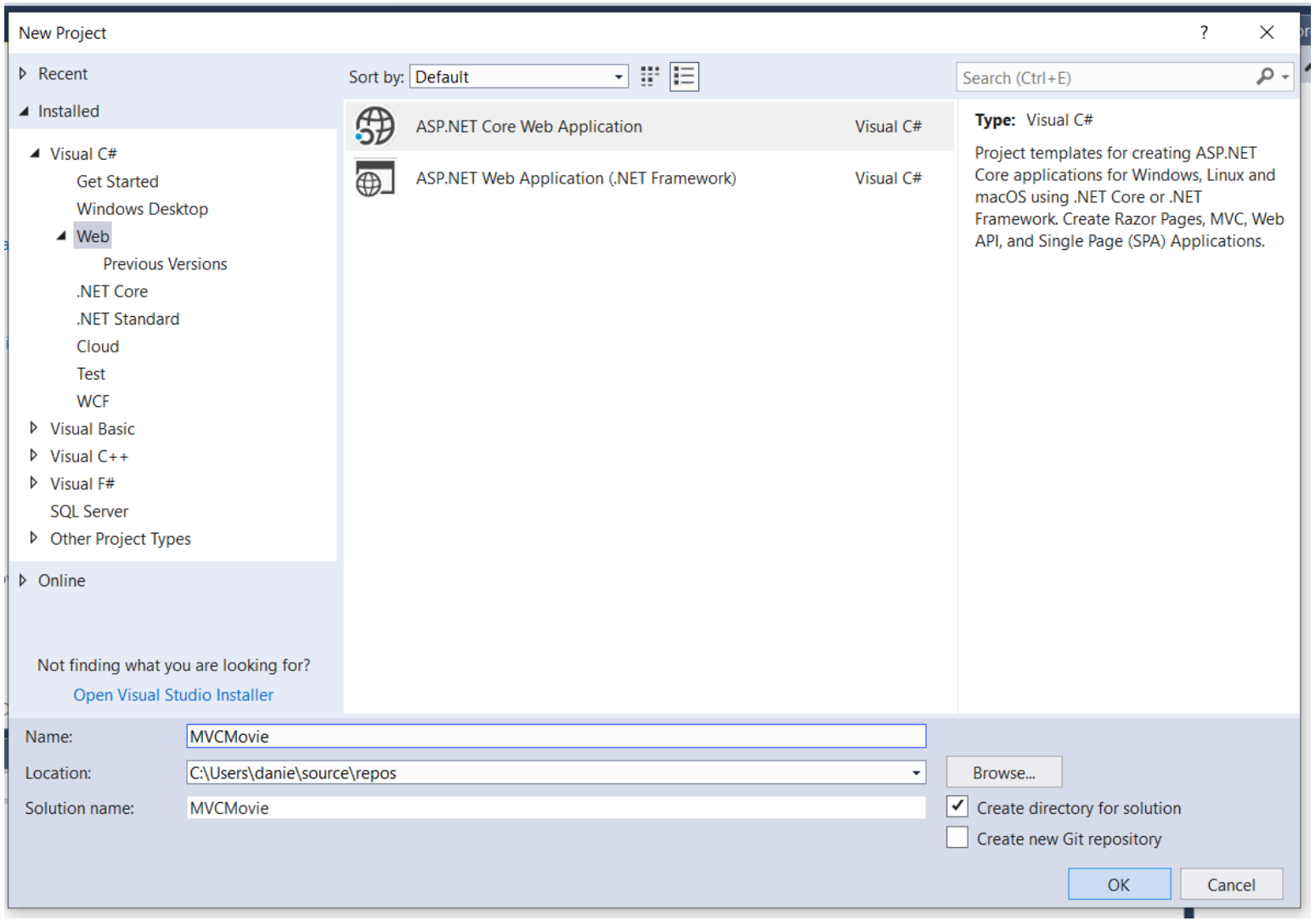
