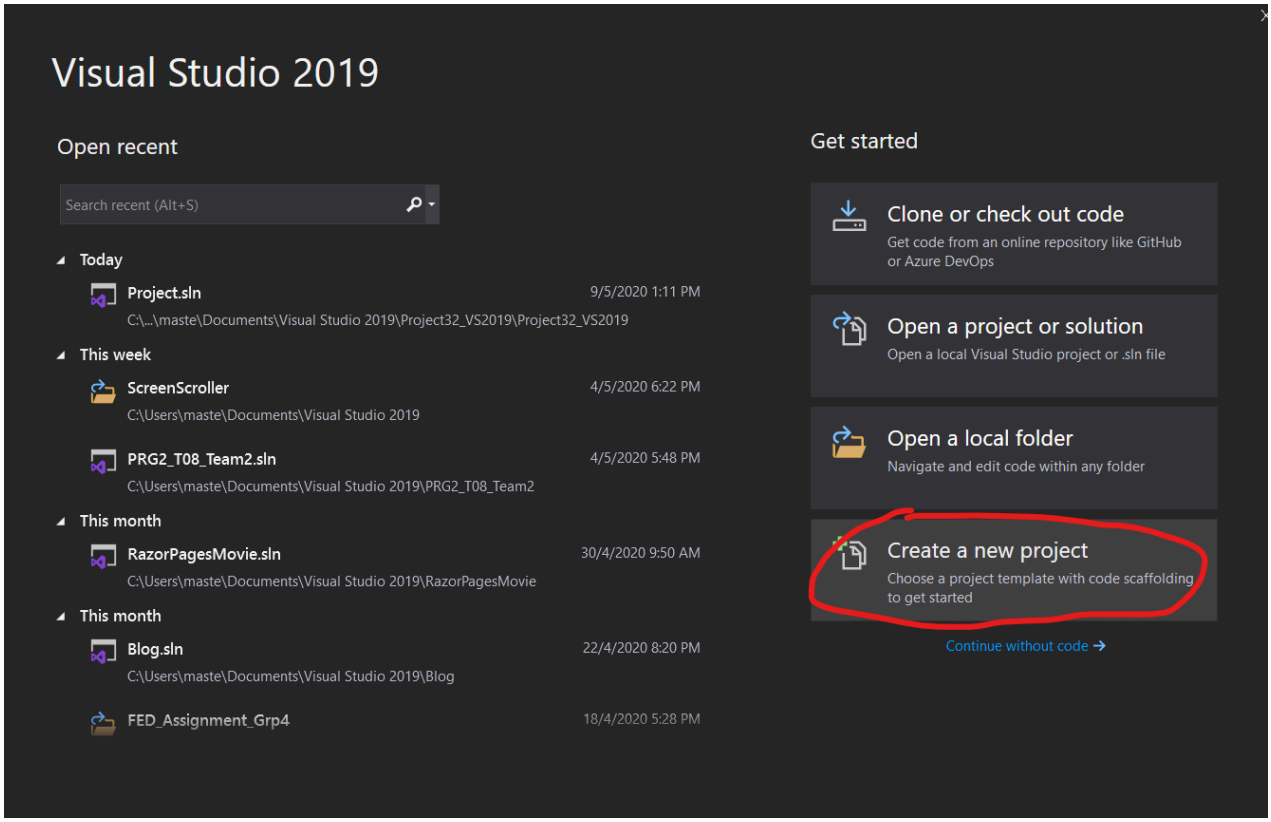
SECURE SOFTWARE DEVELOPMENT MISSION 3.3—RAZOR PAGES SETUP

NG CHIN TIONG RYAN CSF03

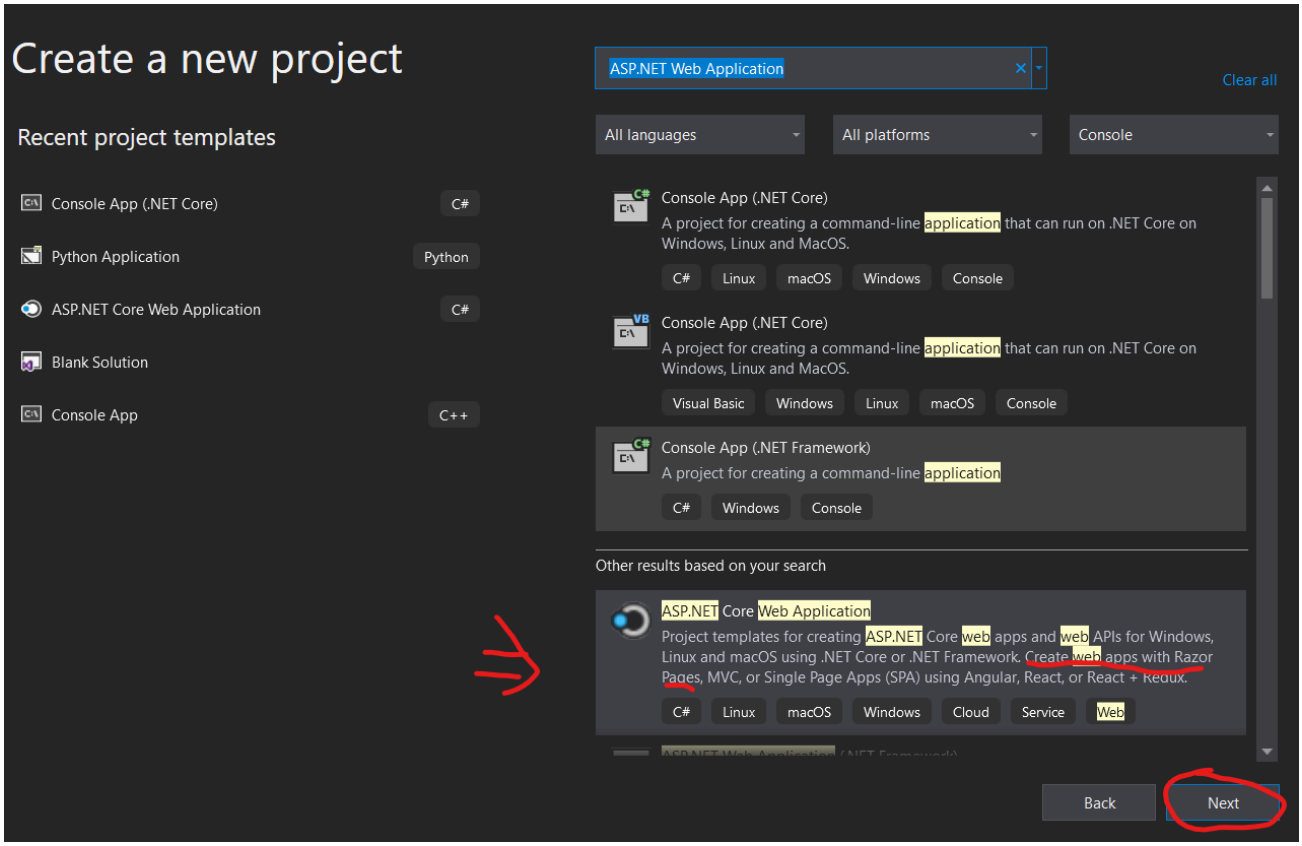
Section 1: Setup

Important Note: This instruction guide uses Visual Studio 2019 and ASP.NET Core Version 3.1. This project is named RazorPagesMovie2 and thus some of the codes may vary.

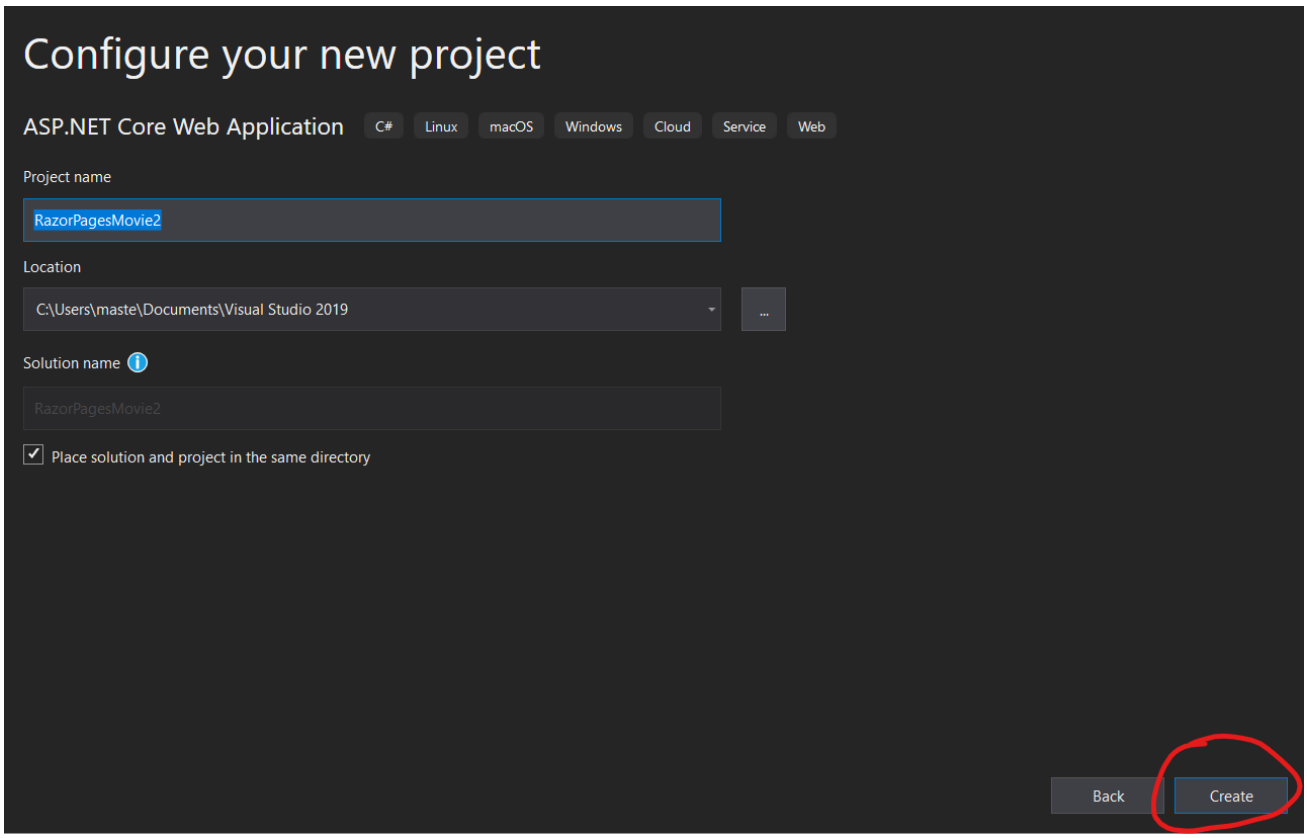
1. From the Visual Studio **File** menu, select **Create a new Project**.



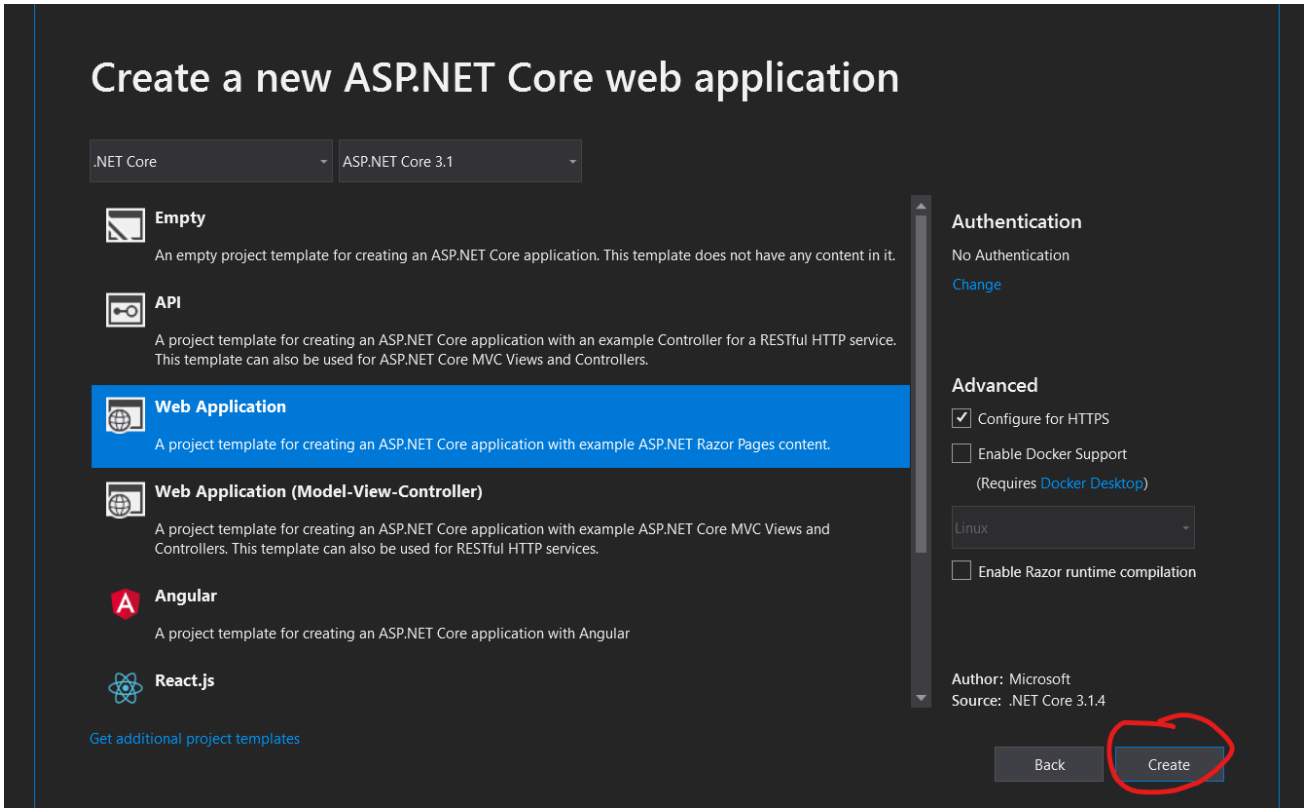
2. Search ASP.NET Core Web Application if its not already in the “Recent Project Templates”. Make sure the description has the line “Create Web Apps with Razor Pages” and select **Next**.



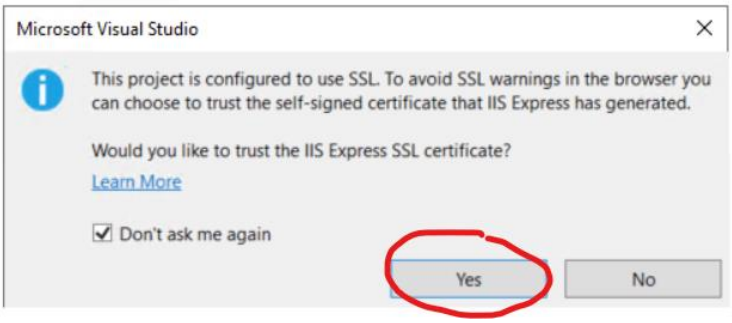
3. Name the project **RazorPagesMovie**. (In this case I am creating a second iteration of the project so I will name it **RazorPagesMovie2**) and click **Create.**

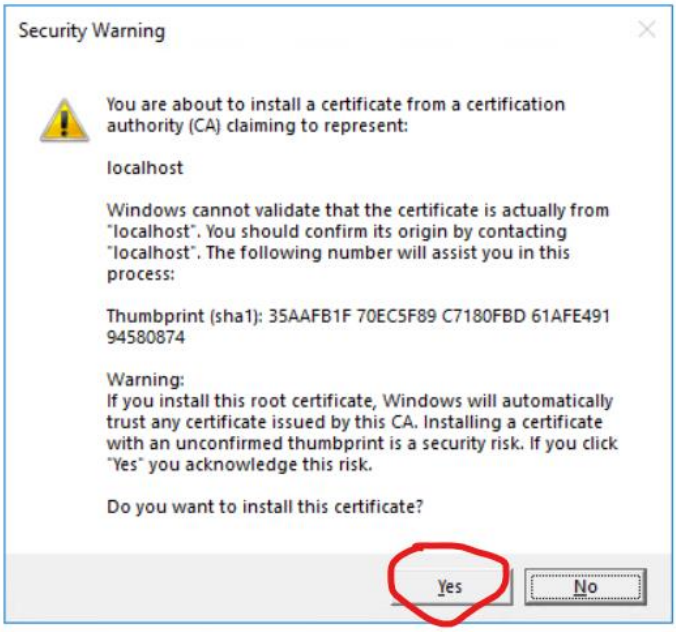


4. Select **ASP.NET Core 3.1** in the dropdown (if not already preselected), **Web Application**, and then select **Create**.



