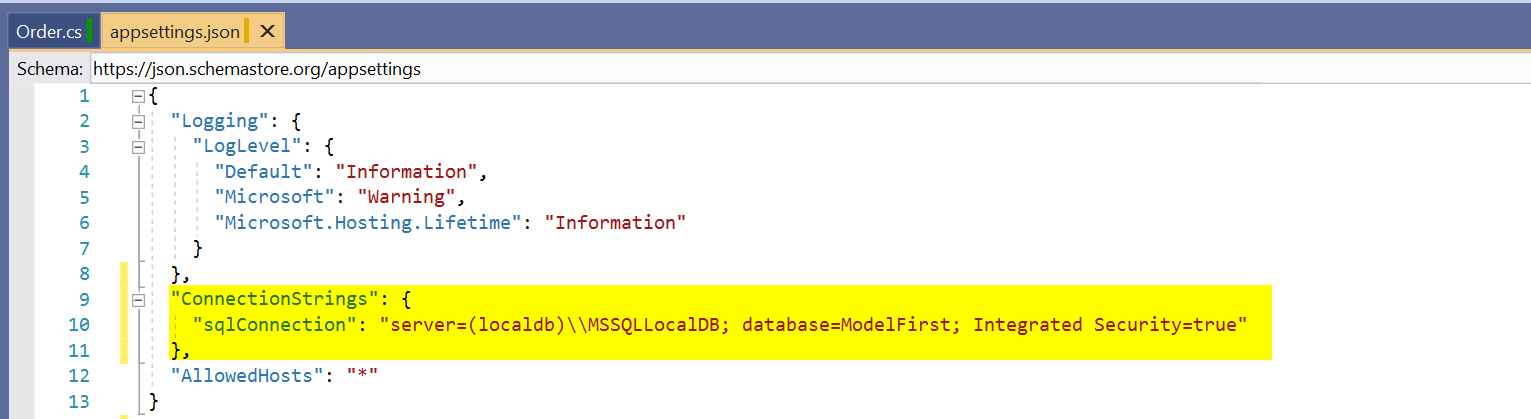
  "ConnectionStrings": {

    "sqlConnection": "server=(localdb)\\MSSQLLocalDB; database=ModelFirst; Integrated Security=true"

  },

  "AllowedHosts": "\*"

}



(5) In the **ConfigureServices** method of **Startup.cs** we can obtain the connection string through the IConfiguration interface done through the IOC container.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.SqlServer;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using ModelFirstWebApplication.Repository;

namespace ModelFirstWebApplication

{

    public class Startup

    {

        public Startup(IConfiguration configuration)

        {

            Configuration = configuration;

        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.

        public void ConfigureServices(IServiceCollection services)

        {

            services.AddDbContext<RepositoryContext>(opts => opts.UseSqlServer(Configuration.GetConnectionString("sqlConnection")));

            services.AddControllersWithViews();

        }

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

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

        {

            if (env.IsDevelopment())

            {

                app.UseDeveloperExceptionPage();

            }

            else

            {

                app.UseExceptionHandler("/Home/Error");

            }

            app.UseStaticFiles();

            app.UseRouting();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>

            {

                endpoints.MapControllerRoute(

                    name: "default",

                    pattern: "{controller=Home}/{action=Index}/{id?}");

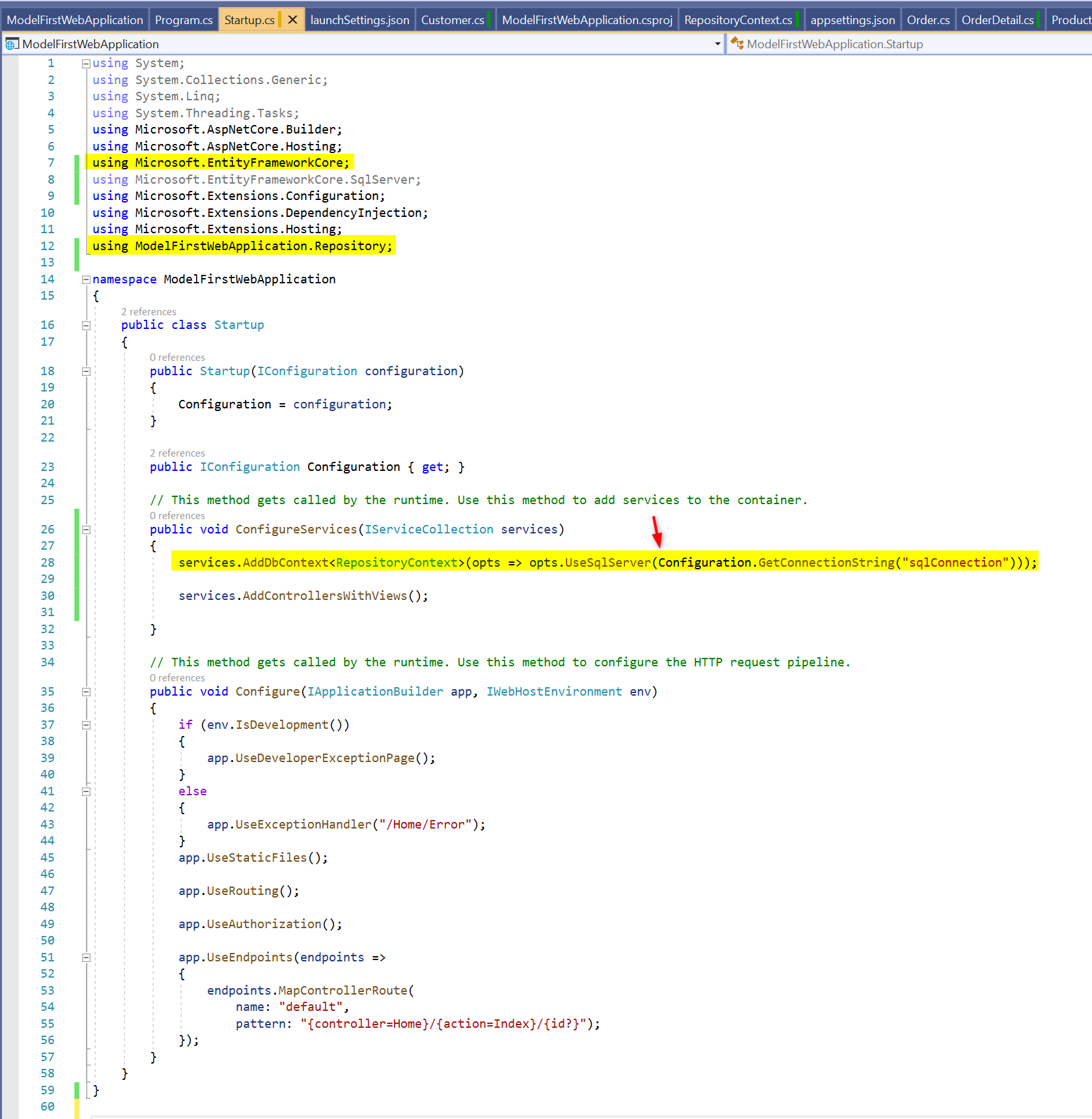
            });