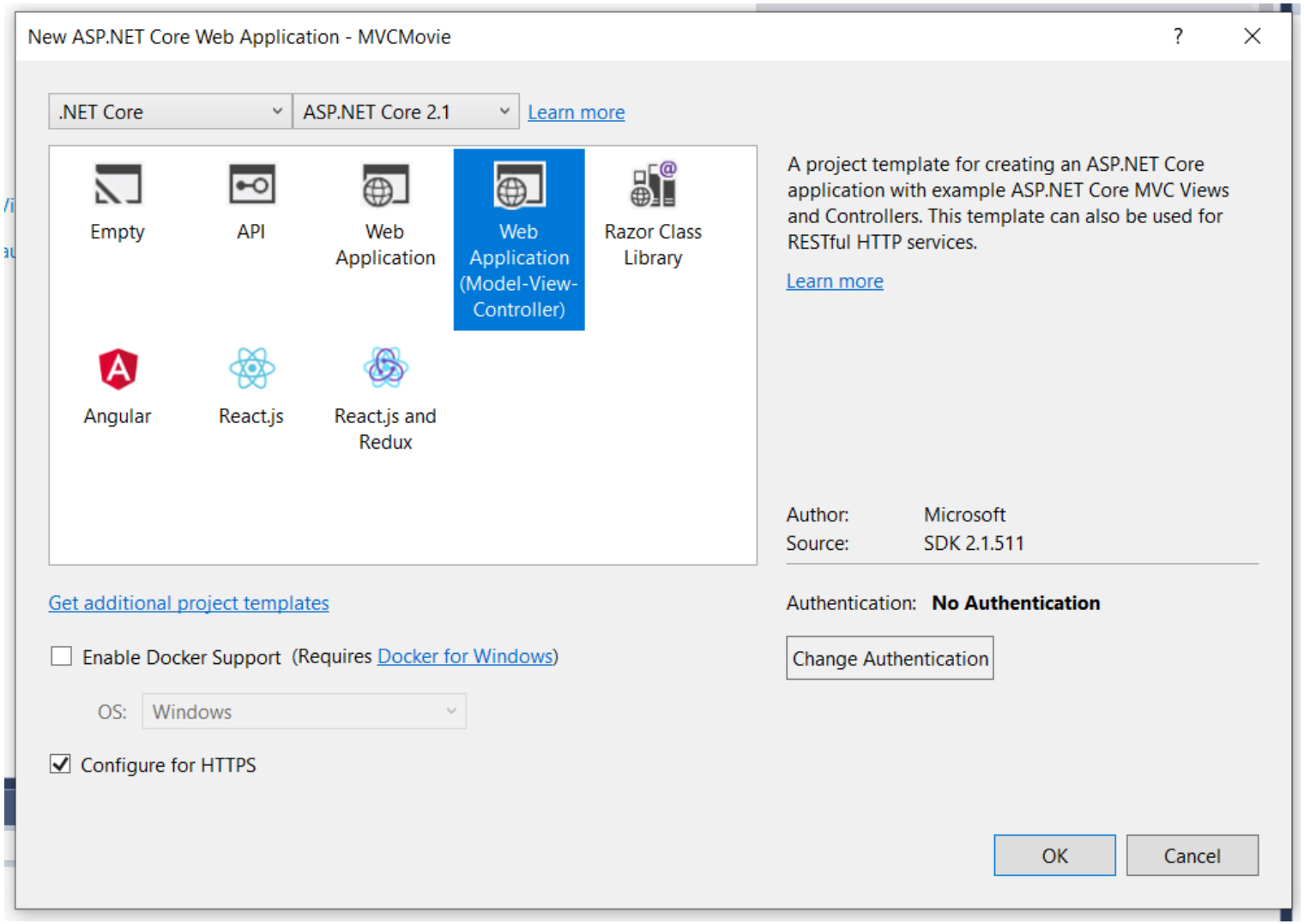
Развој на серверски WEB апликации