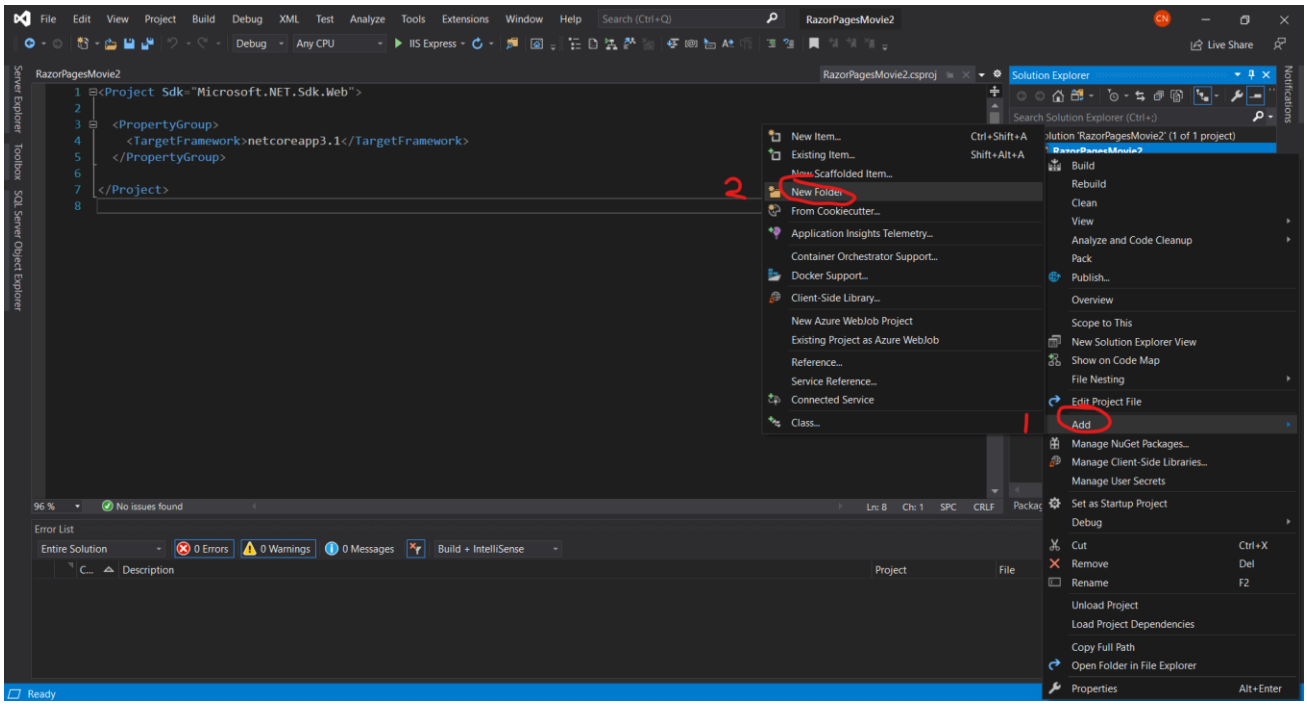
5. If this is the first time creating a Web App, Visual Studio will prompt you to accept SSL Certificates. Simply **accept** them as follows:

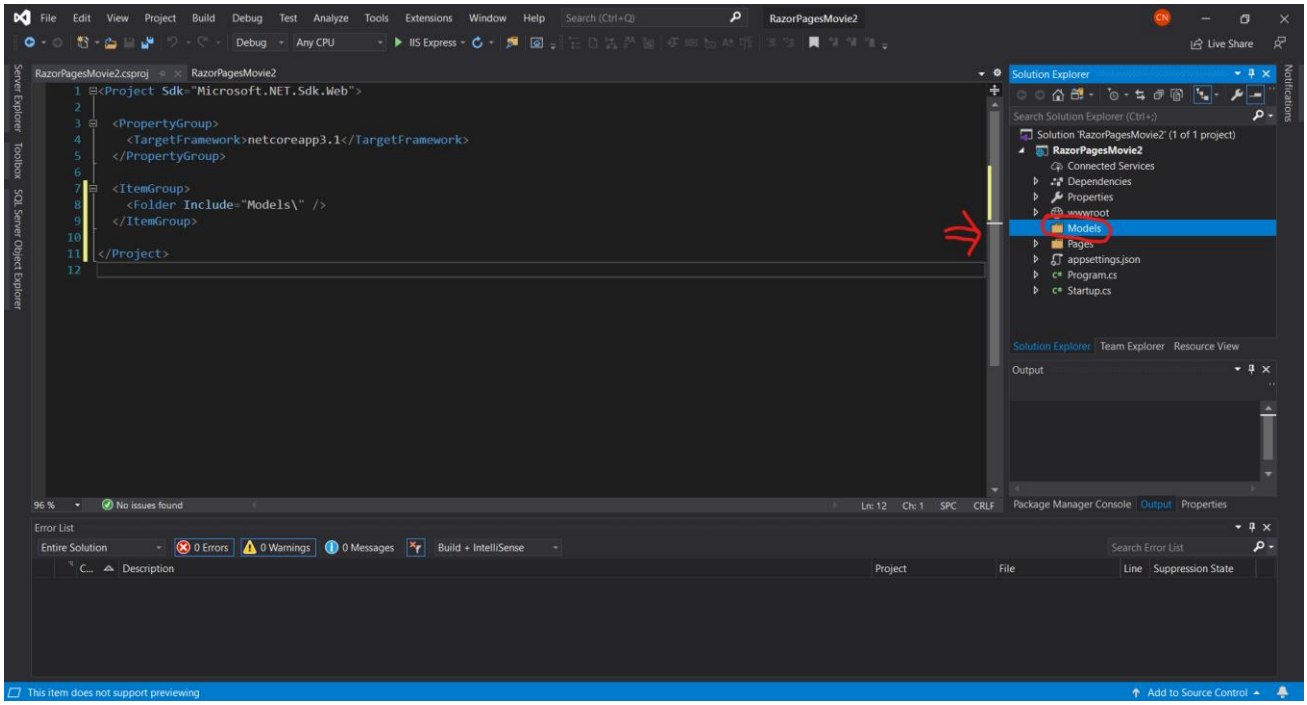




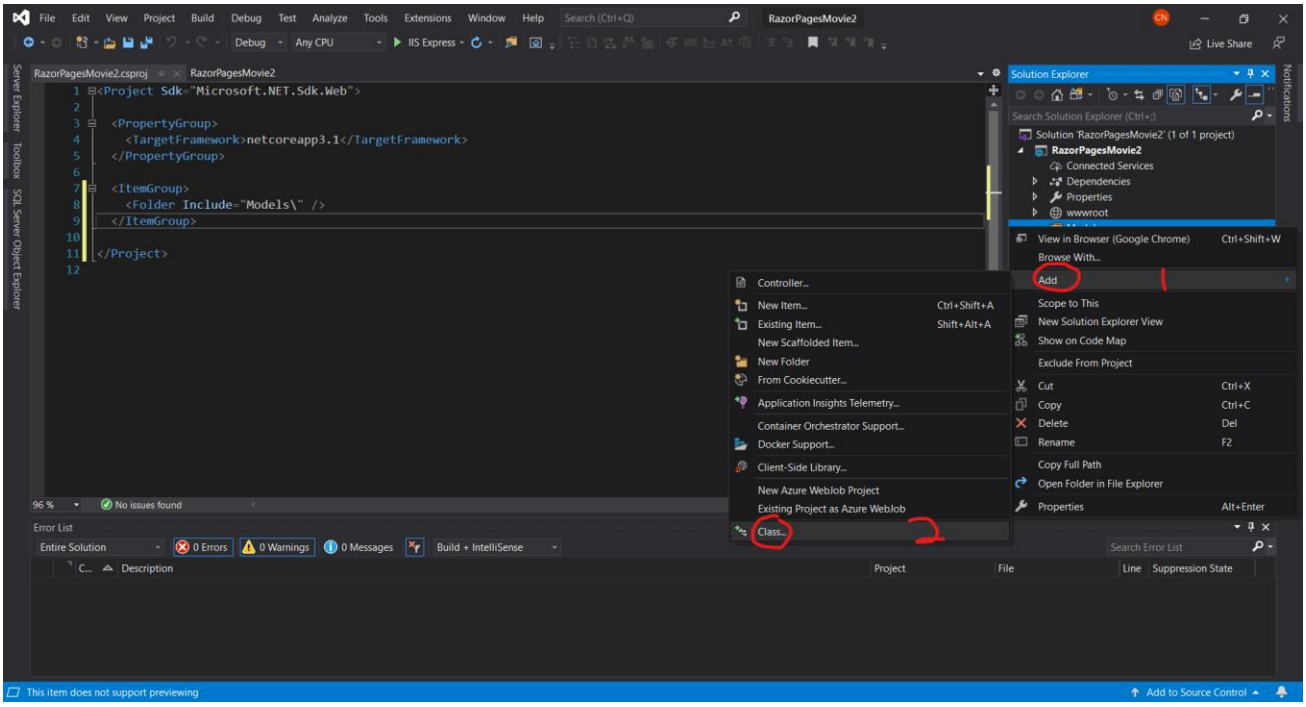
Section 2: Add a Data Model

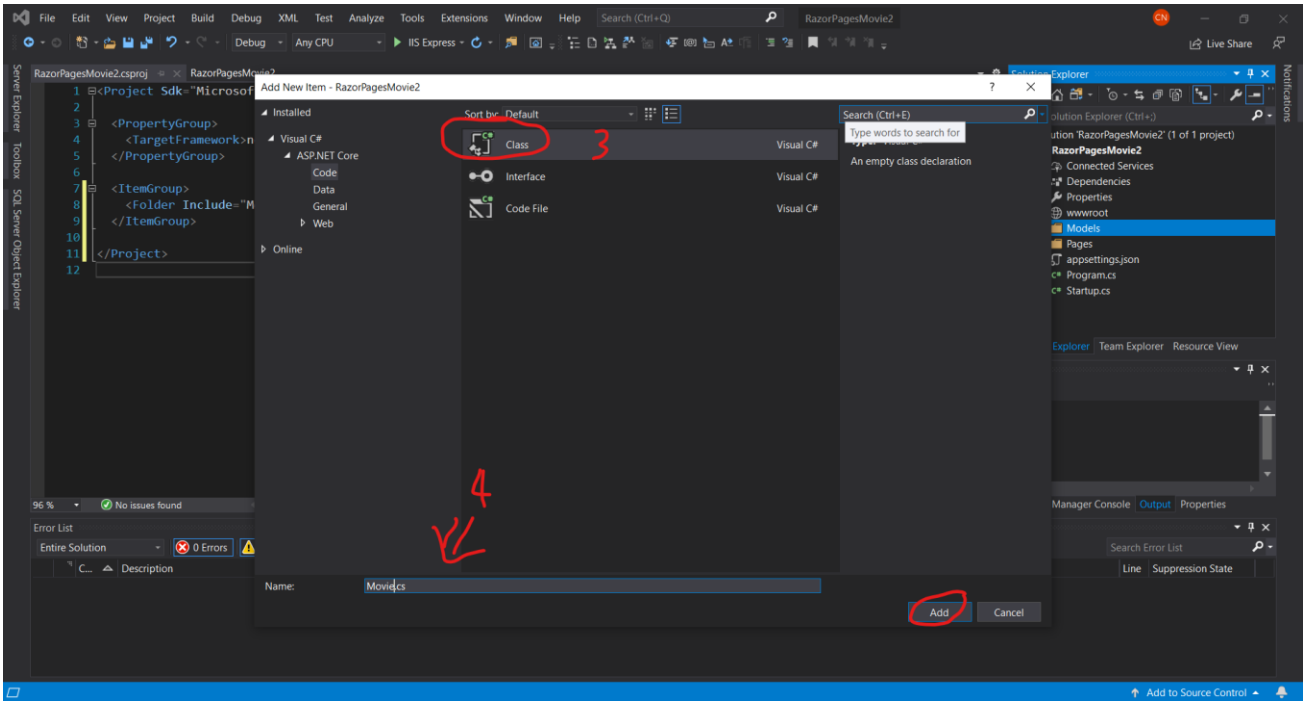
6. On the solution explorer pane, right-click the **RazorPagesMovie** project > **Add** > **New Folder**. Name the folder *Models*.



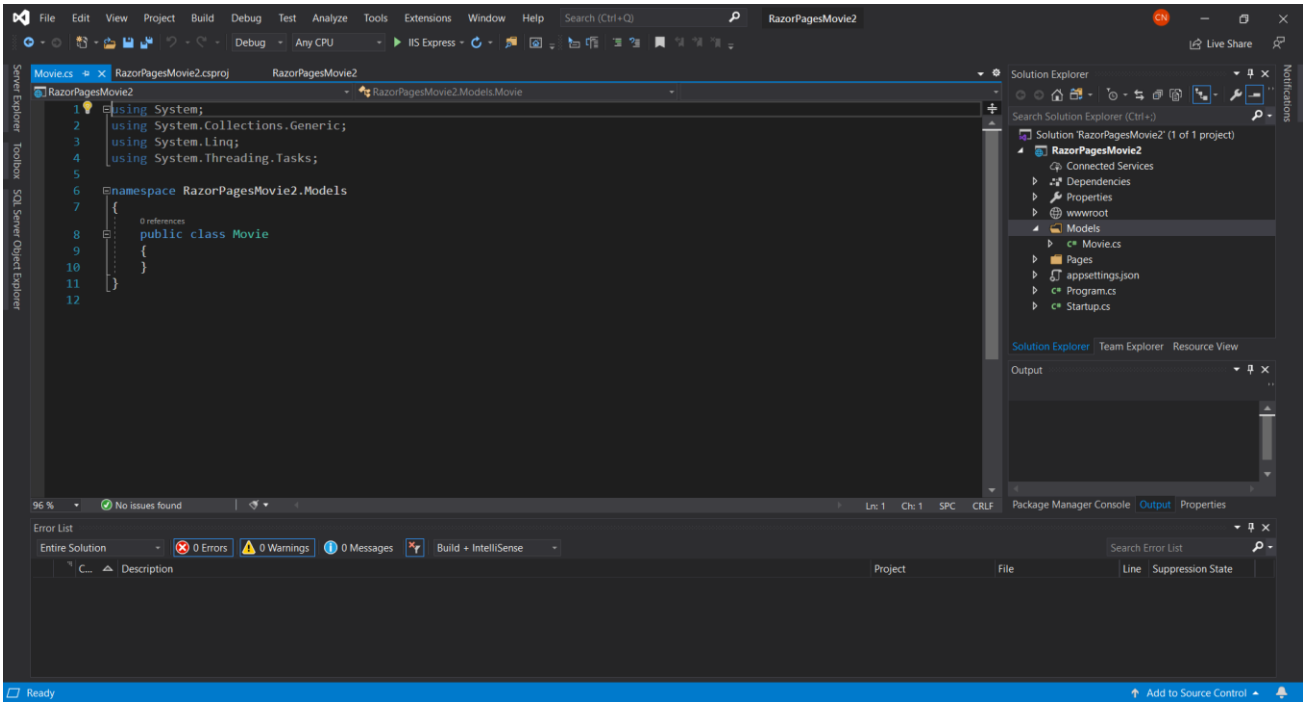


7. Right click the *Models* folder. Select **Add** > **Class**. Name the class **Movie**.





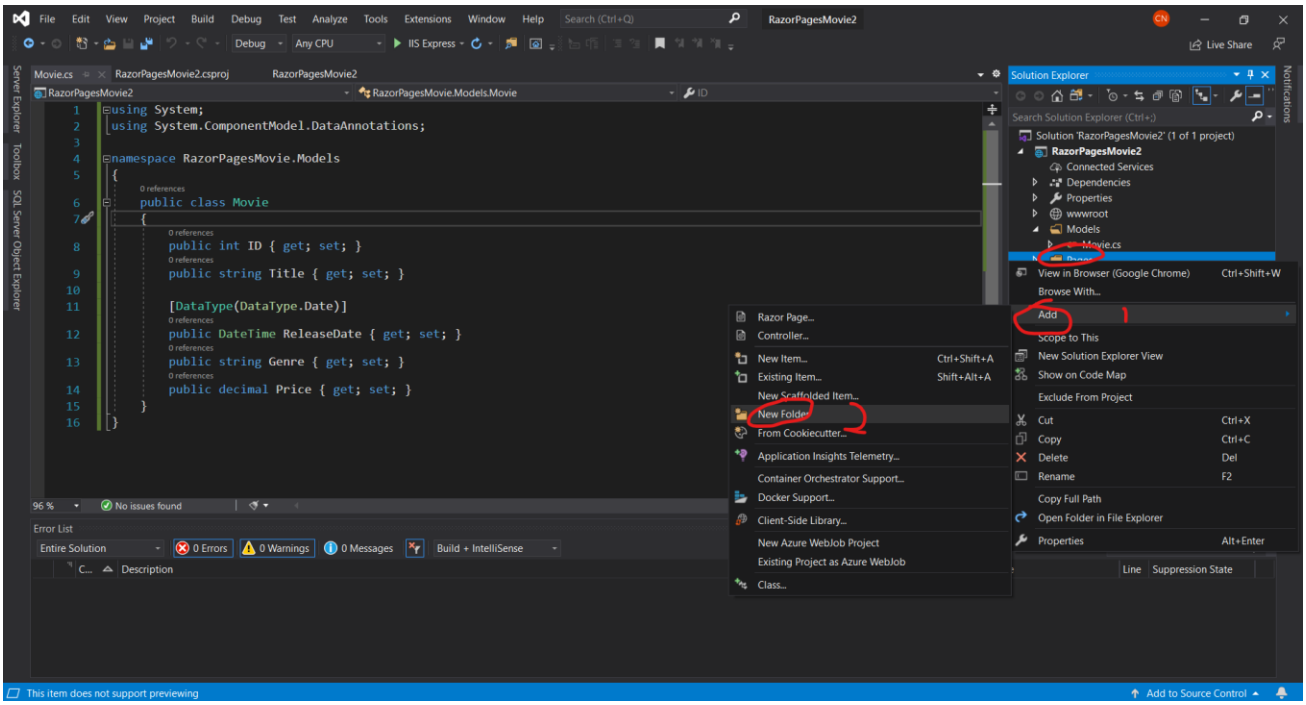
8. The boiler template for the class appears as follows. Remove all the content and paste the following code and save the file.