        }

    }

}

These are the only changes that we need to make (highlighted)



(6) Create the context class (RepositoryContext) which is the middleware component for communication with the database. This inherit's from EntityFrameworkCore's DbContext

**RepositoryContext.cs**

using Microsoft.EntityFrameworkCore;

using ModelFirstWebApplication.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ModelFirstWebApplication.Repository

{

    public class RepositoryContext : DbContext

    {

        public RepositoryContext(DbContextOptions options) : base(options) { }

        public DbSet<Customer> Customer { get; set; }

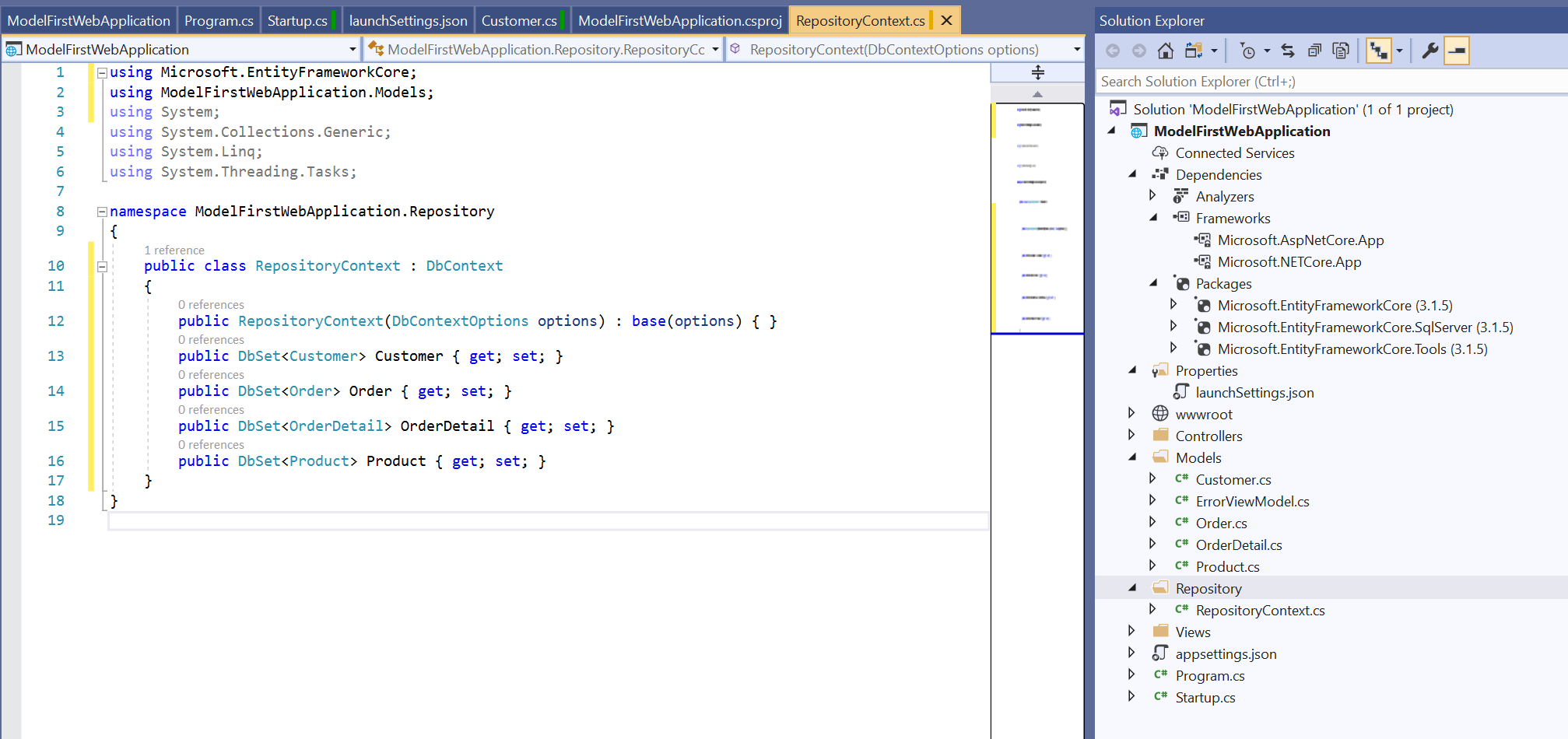
        public DbSet<Order> Order { get; set; }