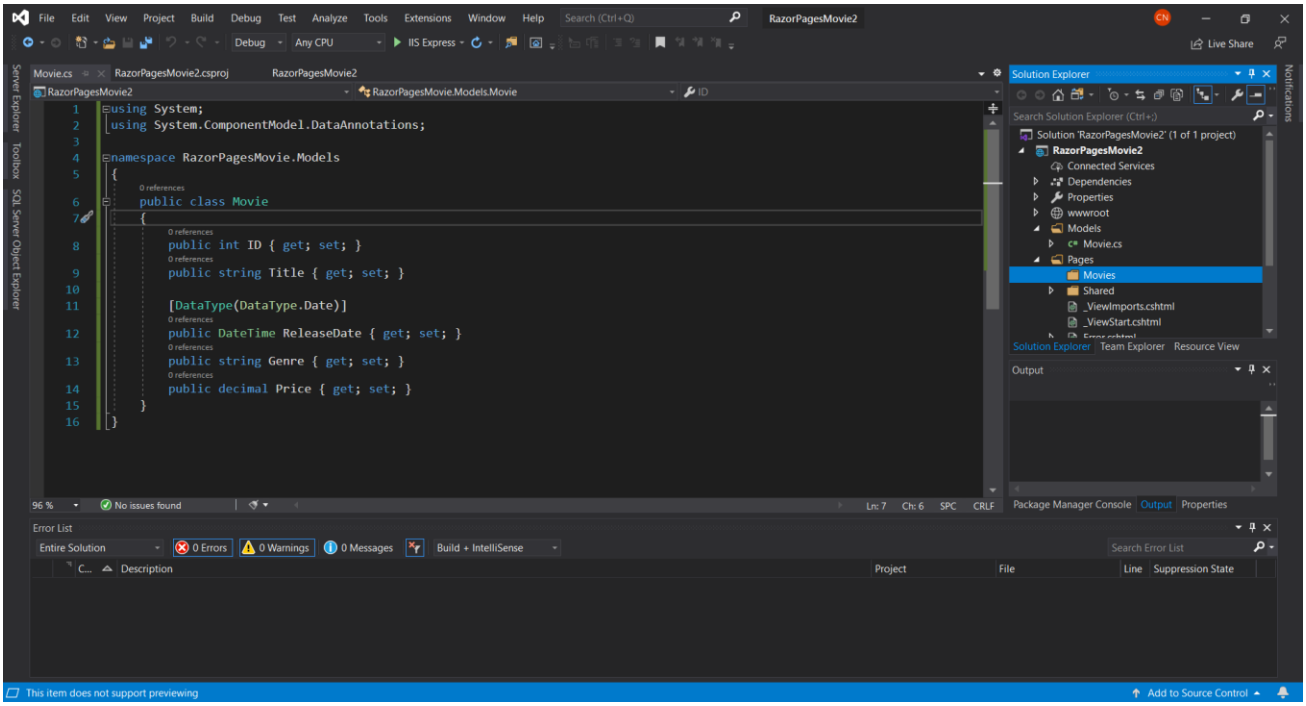



```
using System;
using System.ComponentModel.DataAnnotations;

namespace RazorPagesMovie.Models
{
    public class Movie
    {
```

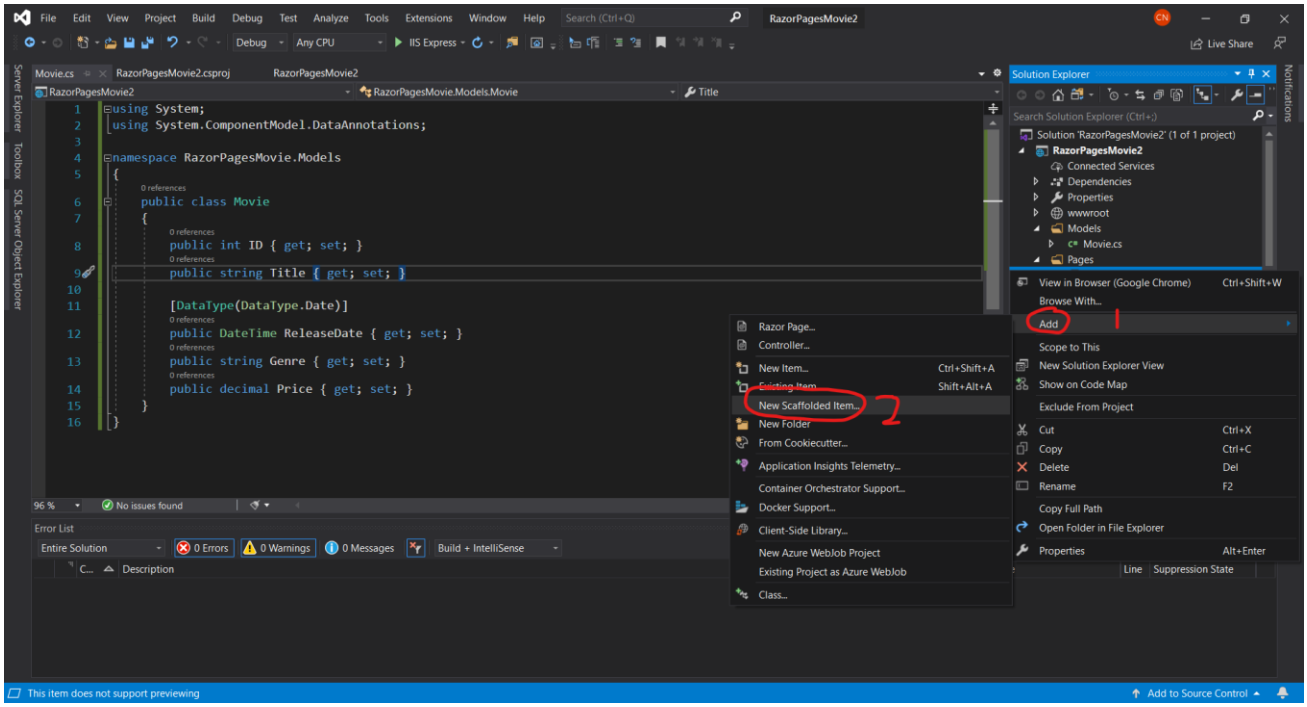
```
        public int ID { get; set; }
        public string Title { get; set; }

        [DataType(DataType.Date)]
        public DateTime ReleaseDate { get; set; }
        public string Genre { get; set; }
        public decimal Price { get; set; }
    }
}
```

9. Right click on the *Pages* folder > **Add** > **New Folder**. Name the folder *Movies*

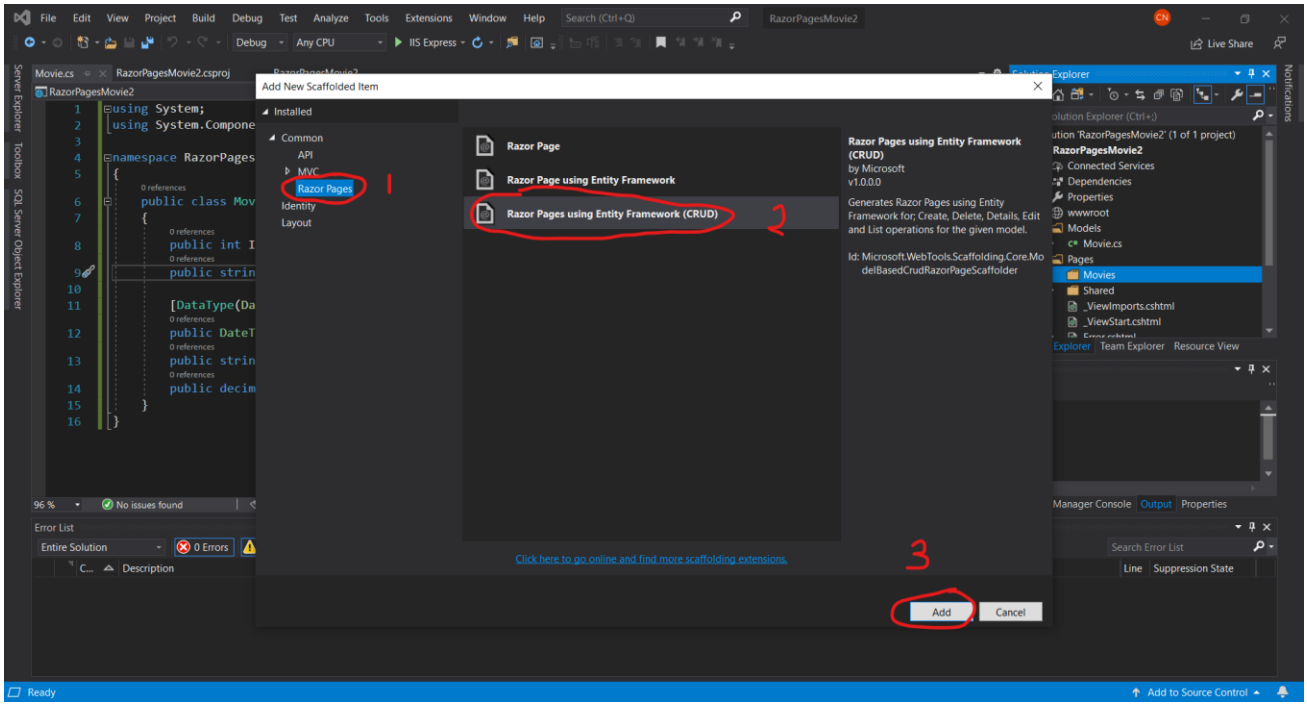




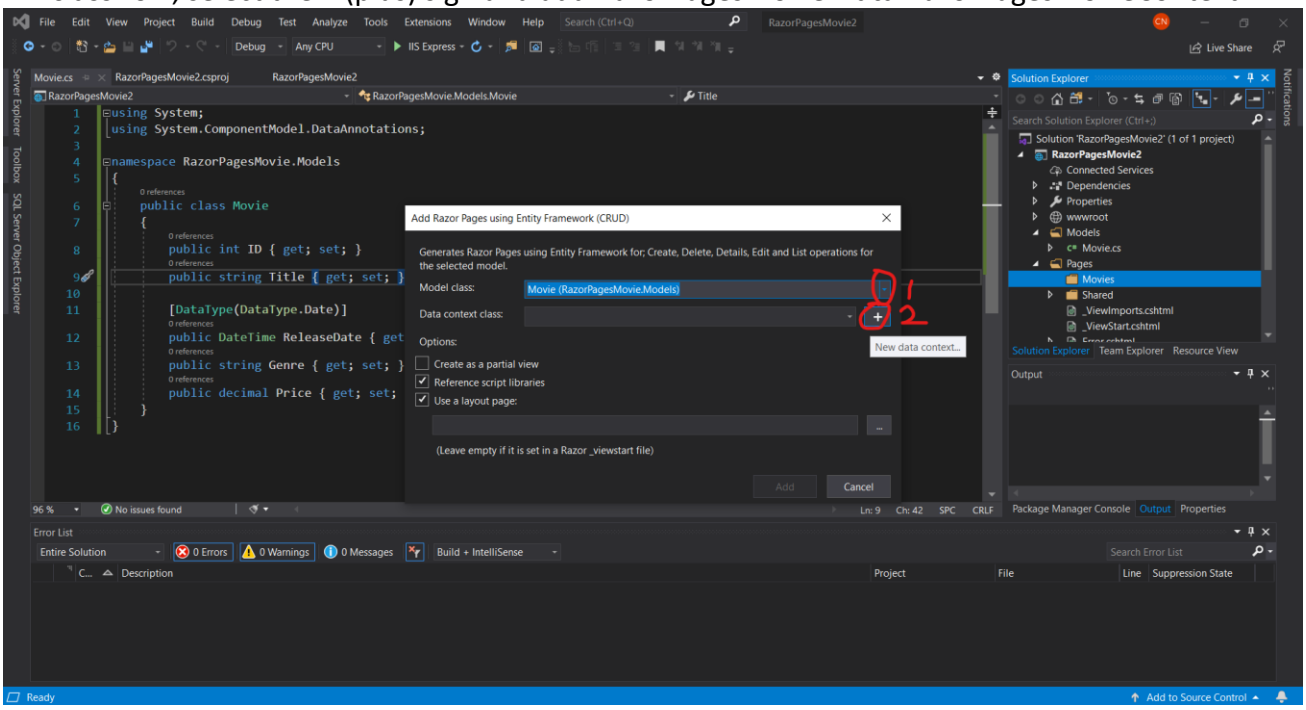
10. Right click on the *Pages/Movies* folder > **Add** > **New Scaffolded Item**.

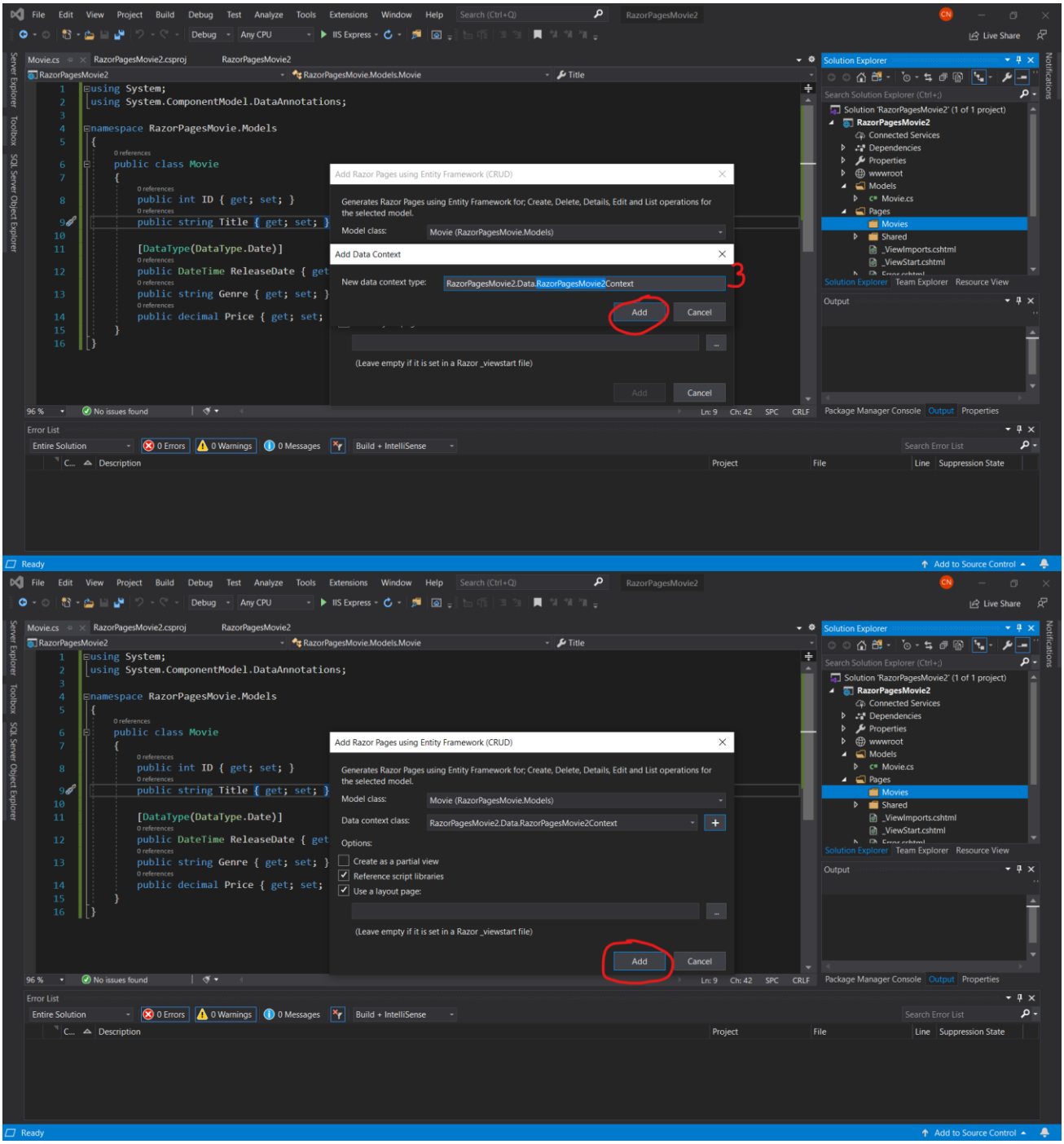


11. In the **Add Scaffold** dialog, select **Razor Pages using Entity Framework (CRUD)** > **Add**.

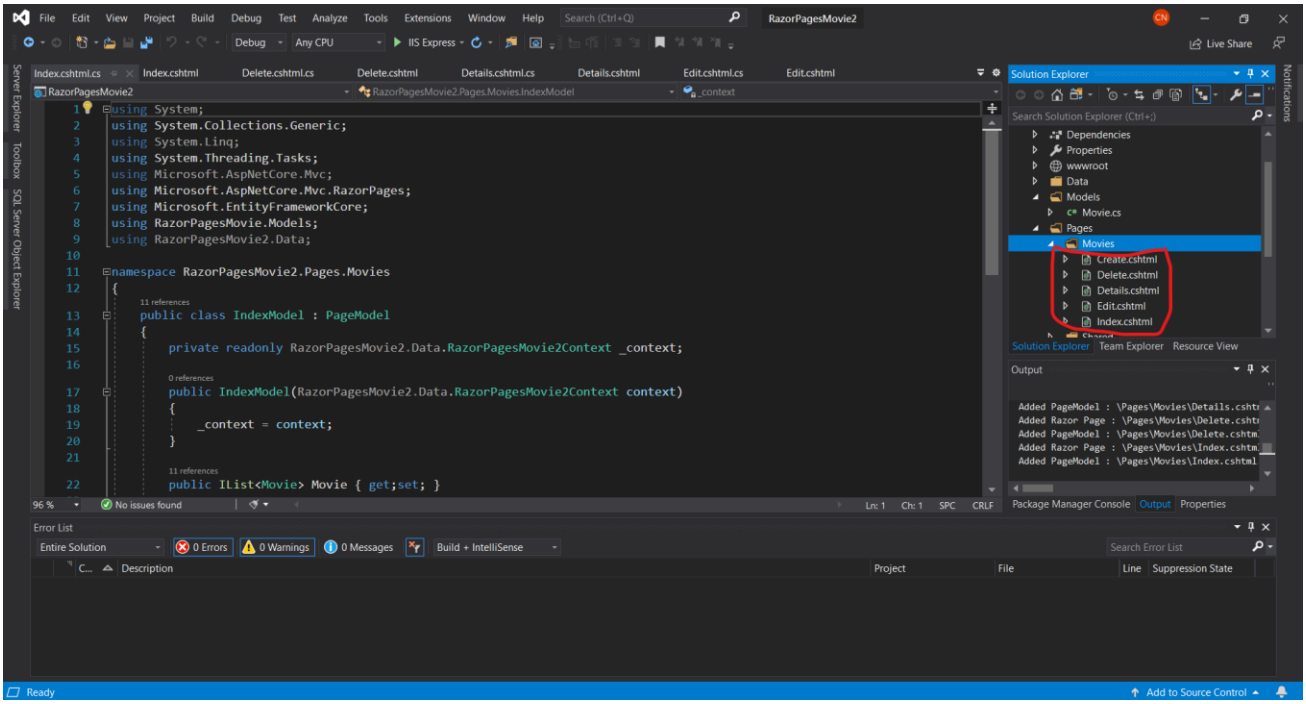


12. In the **Model class** drop down, select **Movie (RazorPagesMovie.Models)**. In the **Data context class** row, select the **+** (plus) sign and add RazorPagesMovie.**Data**.RazorPagesMovieContext.

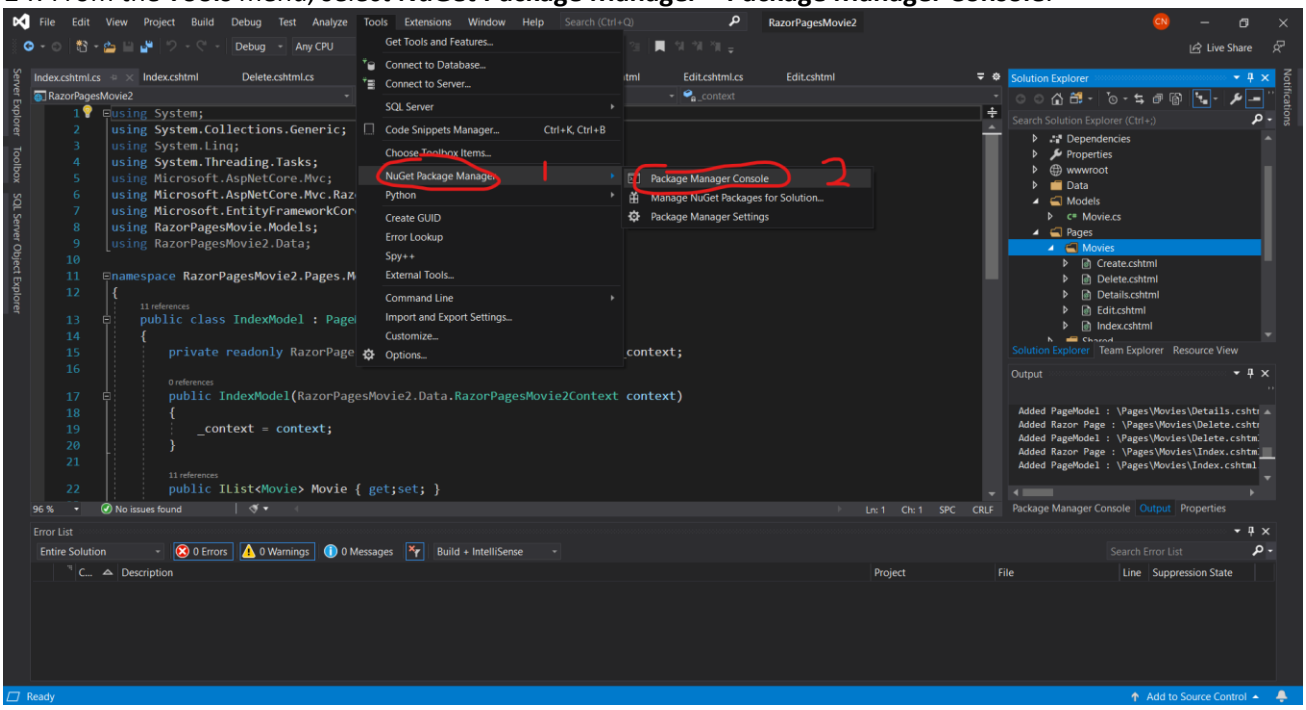




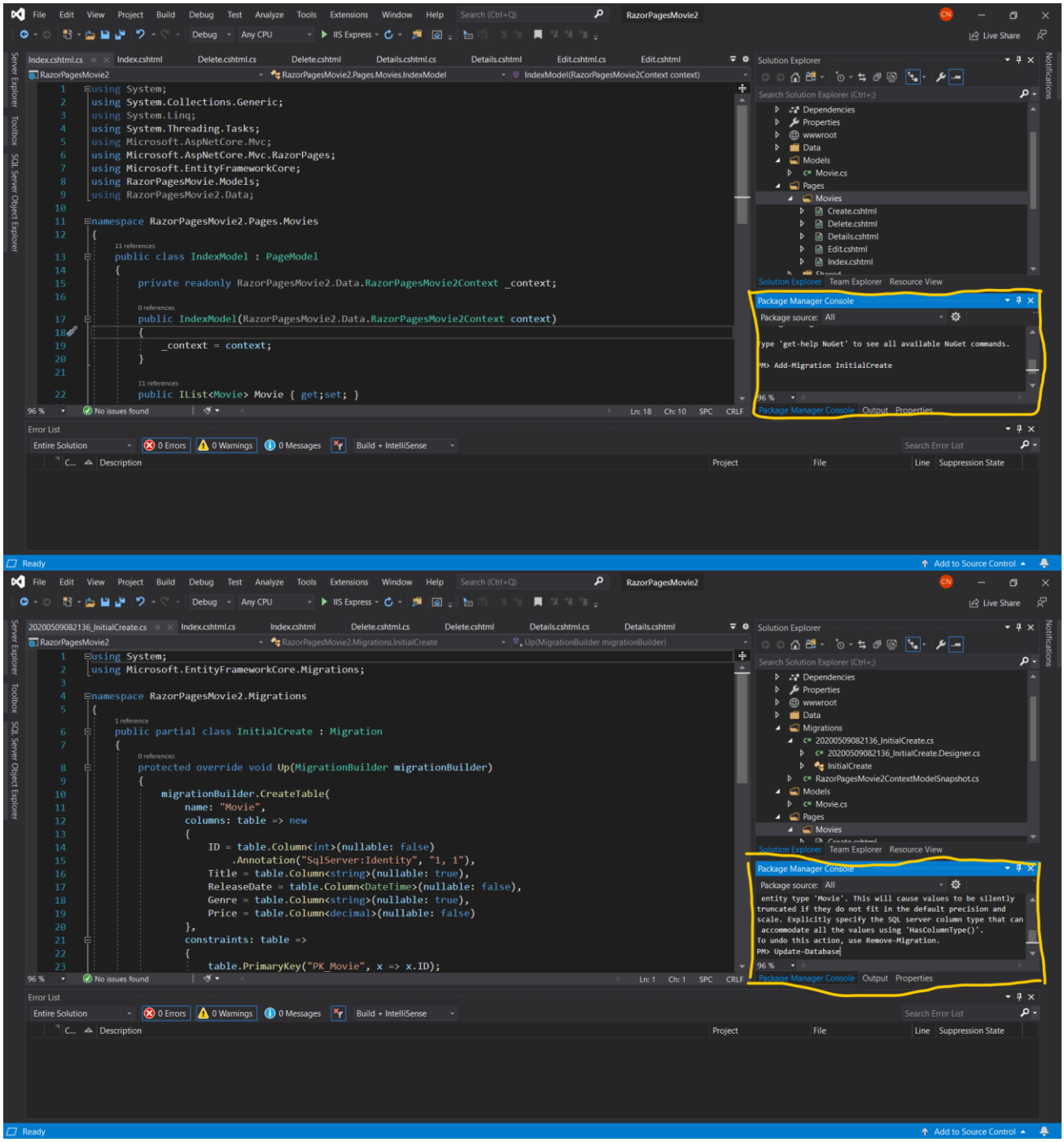
13. Check that the following pages have been created under the *Movies* Folder.



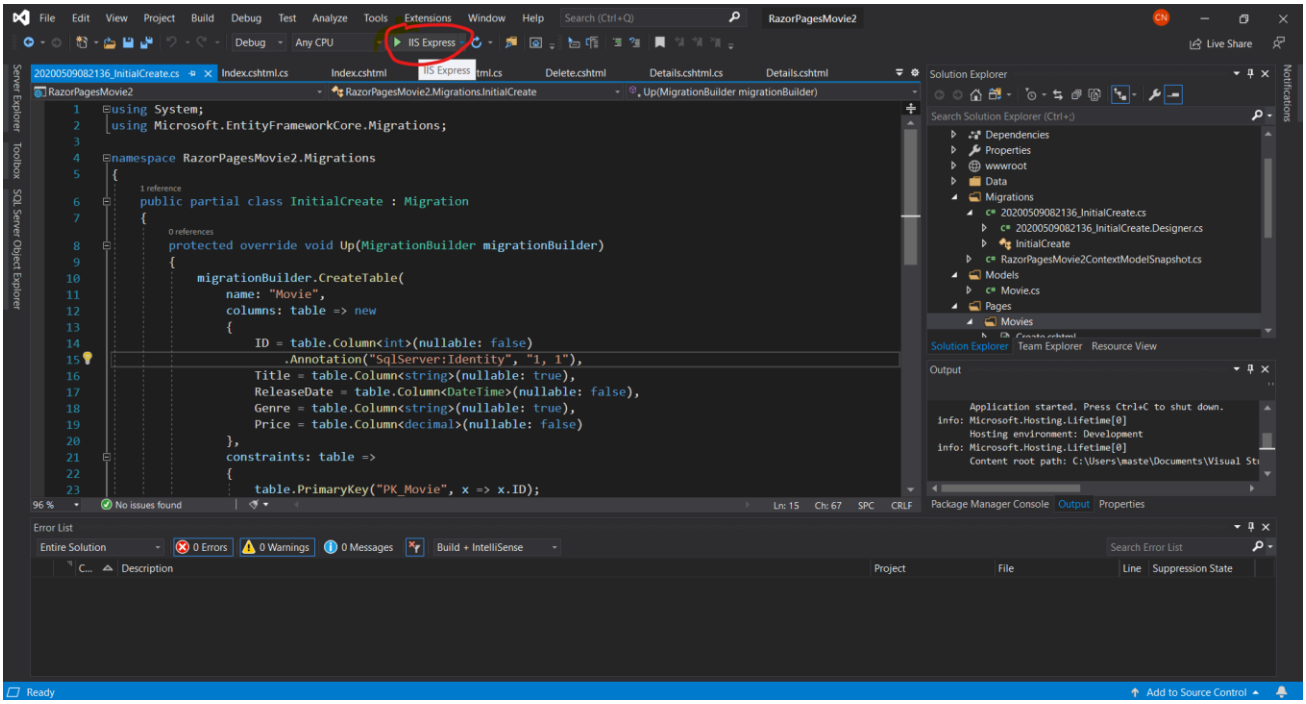
14. From the **Tools** menu, select **NuGet Package Manager** > **Package Manager Console**.