        public DbSet<OrderDetail> OrderDetail { get; set; }

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

    }

}



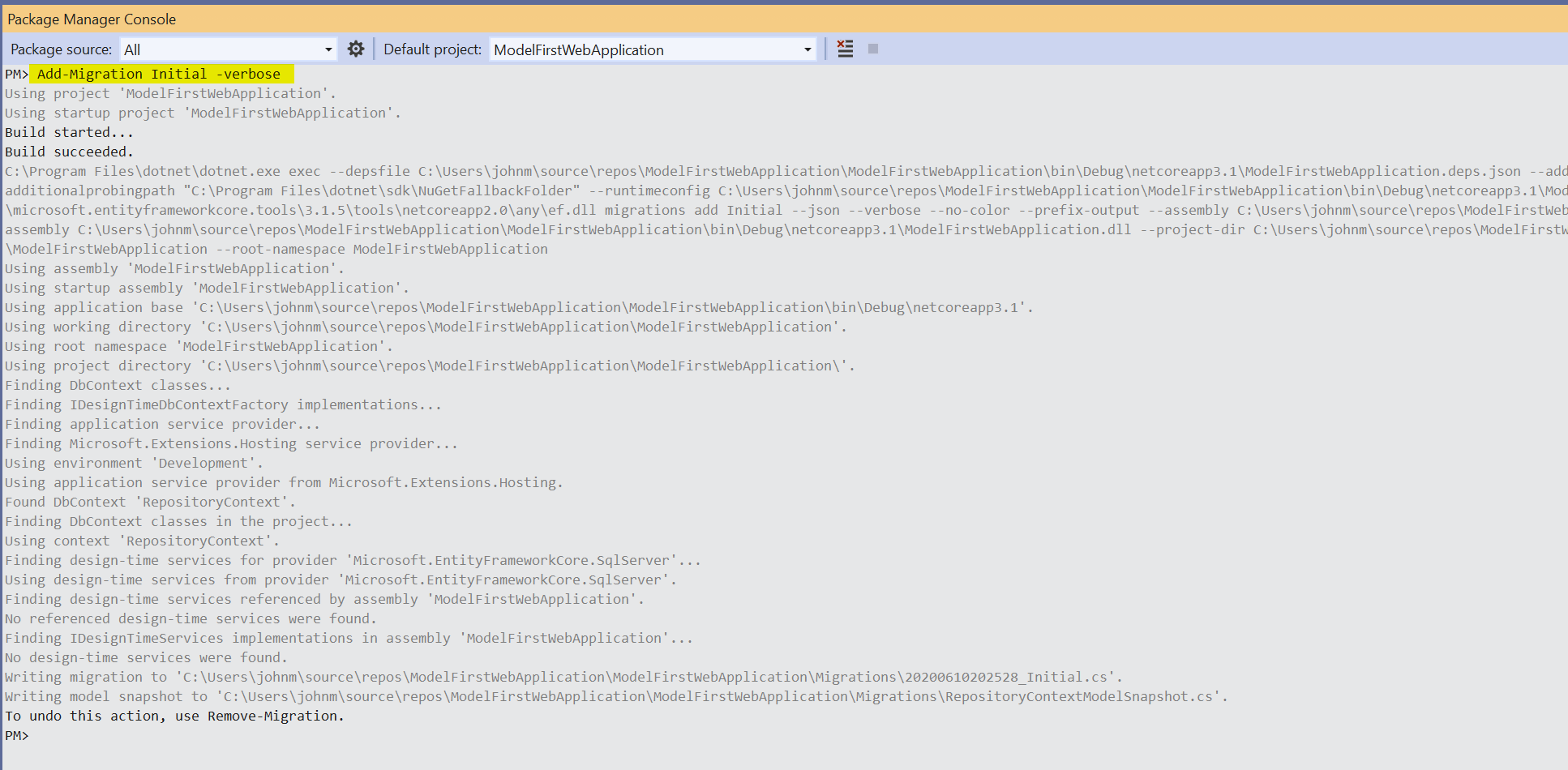
(6) Build the database tables (including foreign keys and integrity) using package manager console

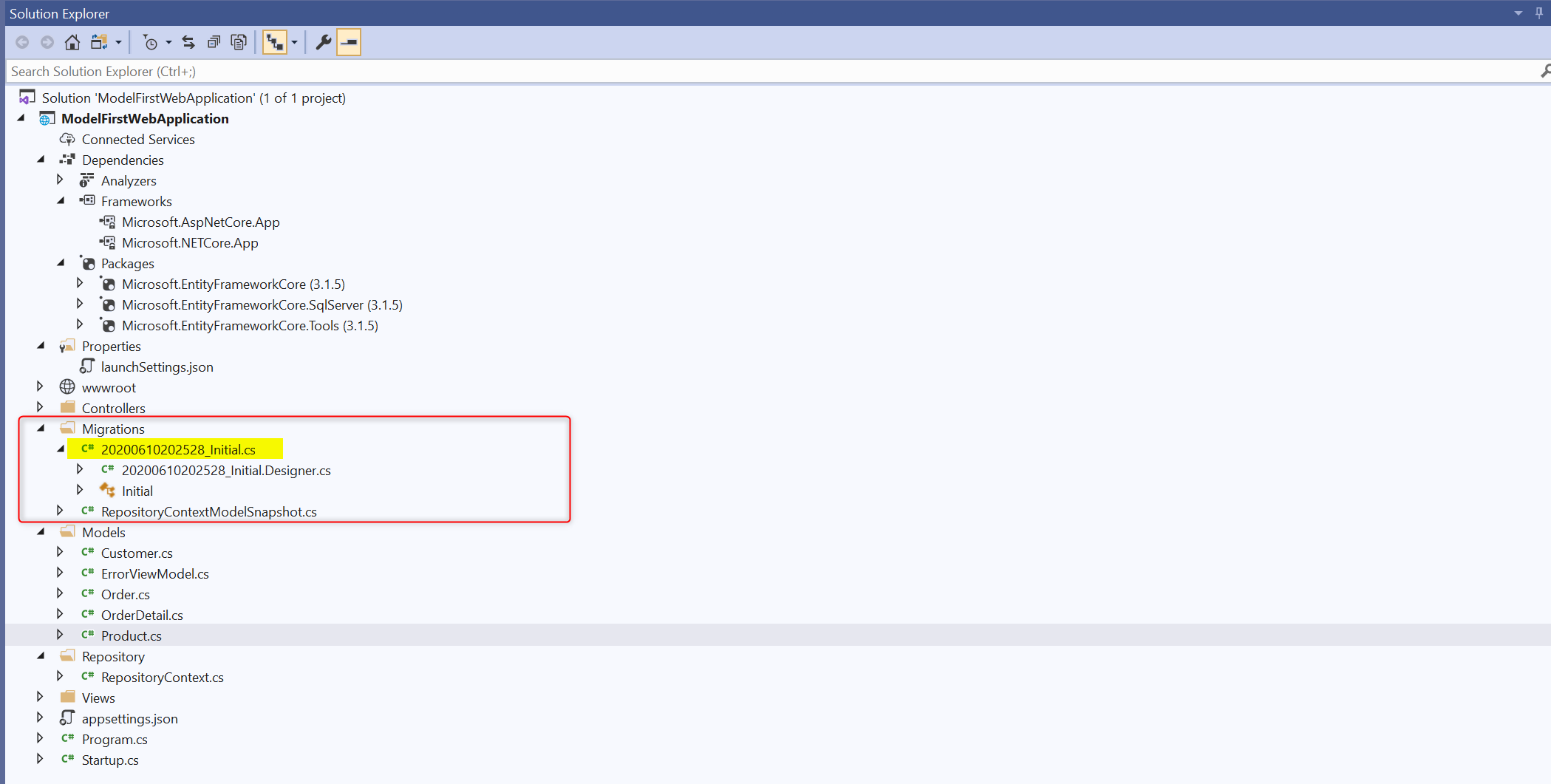
**BEFORE**



You do NOT need to enable-migrations; just do the following:

PM> Add-Migration Initial -verbose







using Microsoft.EntityFrameworkCore.Migrations;

namespace ModelFirstWebApplication.Migrations

{

    public partial class Initial : Migration

    {

        protected override void Up(MigrationBuilder migrationBuilder)

        {

            migrationBuilder.CreateTable(

                name: "Customer",

                columns: table => new

                {

                    CustomerId = table.Column<int>(nullable: false)

                        .Annotation("SqlServer:Identity", "1, 1"),

                    Name = table.Column<string>(nullable: false)

                },

                constraints: table =>

                {

                    table.PrimaryKey("PK\_Customer", x => x.CustomerId);

                });

            migrationBuilder.CreateTable(

                name: "Product",

                columns: table => new

                {

                    ProductId = table.Column<int>(nullable: false)

                        .Annotation("SqlServer:Identity", "1, 1"),

                    Name = table.Column<string>(nullable: false)

                },

                constraints: table =>

                {

                    table.PrimaryKey("PK\_Product", x => x.ProductId);

                });

            migrationBuilder.CreateTable(

                name: "Order",

                columns: table => new

                {

                    OrderId = table.Column<int>(nullable: false)

                        .Annotation("SqlServer:Identity", "1, 1"),

                    Date = table.Column<string>(nullable: false),

                    OrderState = table.Column<string>(nullable: false),

                    CustomerId = table.Column<int>(nullable: false)

                },

                constraints: table =>

                {

                    table.PrimaryKey("PK\_Order", x => x.OrderId);

                    table.ForeignKey(

                        name: "FK\_Order\_Customer\_CustomerId",

                        column: x => x.CustomerId,

                        principalTable: "Customer",

                        principalColumn: "CustomerId",

                        onDelete: ReferentialAction.Cascade);

                });

            migrationBuilder.CreateTable(

                name: "OrderDetail",

                columns: table => new

                {

                    OrderDetailId = table.Column<int>(nullable: false)

                        .Annotation("SqlServer:Identity", "1, 1"),

                    Count = table.Column<int>(nullable: false),

                    Amount = table.Column<decimal>(type: "decimal(18,2)", nullable: false),

                    OrderId = table.Column<int>(nullable: false),

                    ProductId = table.Column<int>(nullable: false)

                },

                constraints: table =>

                {

                    table.PrimaryKey("PK\_OrderDetail", x => x.OrderDetailId);

                    table.ForeignKey(

                        name: "FK\_OrderDetail\_Order\_OrderId",

                        column: x => x.OrderId,

                        principalTable: "Order",

                        principalColumn: "OrderId",

                        onDelete: ReferentialAction.Cascade);

                    table.ForeignKey(

                        name: "FK\_OrderDetail\_Product\_ProductId",

                        column: x => x.ProductId,

                        principalTable: "Product",

                        principalColumn: "ProductId",

                        onDelete: ReferentialAction.Cascade);

                });

            migrationBuilder.CreateIndex(

                name: "IX\_Order\_CustomerId",

                table: "Order",

                column: "CustomerId");

            migrationBuilder.CreateIndex(

                name: "IX\_OrderDetail\_OrderId",

                table: "OrderDetail",

                column: "OrderId");

            migrationBuilder.CreateIndex(

                name: "IX\_OrderDetail\_ProductId",

                table: "OrderDetail",

                column: "ProductId");

        }

        protected override void Down(MigrationBuilder migrationBuilder)

        {

            migrationBuilder.DropTable(

                name: "OrderDetail");

            migrationBuilder.DropTable(

                name: "Order");

            migrationBuilder.DropTable(

                name: "Product");

            migrationBuilder.DropTable(

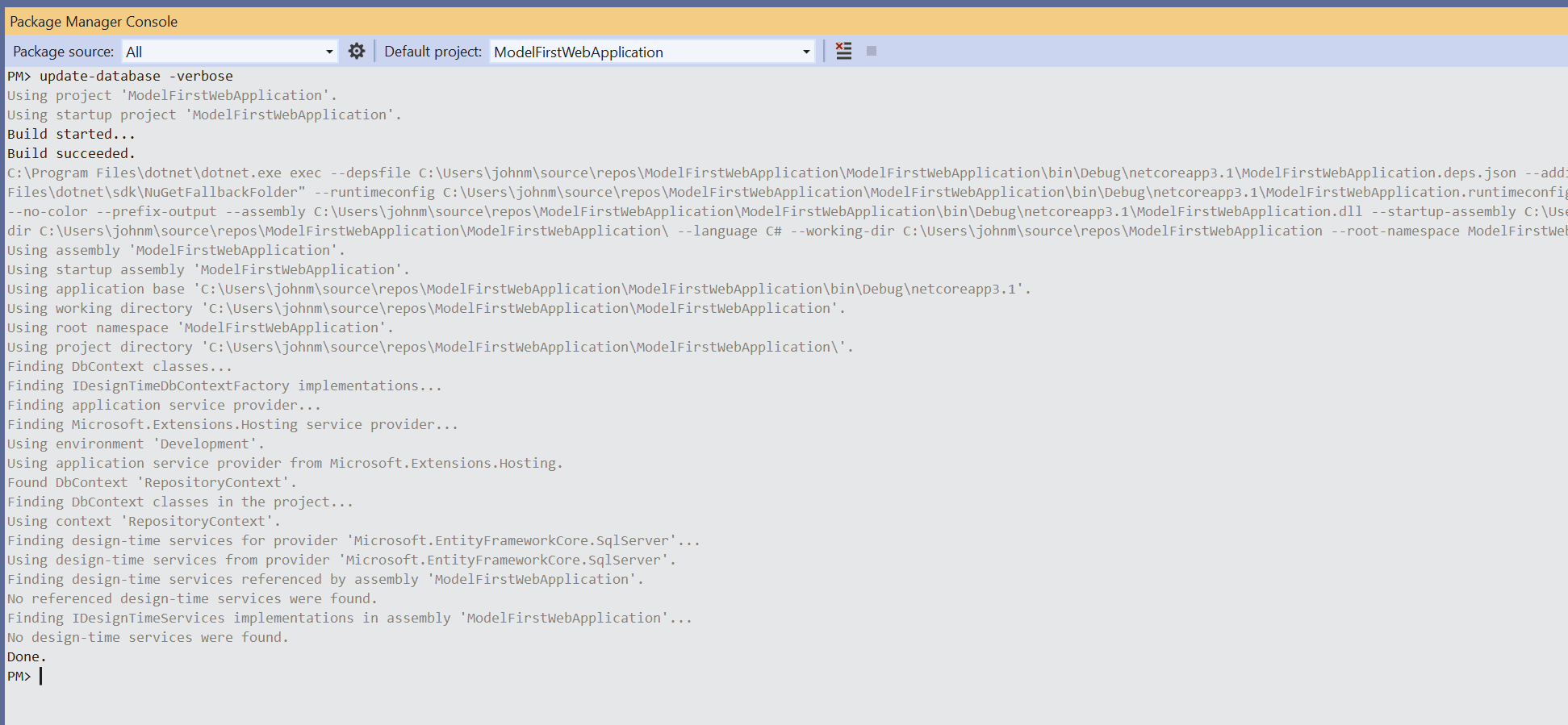
                name: "Customer");

        }

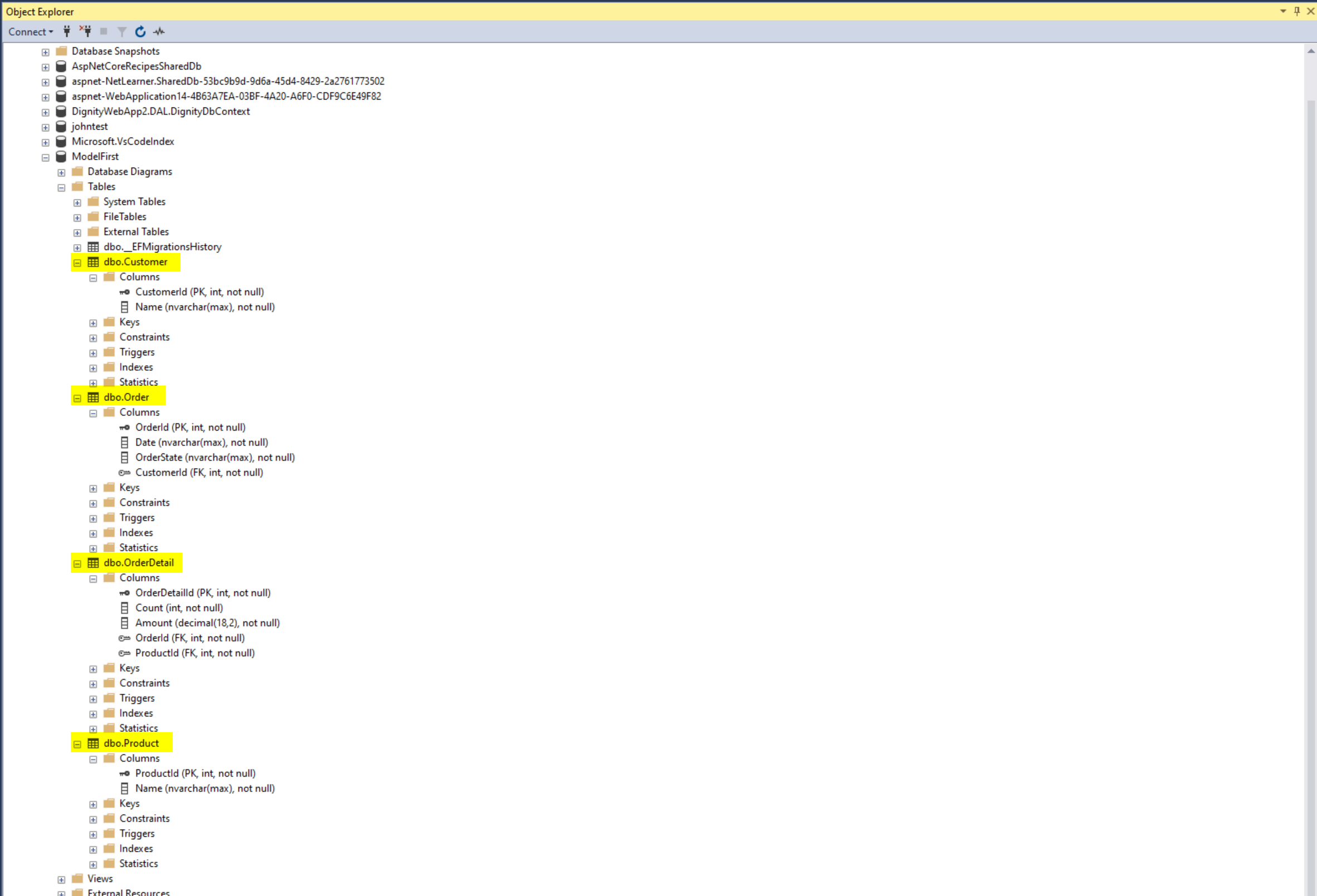
    }

}

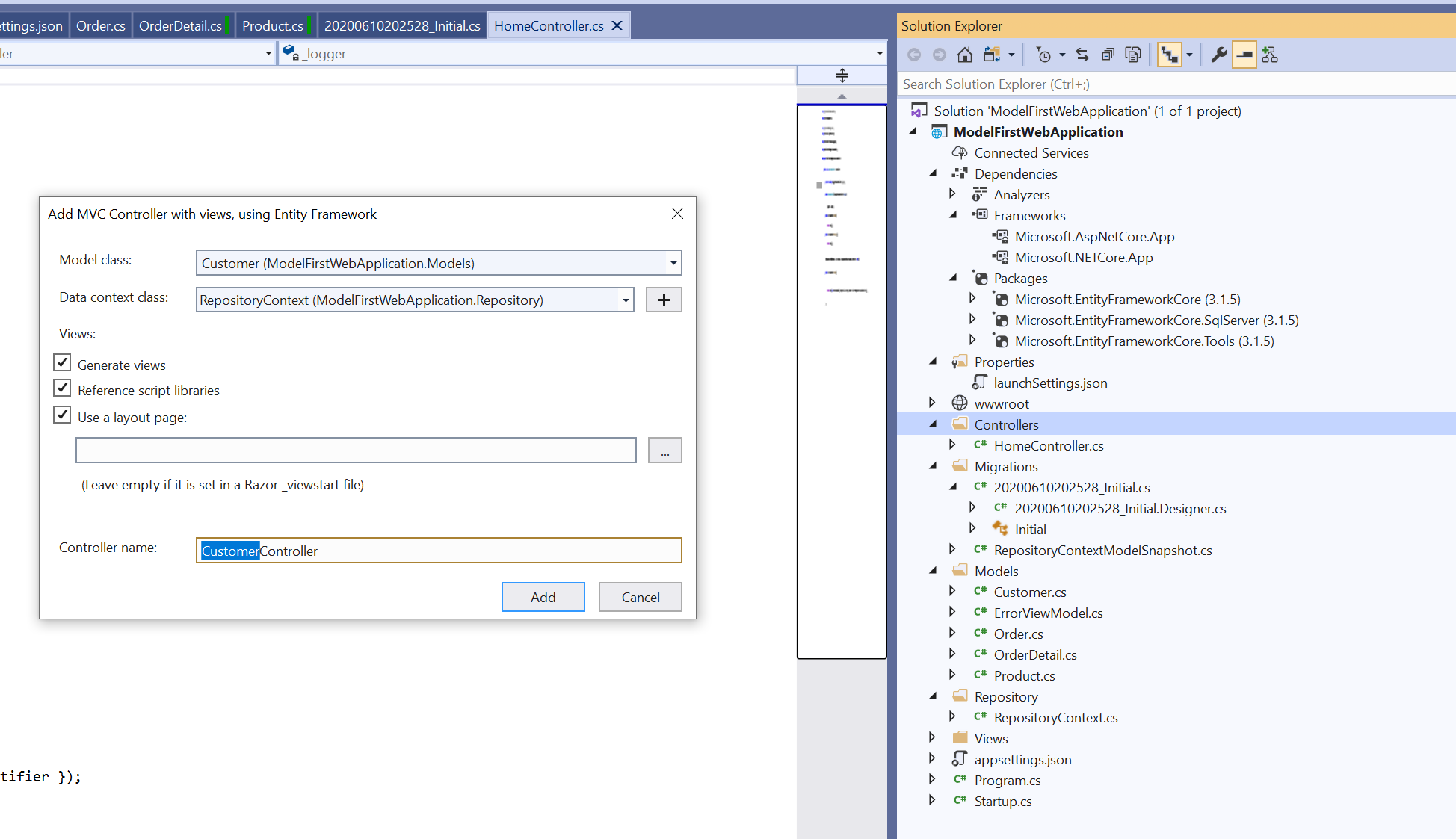
PM> update-database -verbose

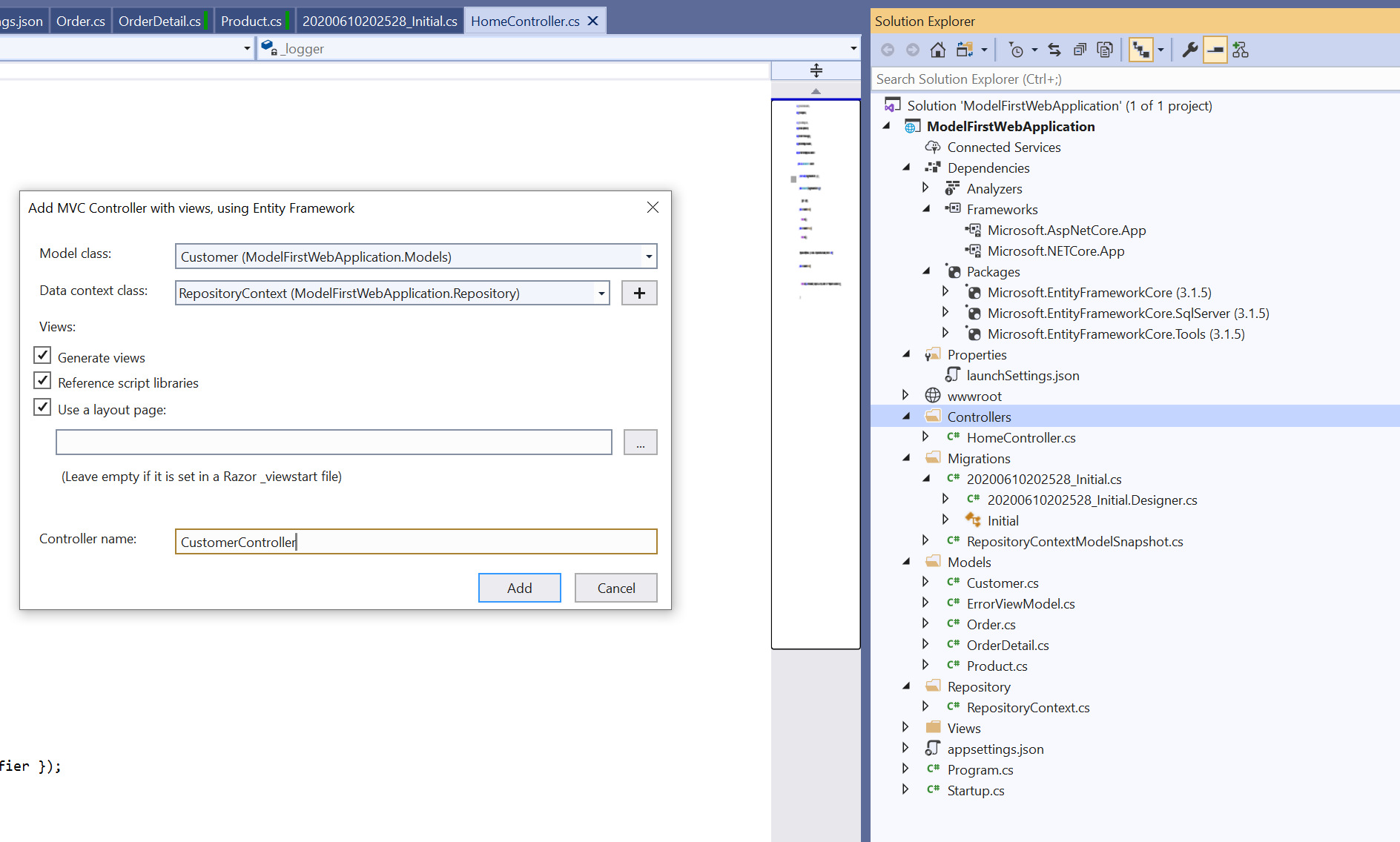