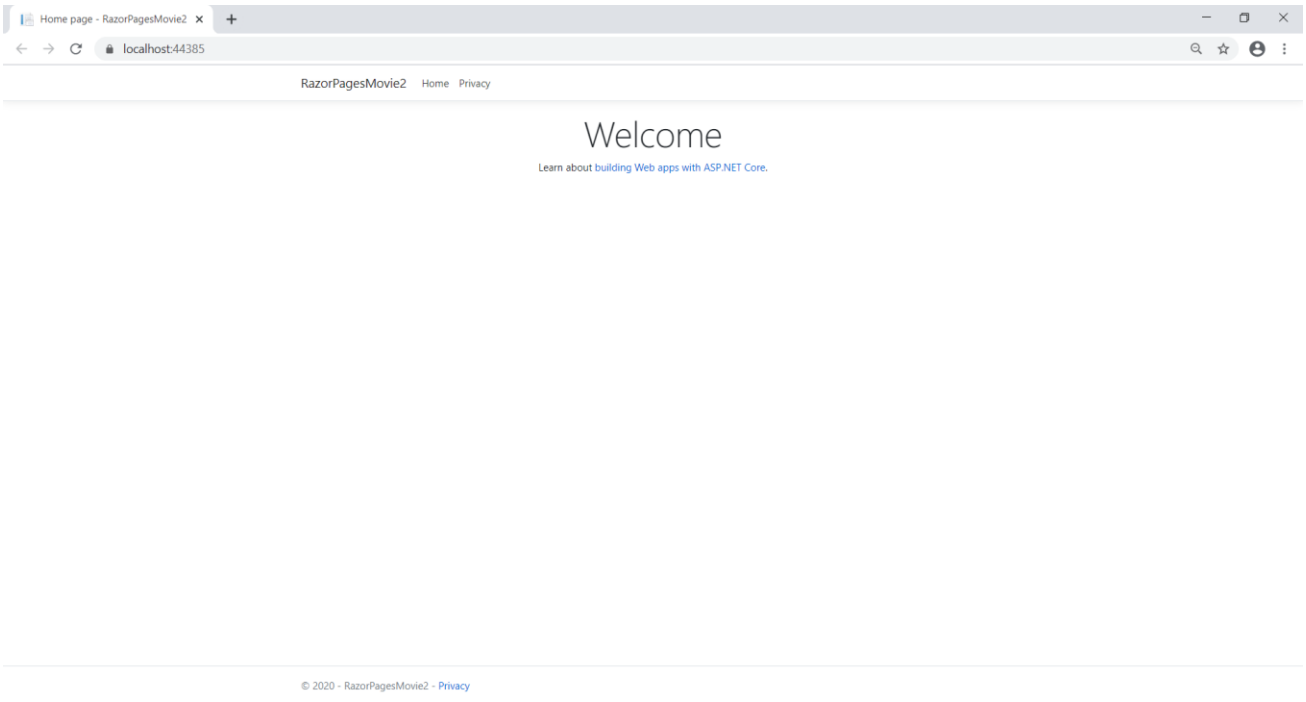


15. In the PMC, enter **Add-Migration InitialCreate**. Then Enter **Update-Database.**


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
RazorPagesMovie2
Debug Any CPU IIS Express
Live Share
Index.cshtml.cs Index.cshtml Delete.cshtml.cs Delete.cshtml Details.cshtml.cs Details.cshtml Edit.cshtml.cs Edit.cshtml
RazorPagesMovie2 RazorPagesMovie2.Pages.Movies.IndexModel IndexModel(RazorPagesMovie2Context context)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.EntityFrameworkCore;
using RazorPagesMovie.Models;
using RazorPagesMovie2.Data;
namespace RazorPagesMovie2.Pages.Movies
{
11 references
public class IndexModel : PageModel
{
private readonly RazorPagesMovie2.Data.RazorPagesMovie2Context _context;
0 references
public IndexModel(RazorPagesMovie2.Data.RazorPagesMovie2Context context)
{
_context = context;
}
11 references
public IList<Movie> Movie { get;set; }
Solution Explorer
Search Solution Explorer (Ctrl+;)
Dependencies
Properties
wwwroot
Data
Models
Movie.cs
Pages
Movies
Create.cshtml
Delete.cshtml
Details.cshtml
Edit.cshtml
Index.cshtml
Solution Explorer Team Explorer Resource View
Package Manager Console
Package source: All
ype 'get-help NuGet' to see all available NuGet commands.
PM> Add-Migration InitialCreate
Package Manager Console Output Properties
96 %
No issues found
Ln: 18 Ch: 10 SPC CRLF
Error List
Entire Solution
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Search Error List
Description
Project
File
Line
Suppression State
Ready
Add to Source Control
File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
RazorPagesMovie2
Debug Any CPU IIS Express
Live Share
20200509082136_InitialCreate.cs Index.cshtml.cs Index.cshtml Delete.cshtml.cs Delete.cshtml Details.cshtml.cs Details.cshtml
RazorPagesMovie2 RazorPagesMovie2.Migrations.InitialCreate Up(MigrationBuilder migrationBuilder)
using System;
using Microsoft.EntityFrameworkCore.Migrations;
namespace RazorPagesMovie2.Migrations
{
1 reference
public partial class InitialCreate : Migration
{
0 references
protected override void Up(MigrationBuilder migrationBuilder)
{
migrationBuilder.CreateTable(
name: "Movie",
columns: table => new
{
ID = table.Column<int>(nullable: false)
.Annotation("SqlServer:Identity", "1, 1"),
Title = table.Column<string>(nullable: true),
ReleaseDate = table.Column<DateTime>(nullable: false),
Genre = table.Column<string>(nullable: true),
Price = table.Column<decimal>(nullable: false)
},
constraints: table =>
{
table.PrimaryKey("PK_Movie", x => x.ID);
Solution Explorer
Search Solution Explorer (Ctrl+;)
Dependencies
Properties
wwwroot
Data
Migrations
20200509082136_InitialCreate.cs
20200509082136_InitialCreate.Designer.cs
InitialCreate
RazorPagesMovie2ContextModelSnapshot.cs
Models
Movie.cs
Pages
Movies
Solution Explorer Team Explorer Resource View
Package Manager Console
Package source: All
entity type 'Movie'. This will cause values to be silently
truncated if they do not fit in the default precision and
scale. Explicitly specify the SQL server column type that can
accommodate all the values using 'HasColumnType()'.
To undo this action, use Remove-Migration.
PM> Update-Database
Package Manager Console Output Properties
96 %
No issues found
Ln: 1 Ch: 1 SPC CRLF
Error List
Entire Solution
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Search Error List
Description
Project
File
Line
Suppression State
Ready
Add to Source Control

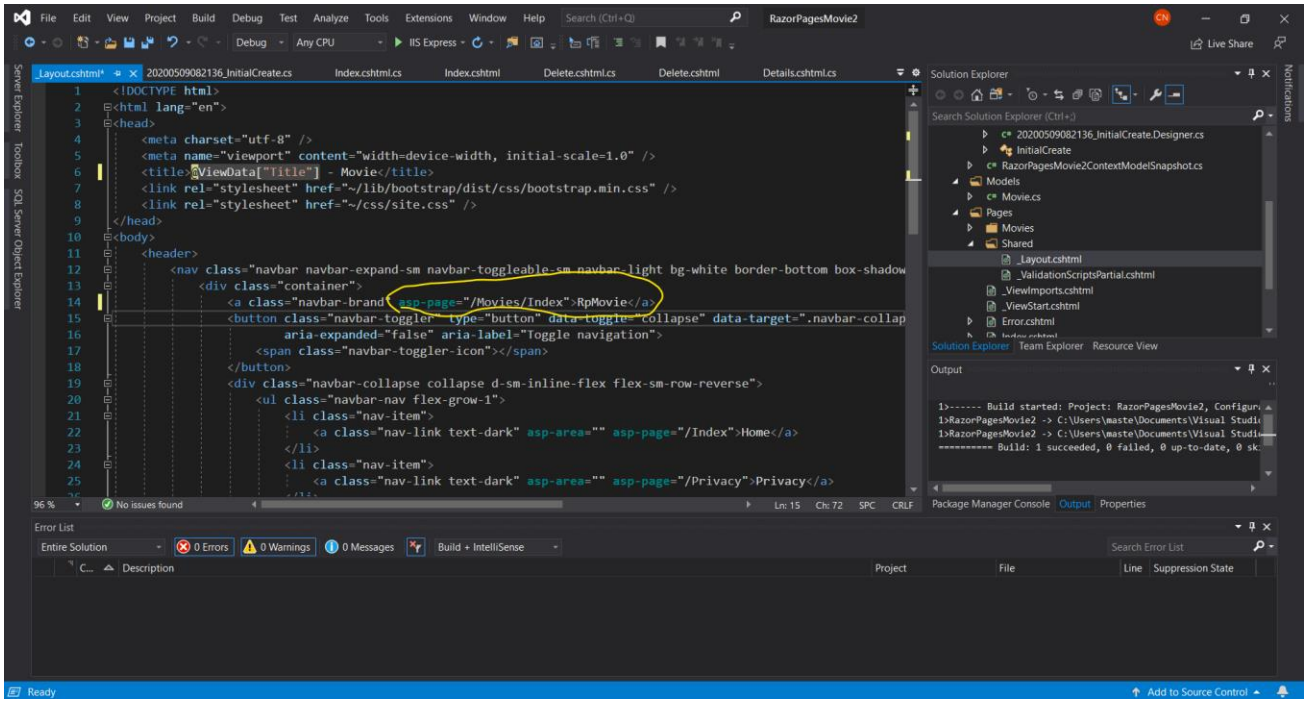
16. **Run** the app by using the Green IIS Express Button or using the shortcut Ctrl + F5. You should be redirected to this page in your default web browser specified in Visual Studio.



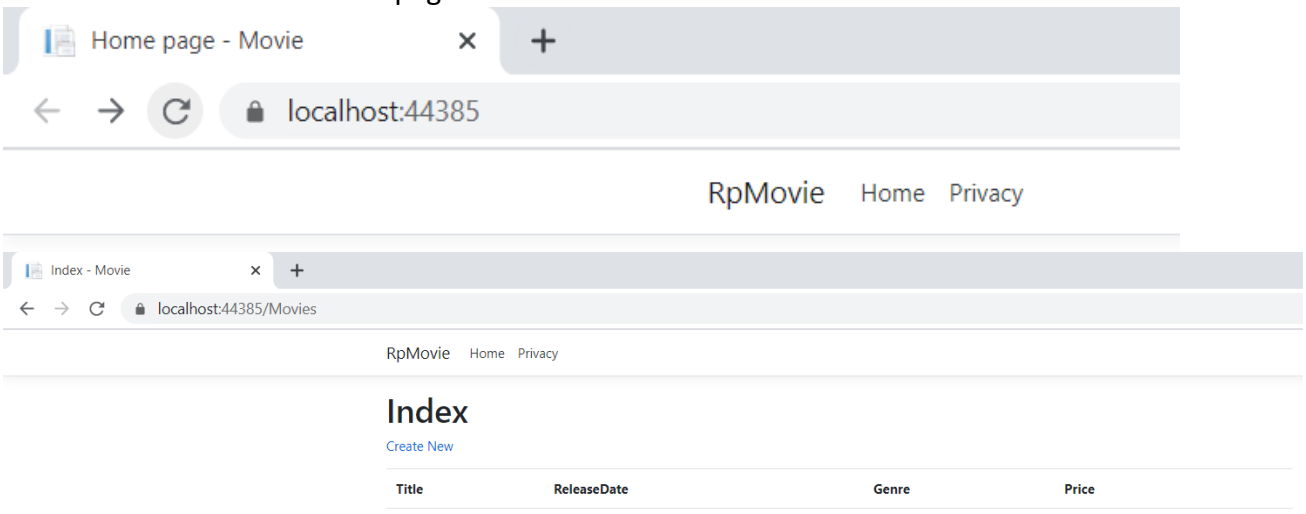


Section 3: Scaffolded Razor Pages

17. Change the <title> element in the *Pages/Shared/_Layout.cshtml* file to display **RpMovie** rather than **RazorPagesMovie** and the <a> element so it links to the movie page and the header is "RpMovie".

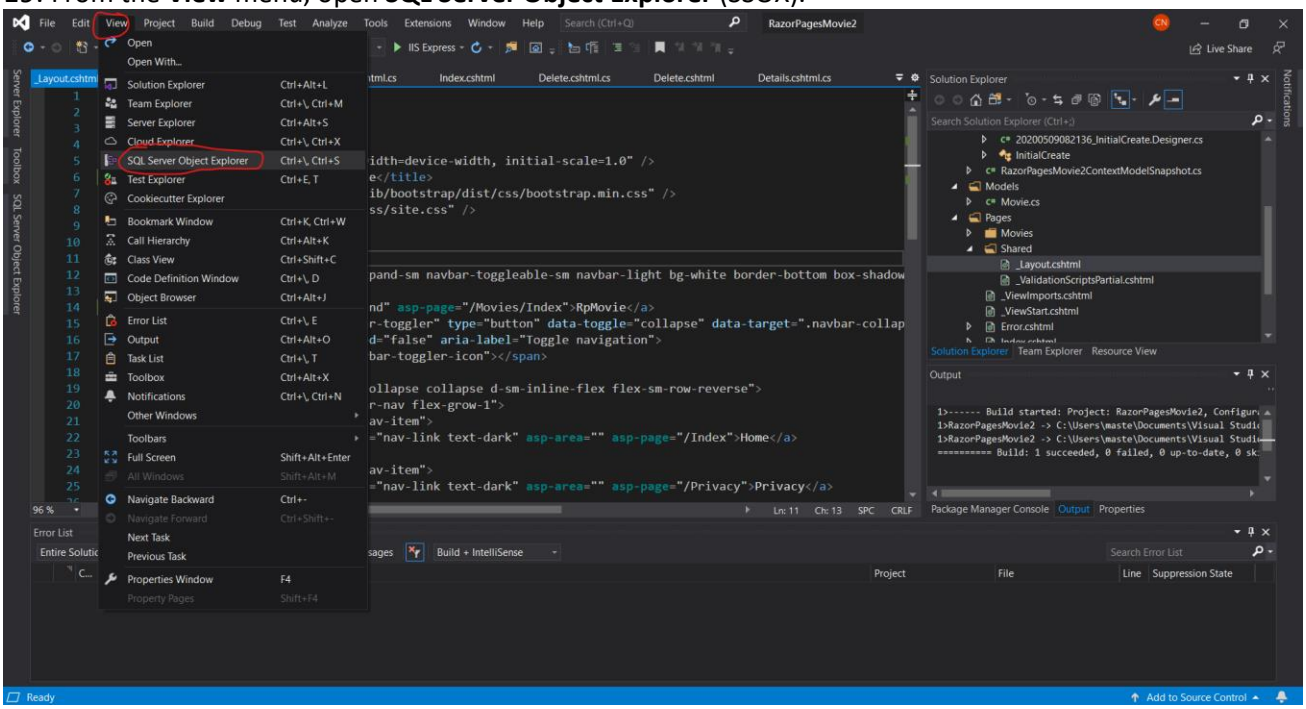


18. Refresh the page in the browser. The header should display "RpMovie" and when it is clicked, should lead to the *Movies* page.

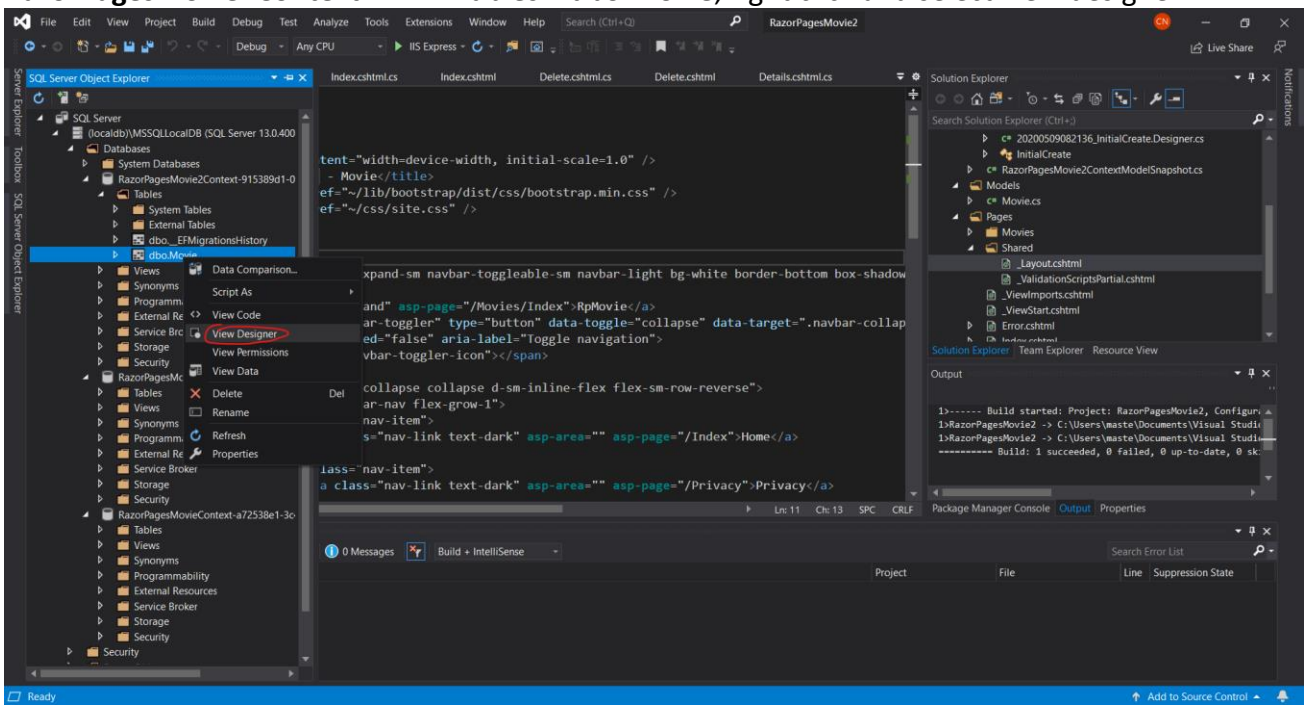


Section 4: Work with a database

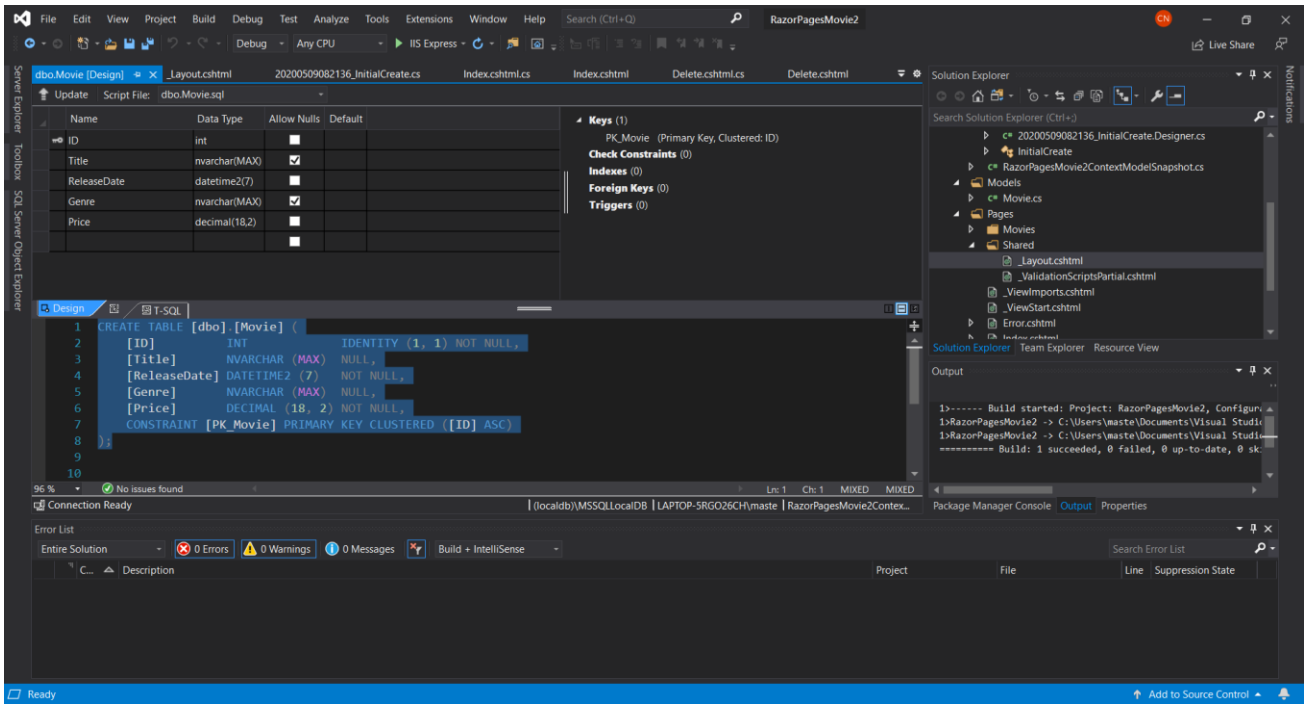
19. From the **View** menu, open **SQL Server Object Explorer** (SSOX).