**AFTER**

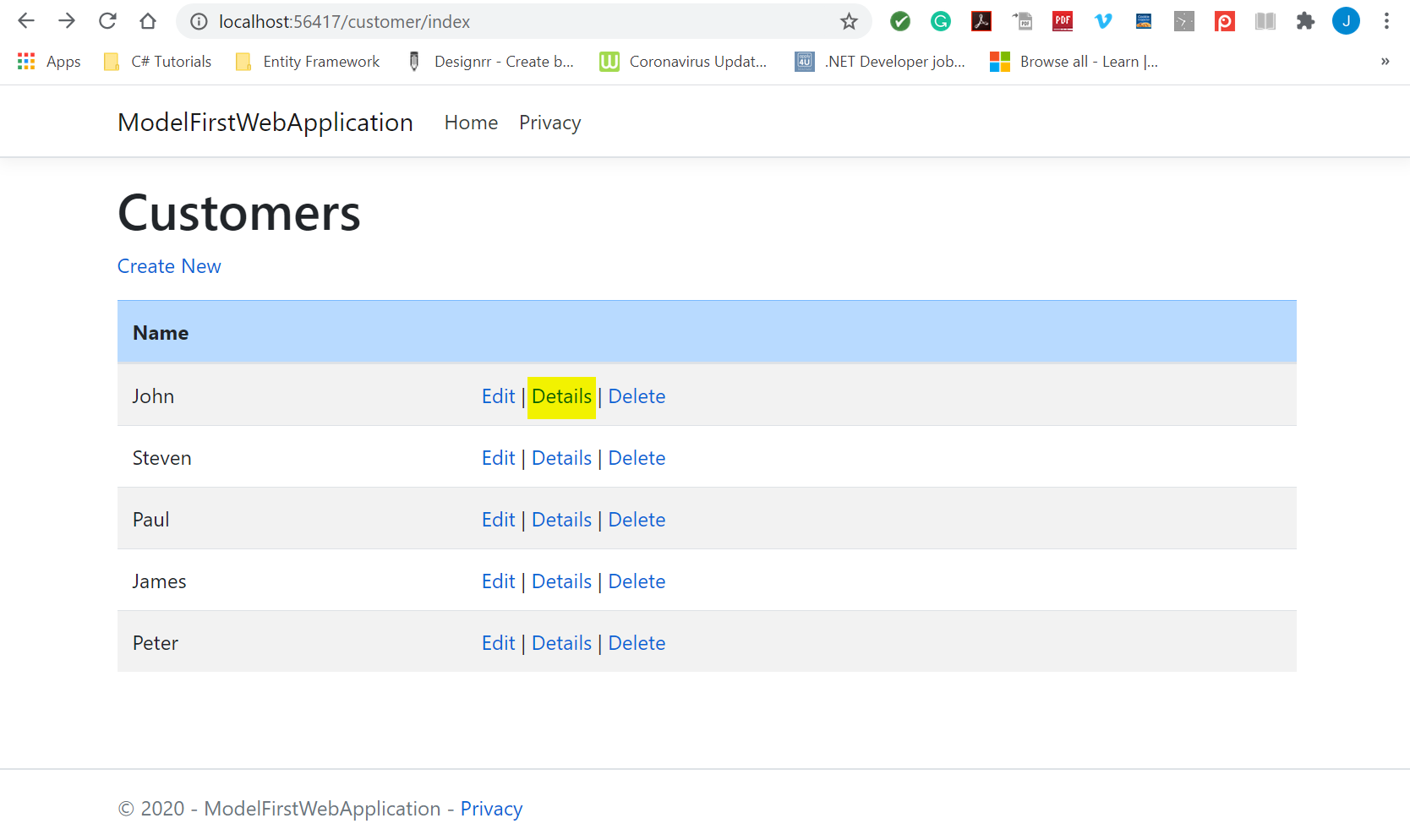


Add Controllers (automatically) for each model

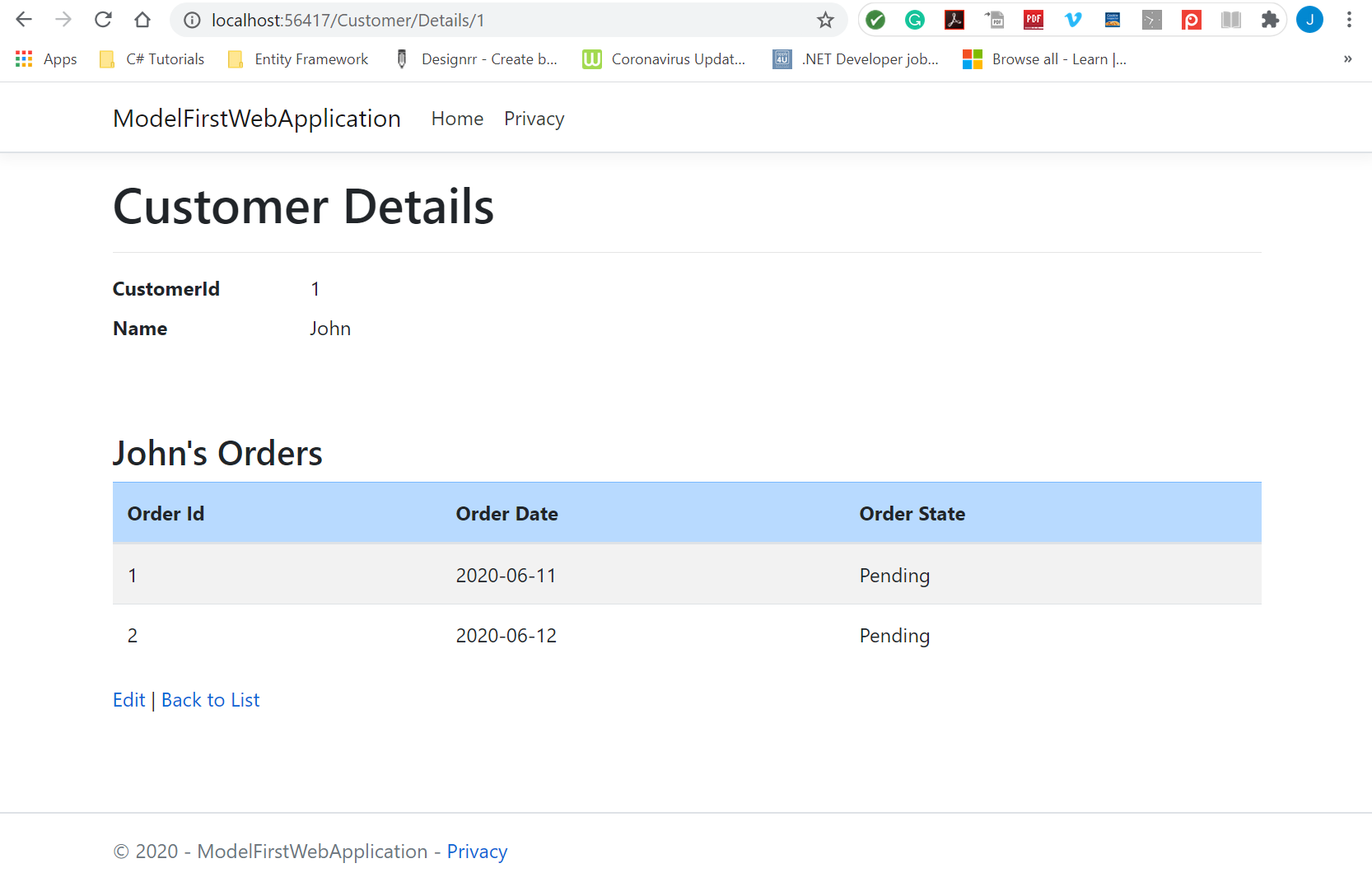