20. Expand the “Tree” structure: **localdb\MSSQLLocalDB > Databases > RazorPagesMovie2Context-XXX > Tables > dbo.Movie**, right click and select view designer.

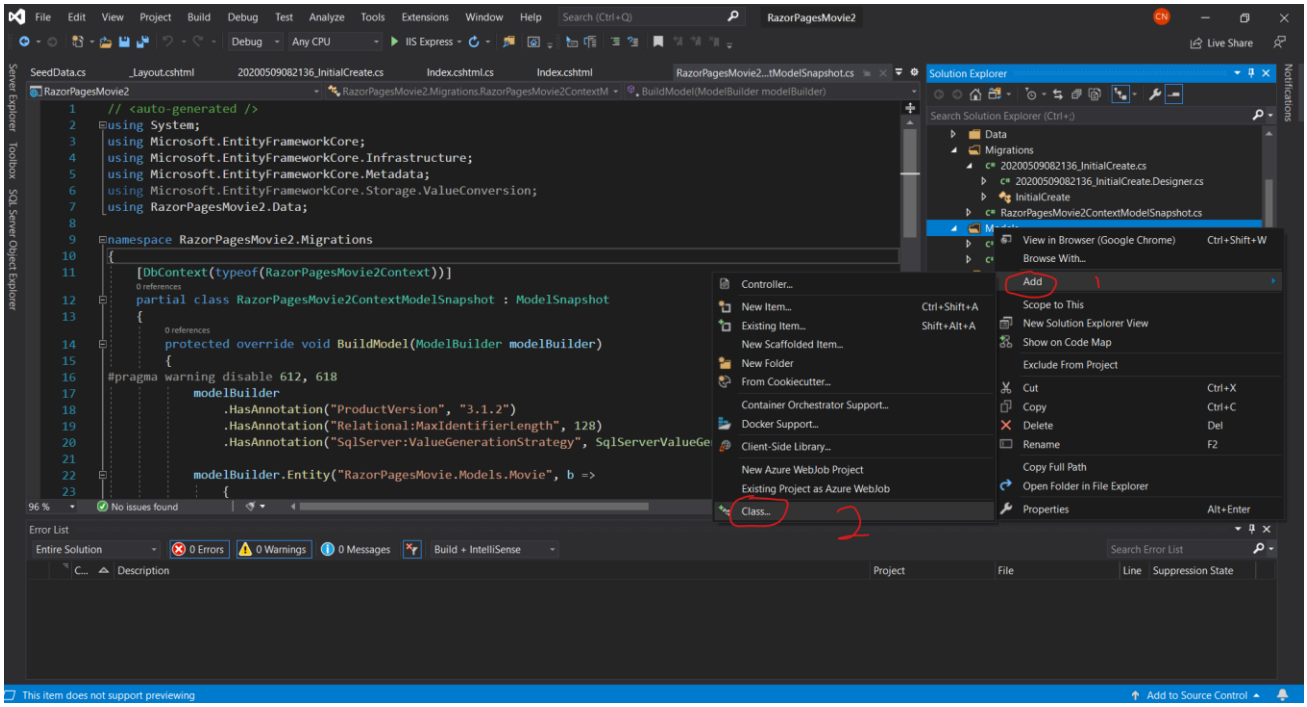


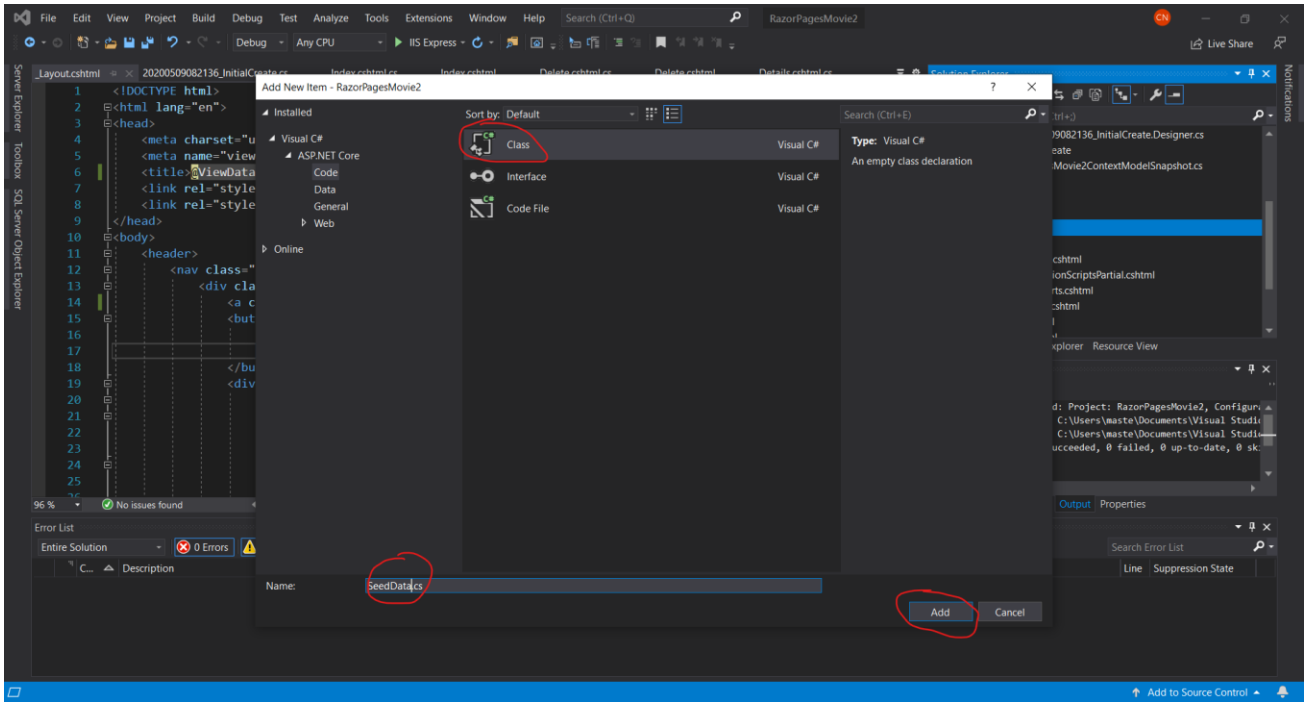
21. The predefined columns will appear here and you can set whether the value can or cannot be NULL.



22. Create a new class named `SeedData` in the *Models* folder with the following code.

Important Note: I had to change using RazorPagesMovie.Data to RazorPagesMovie2.Data due to the difference in naming conventions of the project. Similarly, RazorPagesMovieContext has to be changed to RazorPagesMovie2Context to prevent any errors that may occur.






```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using RazorPagesMovie.Data;
using System;
using System.Linq;

namespace RazorPagesMovie.Models
{
    public static class SeedData
    {
        public static void Initialize(IServiceProvider serviceProvider)
        {
            using (var context = new RazorPagesMovieContext(
                serviceProvider.GetRequiredService<
                    DbContextOptions<RazorPagesMovieContext>>()))
            {
                // Look for any movies.
                if (context.Movie.Any())
                {
                    return;   // DB has been seeded
                }

                context.Movie.AddRange(
                    new Movie
                    {
                        Title = "When Harry Met Sally",
                        ReleaseDate = DateTime.Parse("1989-2-12"),
                        Genre = "Romantic Comedy",
                        Price = 7.99M
                    },

                    new Movie
                    {
                        Title = "Ghostbusters ",
                        ReleaseDate = DateTime.Parse("1984-3-13"),
                        Genre = "Comedy",
                        Price = 8.99M
```

```
                    },

                    new Movie
                    {
                        Title = "Ghostbusters 2",
                        ReleaseDate = DateTime.Parse("1986-2-23"),
                        Genre = "Comedy",
                        Price = 9.99M
                    },

                    new Movie
                    {
                        Title = "Rio Bravo",
                        ReleaseDate = DateTime.Parse("1959-4-15"),
                        Genre = "Western",
                        Price = 3.99M
                    }
                );
                context.SaveChanges();
            }
        }
    }
}
```

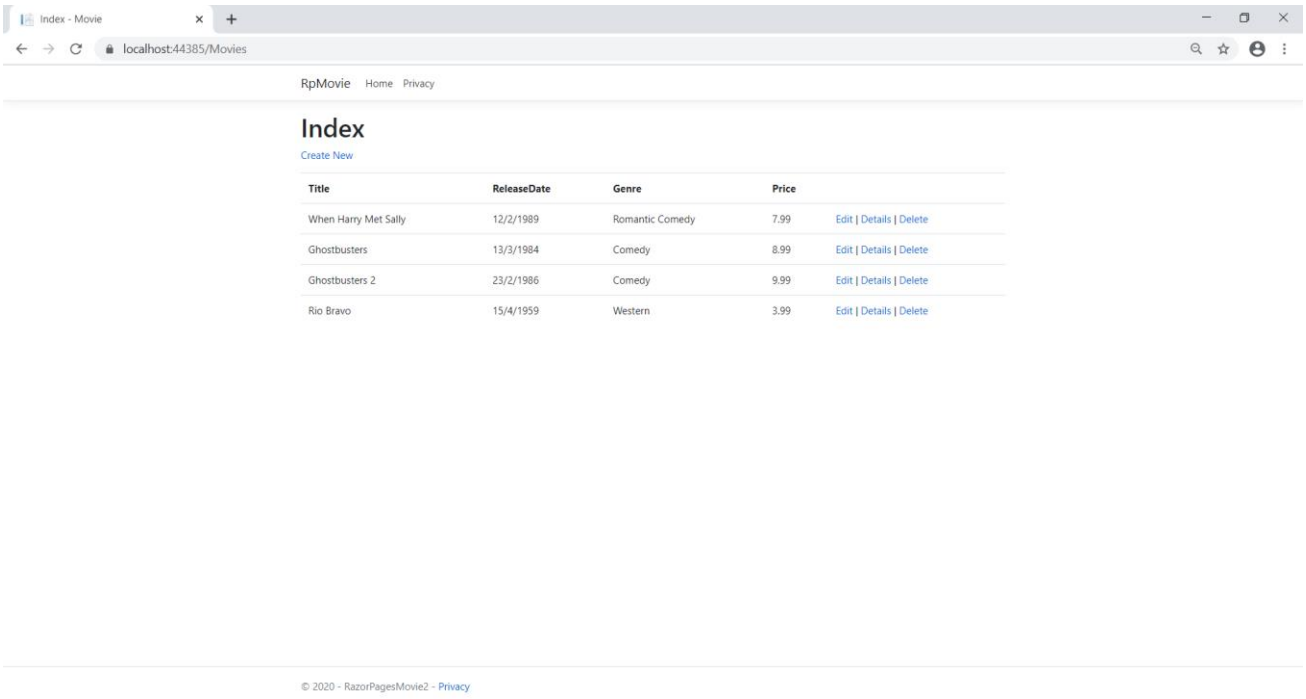
23. In Program.cs, delete all the content and replace it with this.

Important Note: I had to add <u>using RazorPagesMovie2;</u> as an assembly reference.

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using RazorPagesMovie.Models;
using RazorPagesMovie2; //add only if using different naming convention
using System;

namespace RazorPagesMovie
{
    public class Program
    {
        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>();
                });
    }
}
```

24. **Run** the app again. Under the Movies Pages, it should be populated with the boiler plate data.

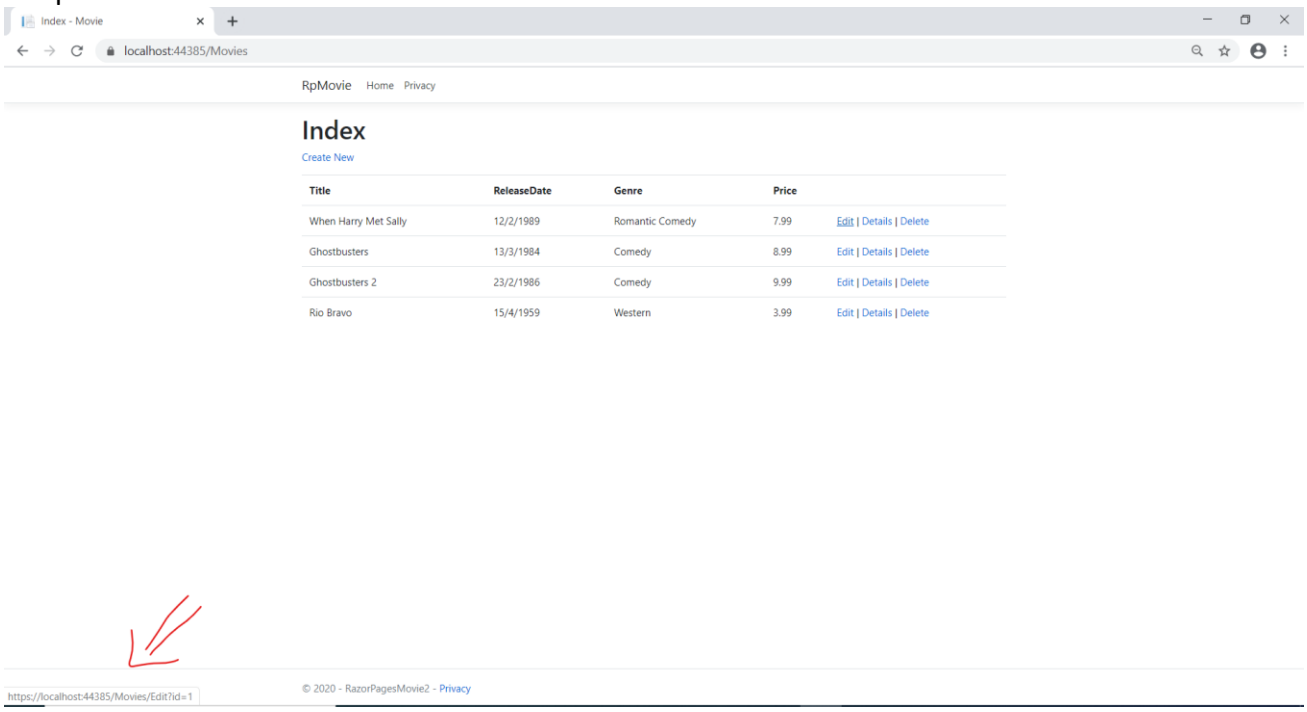


Section 5: Update the Pages

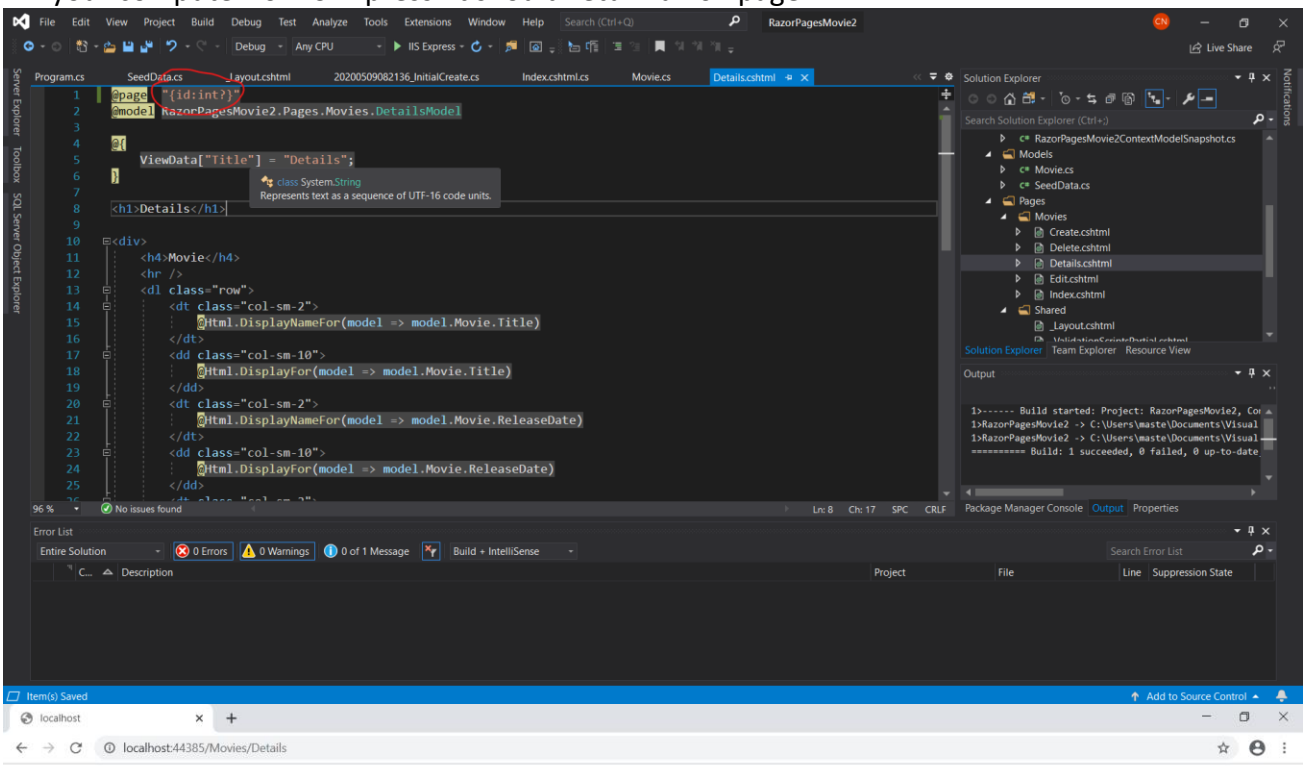
25. Open the *Models/Movie.cs* file and add the highlighted lines shown in the following code:

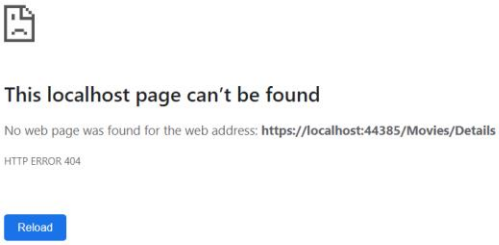
```C#
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace RazorPagesMovie.Models
{
    public class Movie
    {
        public int ID { get; set; }
        public string Title { get; set; }