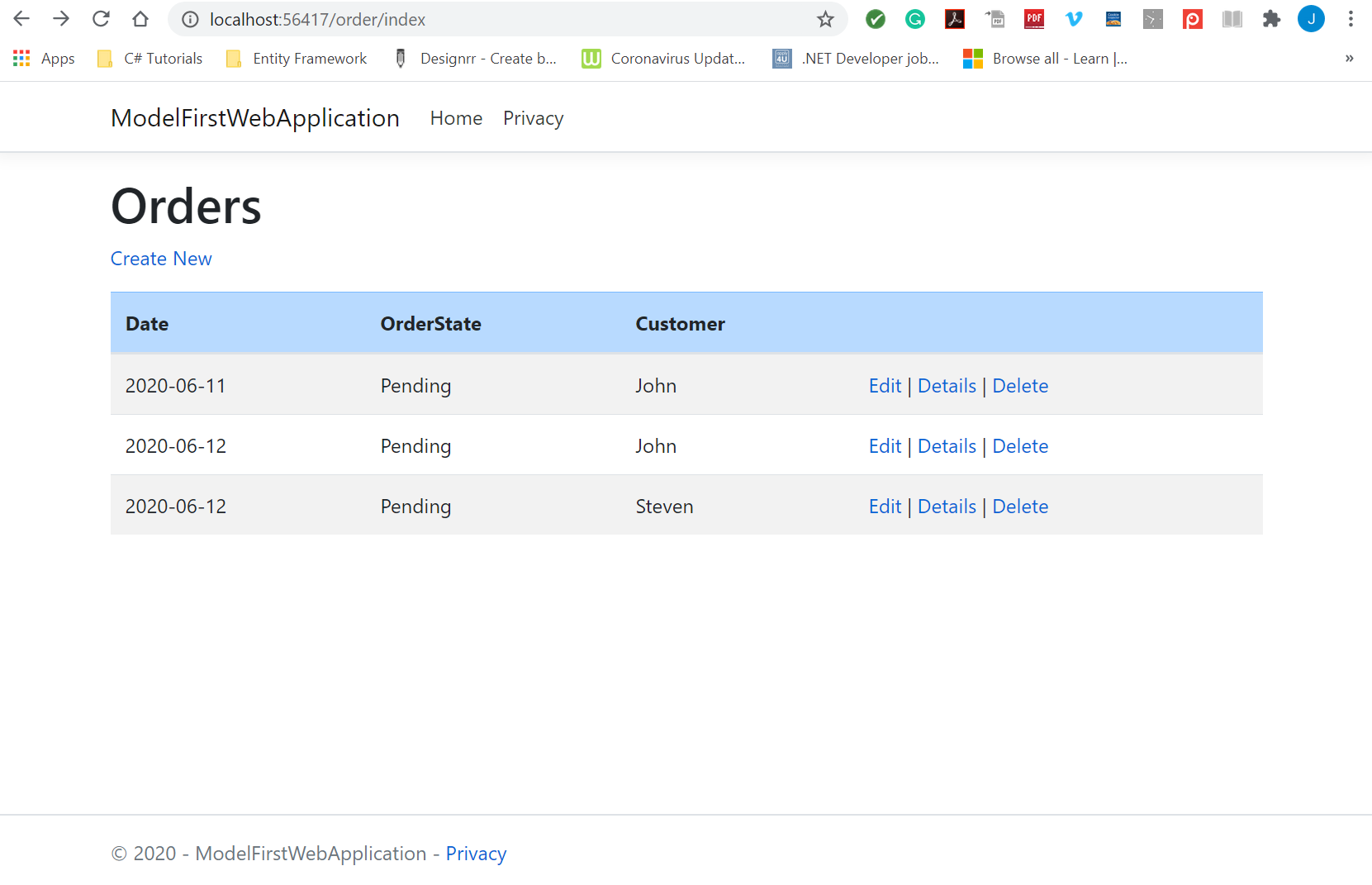
Now you can use the website (I amended the styliong using Bootstrap)



On the Details page, I've created a master-detail layout so you can see the 'Customer Details' and the 'Orders' placed by that customer on the same page



Similarly, here are the dedicated pages for the Orders



and the Order Details