        [Display(Name = "Release Date")]
        [DataType(DataType.Date)]
        public DateTime ReleaseDate { get; set; }
        public string Genre { get; set; }

        [Column(TypeName = "decimal(18, 2)")]
        public decimal Price { get; set; }
    }
}
```

26. Refresh your browser. When you hover over the edit/details/delete links, you should see a unique link for each movie with an ID associated with it.



27. Add “{id:int?}” to the details page behind @page. Refresh your browser. Navigate to https://localhost:XXXX/Movies/Details/ where XXXX is the designated port number by your computer for IIS Express. It should return a 404 page.





Section 6: Add Search

28. Add the following highlighted properties to *Pages/Movies/Index.cshtml.cs*:

Important Note: Be sure to include using Microsoft.AspNetCore.Mvc.Rendering;

C# Copy

```
public class IndexModel : PageModel
{
    private readonly RazorPagesMovie.Data.RazorPagesMovieContext _context;

    public IndexModel(RazorPagesMovie.Data.RazorPagesMovieContext context)
    {
        _context = context;
    }

    public IList<Movie> Movie { get; set; }
    [BindProperty(SupportsGet = true)]
    public string SearchString { get; set; }
    // Requires using Microsoft.AspNetCore.Mvc.Rendering;
    public SelectList Genres { get; set; }
    [BindProperty(SupportsGet = true)]
    public string MovieGenre { get; set; }
```

29. Update the Index page's `OnGetAsync` method with the following code:

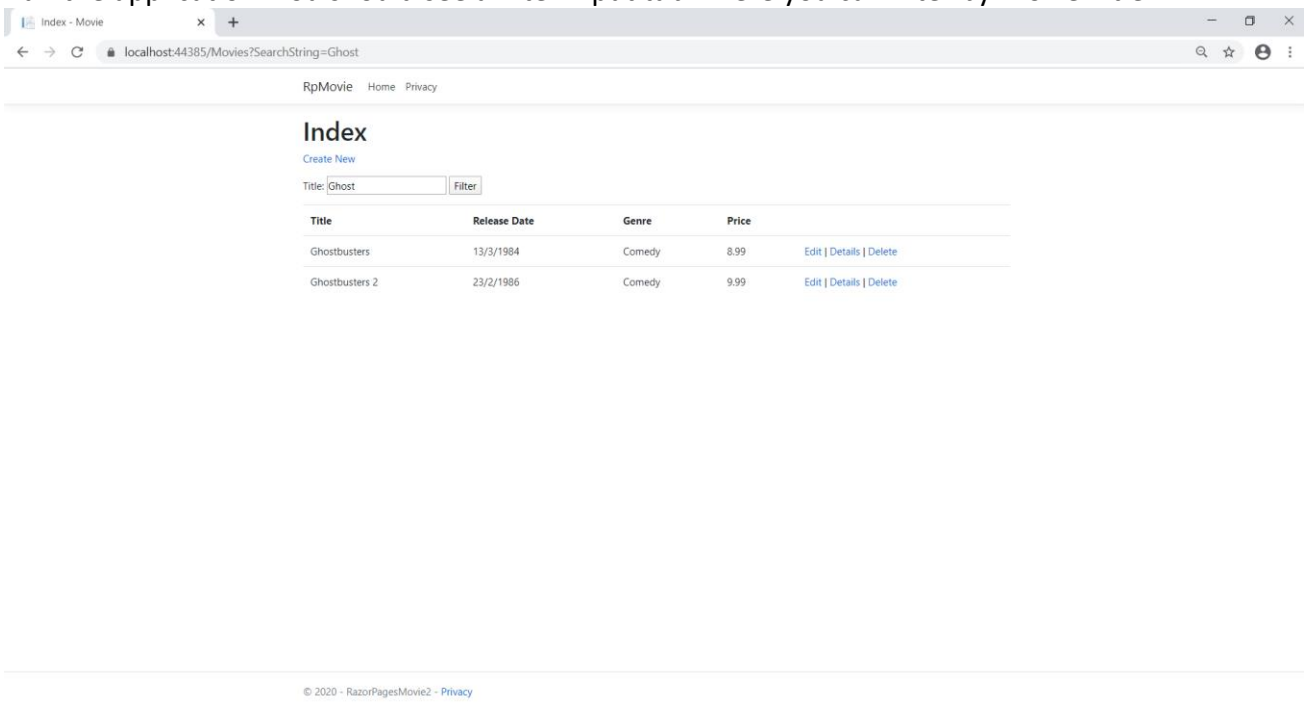
```
public async Task OnGetAsync()
        {
            var movies = from m in _context.Movie
                         select m;
            if (!string.IsNullOrEmpty(SearchString))
            {
               movies = movies.Where(s => s.Title.Contains(SearchString));
            }

            Movie = await movies.ToListAsync();
}
```

30. Open the *Pages/Movies/Index.cshtml* file, and add the `<form>` markup highlighted in the following code:

CSHTML Copy

```
@page
@model RazorPagesMovie.Pages.Movies.IndexModel

@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-page="Create">Create New</a>
</p>

<form>
    <p>
        Title: <input type="text" asp-for="SearchString" />
        <input type="submit" value="Filter" />
    </p>
</form>

<table class="table">
    @*Markup removed for brevity.*@
```

31. Run the application. You should see a filter input tab where you can filter by Movie Title.



32. Update the `OnGetAsync` method with the following code:

```
public async Task OnGetAsync()
{
        // Use LINQ to get list of genres.
        IQueryable<string> genreQuery = from m in _context.Movie
                                        orderby m.Genre
                                        select m.Genre;
```

```
            var movies = from m in _context.Movie
                         select m;

            if (!string.IsNullOrEmpty(SearchString))
            {
               movies = movies.Where(s => s.Title.Contains(SearchString));
            }

            if (!string.IsNullOrEmpty(MovieGenre))
            {
               movies = movies.Where(x => x.Genre == MovieGenre);
            }
            Genres = new SelectList(await genreQuery.Distinct().ToListAsync());
            Movie = await movies.ToListAsync();
}
```

33. Update *Index.cshtml* as follows to allow filtering by genres:

CSHTML Copy

```
@page
@model RazorPagesMovie.Pages.Movies.IndexModel

@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-page="Create">Create New</a>
</p>

<form>
    <p>
        <select asp-for="MovieGenre" asp-items="Model.Genres">
            <option value="">All</option>
        </select>
        Title: <input type="text" asp-for="SearchString" />
        <input type="submit" value="Filter" />
    </p>
</form>

<table class="table">
    @*Markup removed for brevity.*@
```

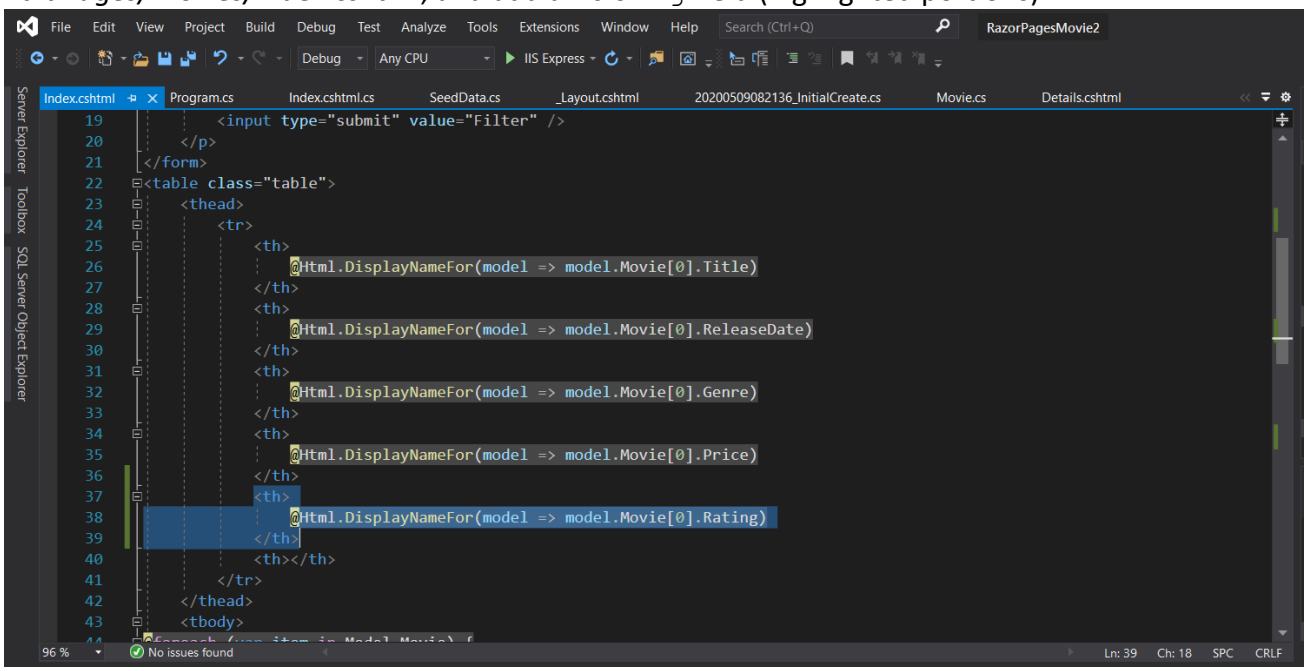
Section 7: Adding a new field

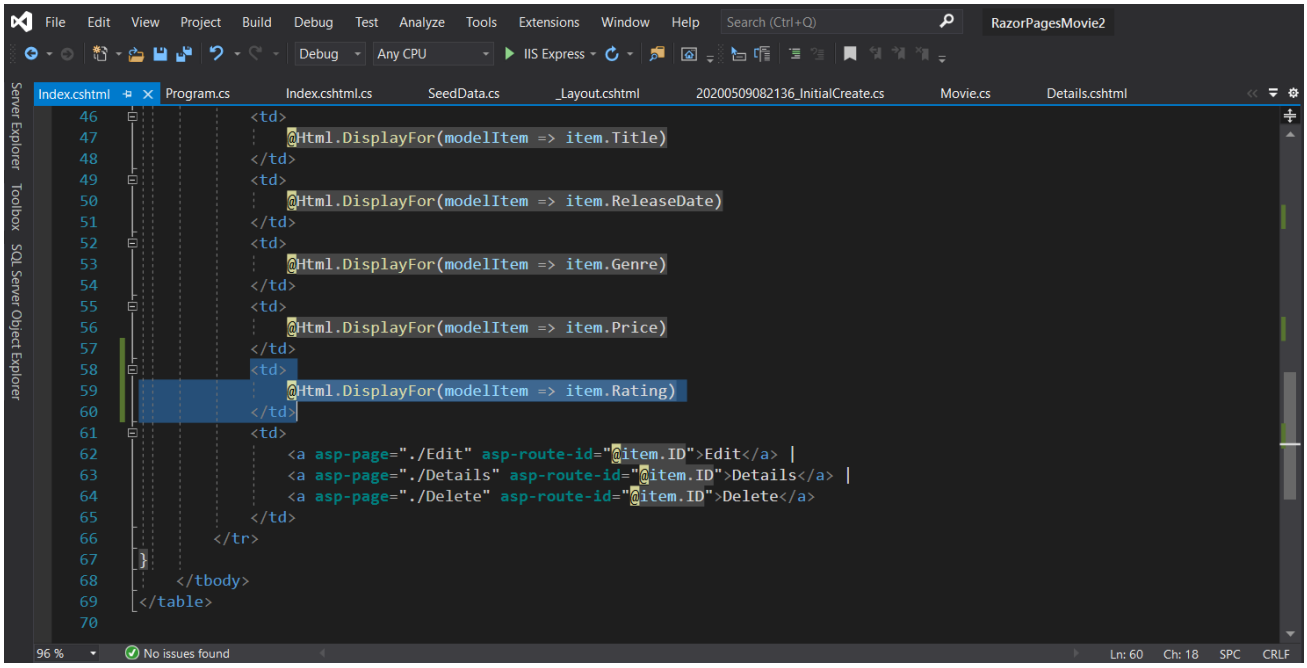
34. Open the *Models/Movie.cs* file and add a `Rating` property:

C# Copy

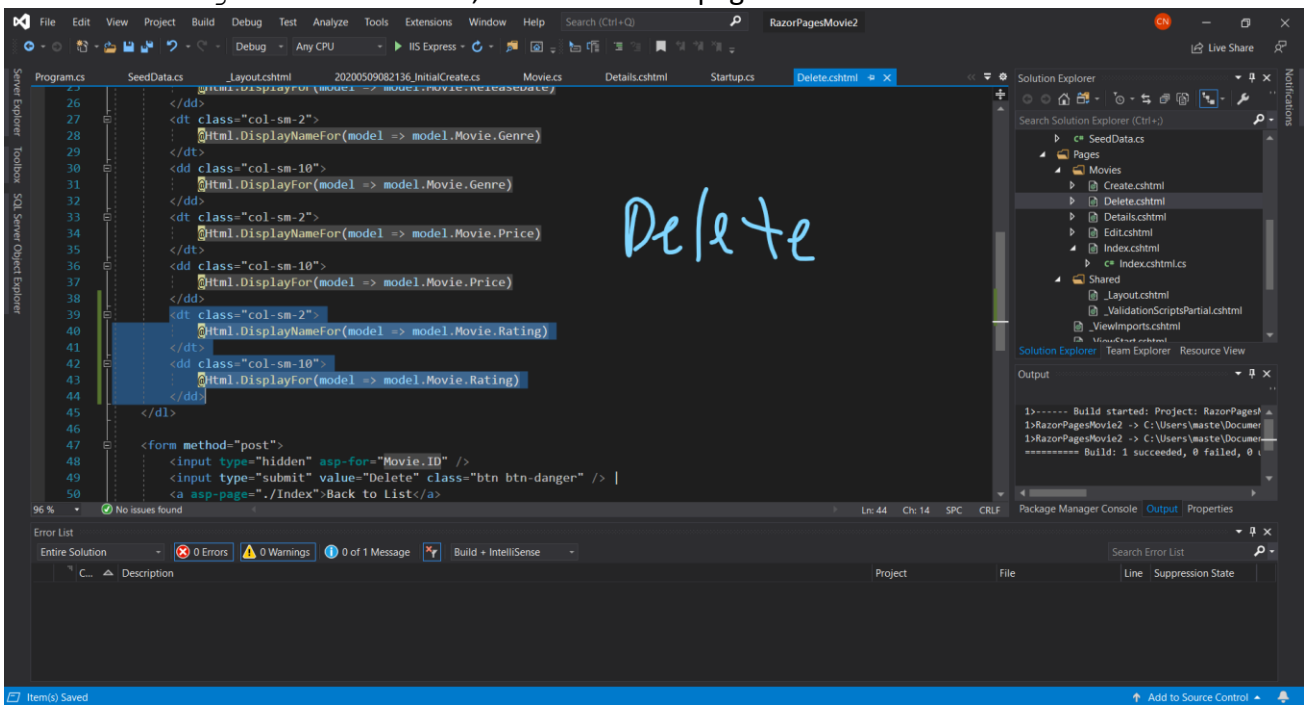
```
public class Movie
{
    public int ID { get; set; }
    public string Title { get; set; }

    [Display(Name = "Release Date")]
    [DataType(DataType.Date)]
    public DateTime ReleaseDate { get; set; }
    public string Genre { get; set; }

    [Column(TypeName = "decimal(18, 2)")]
    public decimal Price { get; set; }
    public string Rating { get; set; }
}
```

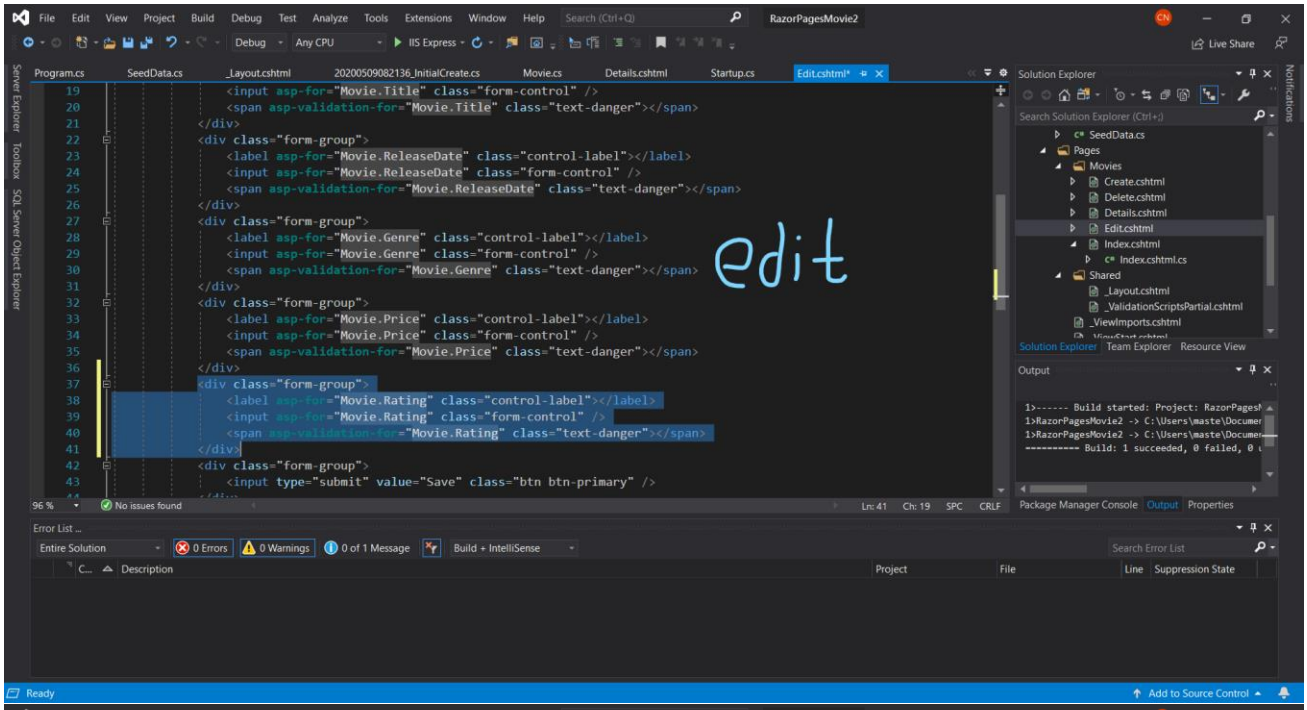
35. Edit *Pages/Movies/Index.cshtml*, and add a `Rating` field (highlighted portions):

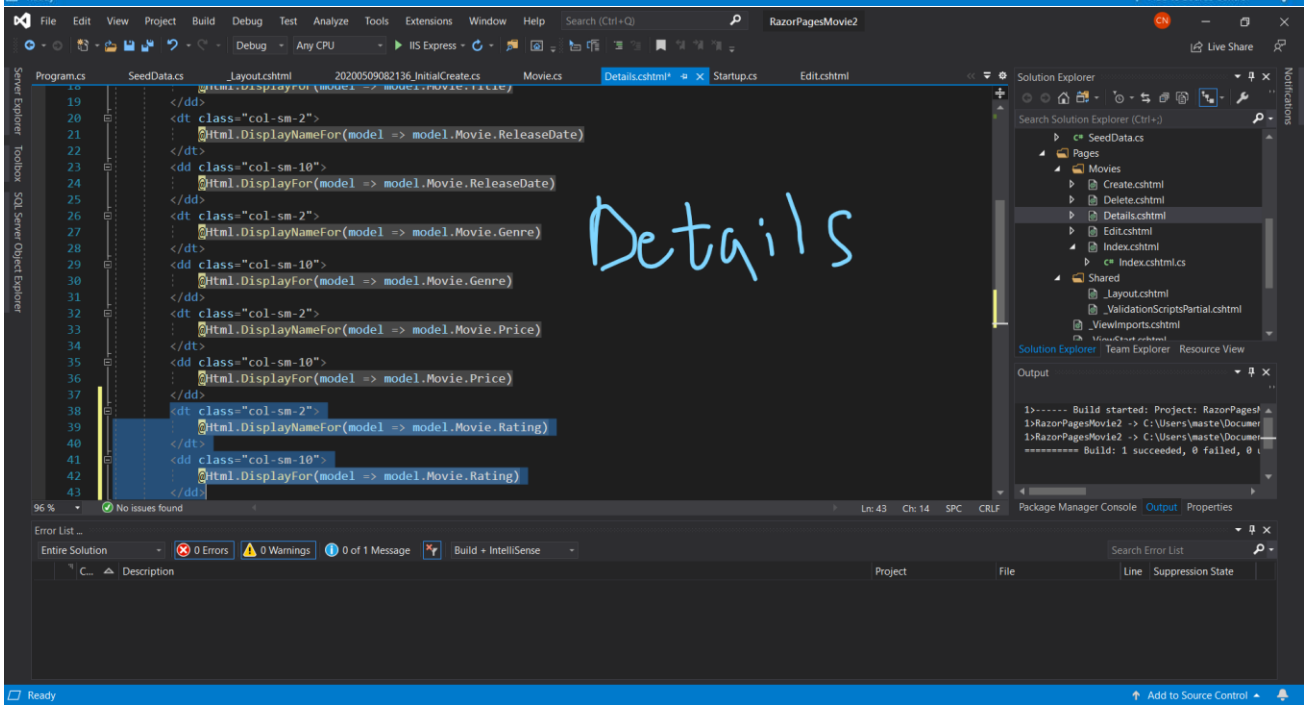


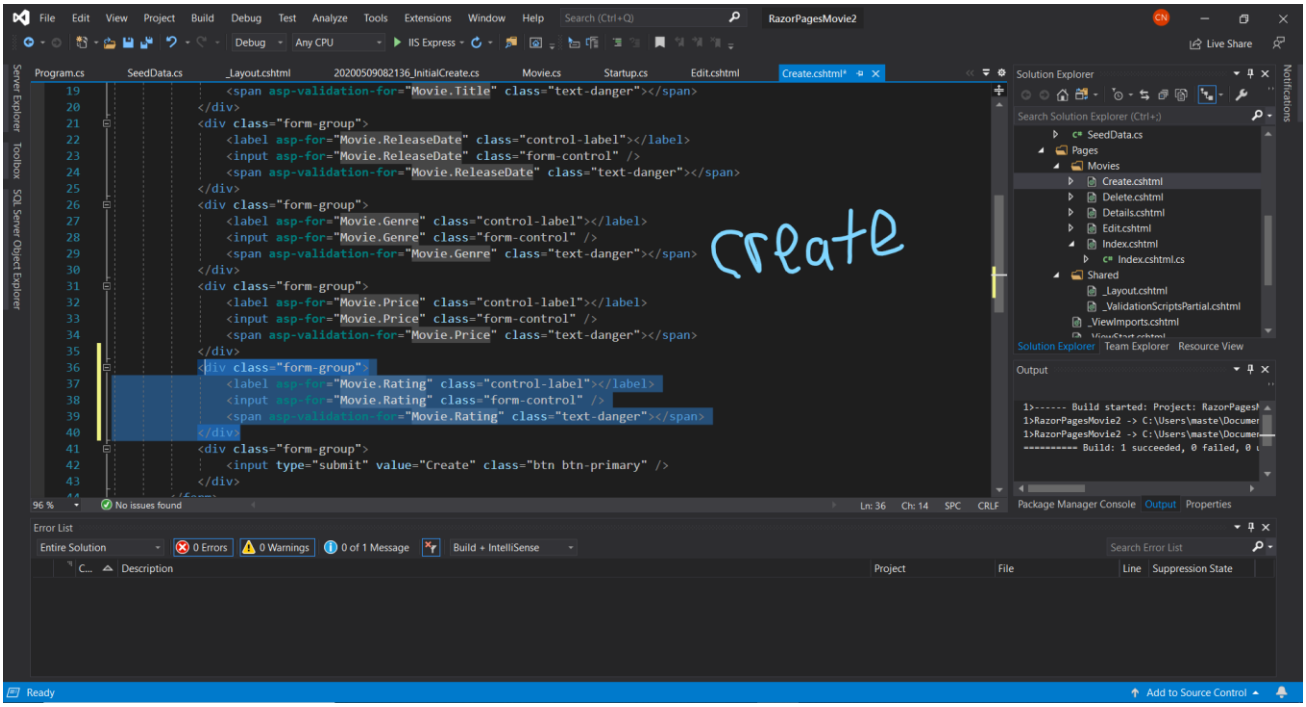


36. Add the `Rating` field to the Delete, Edit and Details pages.

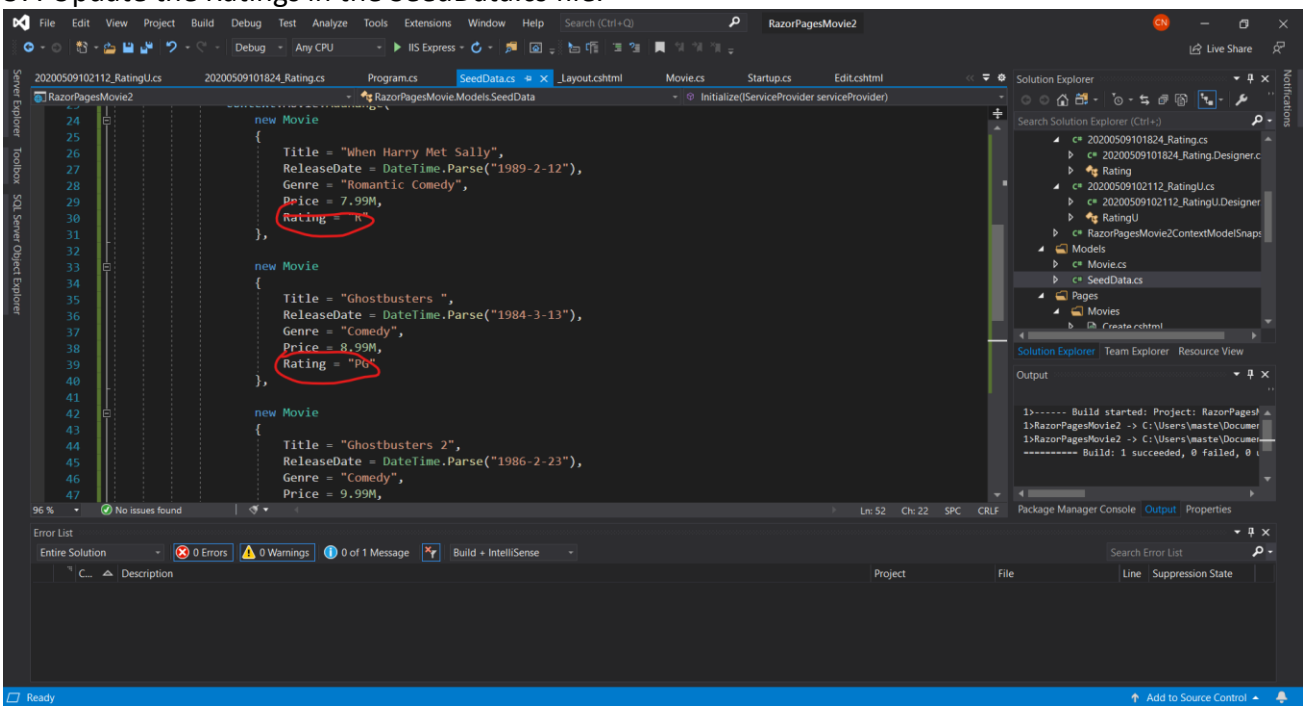



edit


Details



37. Update the Ratings in the SeedData.cs file.



38. From the **Tools** menu, select **NuGet Package Manager > Package Manager Console**. In the PMC, enter the following commands:

PowerShell Copy

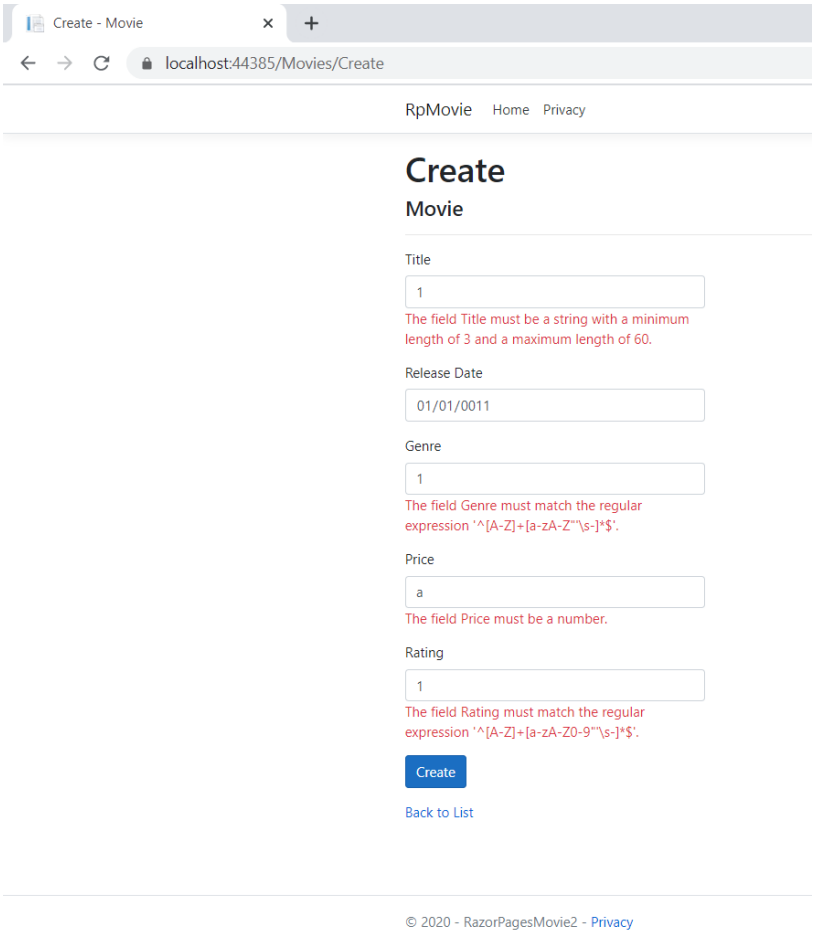
```
Add-Migration Rating
Update-Database
```

Section 8: Add Validation

39. Update the `Movie` class to take advantage of the built-in `Required`, `StringLength`, `RegularExpression`, and `Range` validation attributes.

```
public class Movie
{
    public int ID { get; set; }

    [StringLength(60, MinimumLength = 3)]
    [Required]
    public string Title { get; set; }

    [Display(Name = "Release Date")]
    [DataType(DataType.Date)]
    public DateTime ReleaseDate { get; set; }

    [Range(1, 100)]
    [DataType(DataType.Currency)]
    [Column(TypeName = "decimal(18, 2)")]
    public decimal Price { get; set; }

    [RegularExpression(@"^[A-Z]+[a-zA-Z""'\s-]*$")]
    [Required]
    [StringLength(30)]
    public string Genre { get; set; }

    [RegularExpression(@"^[A-Z]+[a-zA-Z0-9""'\s-]*$")]
    [StringLength(5)]
    [Required]
    public string Rating { get; set; }
}
```

40. **Run** the app in the browser. On the create movie page, notice the input validation.


Create - Movie
localhost:44385/Movies/Create
RpMovie
Home
Privacy
Create
Movie
Title
1
The field Title must be a string with a minimum length of 3 and a maximum length of 60.
Release Date
01/01/0011
Genre
1
The field Genre must match the regular expression '^[A-Z]+[a-zA-Z''\s-]*$'.
Price
a
The field Price must be a number.
Rating
1
The field Rating must match the regular expression '^[A-Z]+[a-zA-Z0-9''\s-]*$'.
Create
Back to List
© 2020 - RazorPagesMovie2 - Privacy

~ THE END ~