Practical: Building an MVC application with complex data model using EF Core

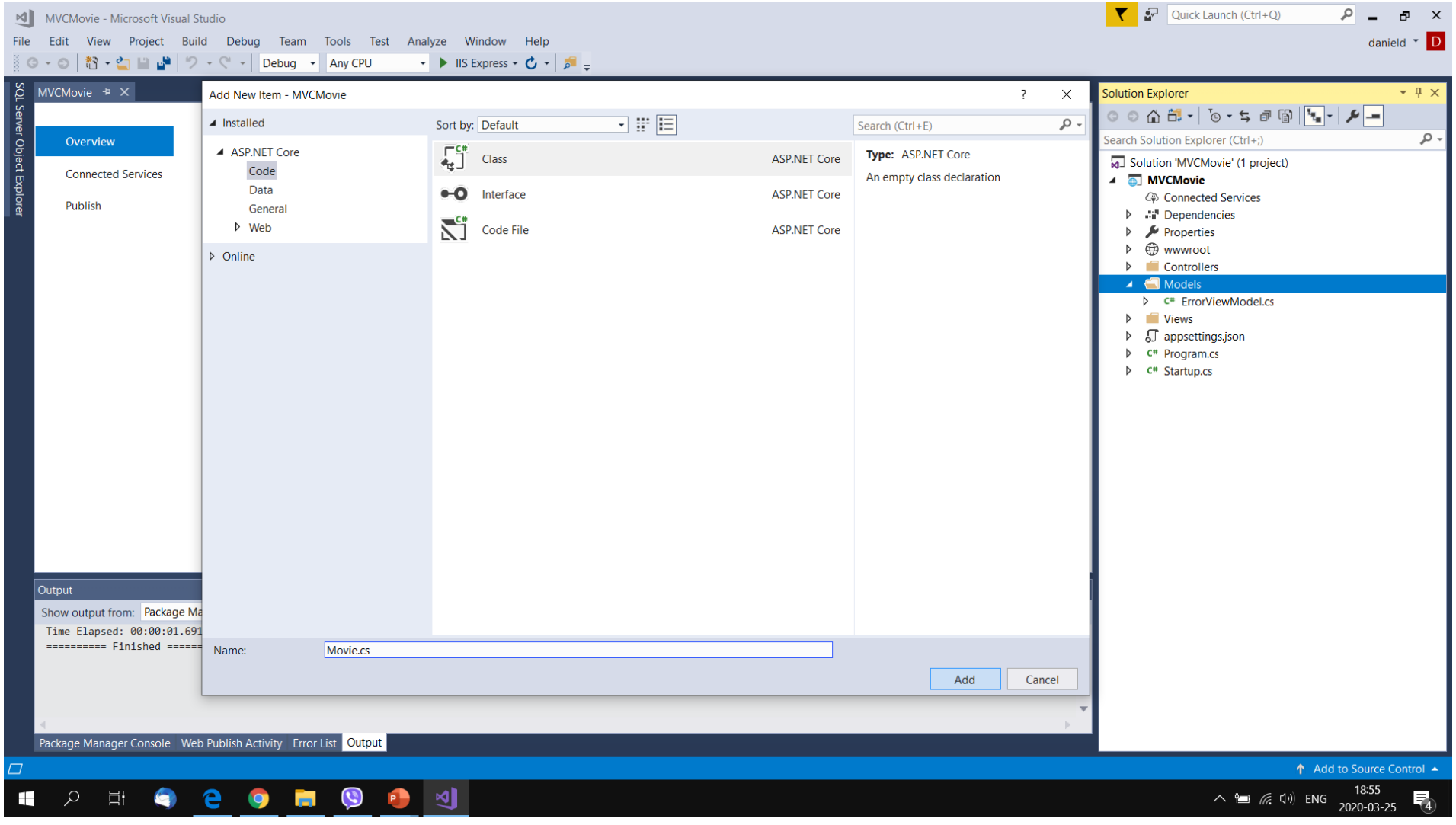
Start a new ASP.NET Core Web project



Choose the MVC design pattern



Add the models
(Right click on Model folder + Add Class)



Movie.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace MVCMovie.Models
{
    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; }

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

        [Range(1, 100)]
        [DataType(DataType.Currency)]
        public decimal? Price { get; set; }

        [Range(1, 10)]
        public decimal? Rating { get; set; }

        [Display(Name = "Director")]
        public int? DirectorId { get; set; }
        public Director? Director { get; set; }

        public ICollection<ActorMovie>? Actors { get; set; }
    }
}
```

Director.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace MVCMovie.Models
{
    public class Director
    {
        public int Id { get; set; }

        [Required]
        [StringLength(50)]
        [Display(Name = "First Name")]
        public string FirstName { get; set; }

        [Required]
        [StringLength(50)]
        [Display(Name = "Last Name")]
        public string LastName { get; set; }

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

        [NotMapped]
        public int Age {
            get {
                TimeSpan span = DateTime.Now - BirthDate;
                double years = (double)span.TotalDays / 365.2425;
                return (int)years; }
        }

        public string FullName {
            get { return String.Format("{0} {1}", FirstName, LastName); }
        }

        public ICollection<Movie>? Movies { get; set; }
    }
}
```

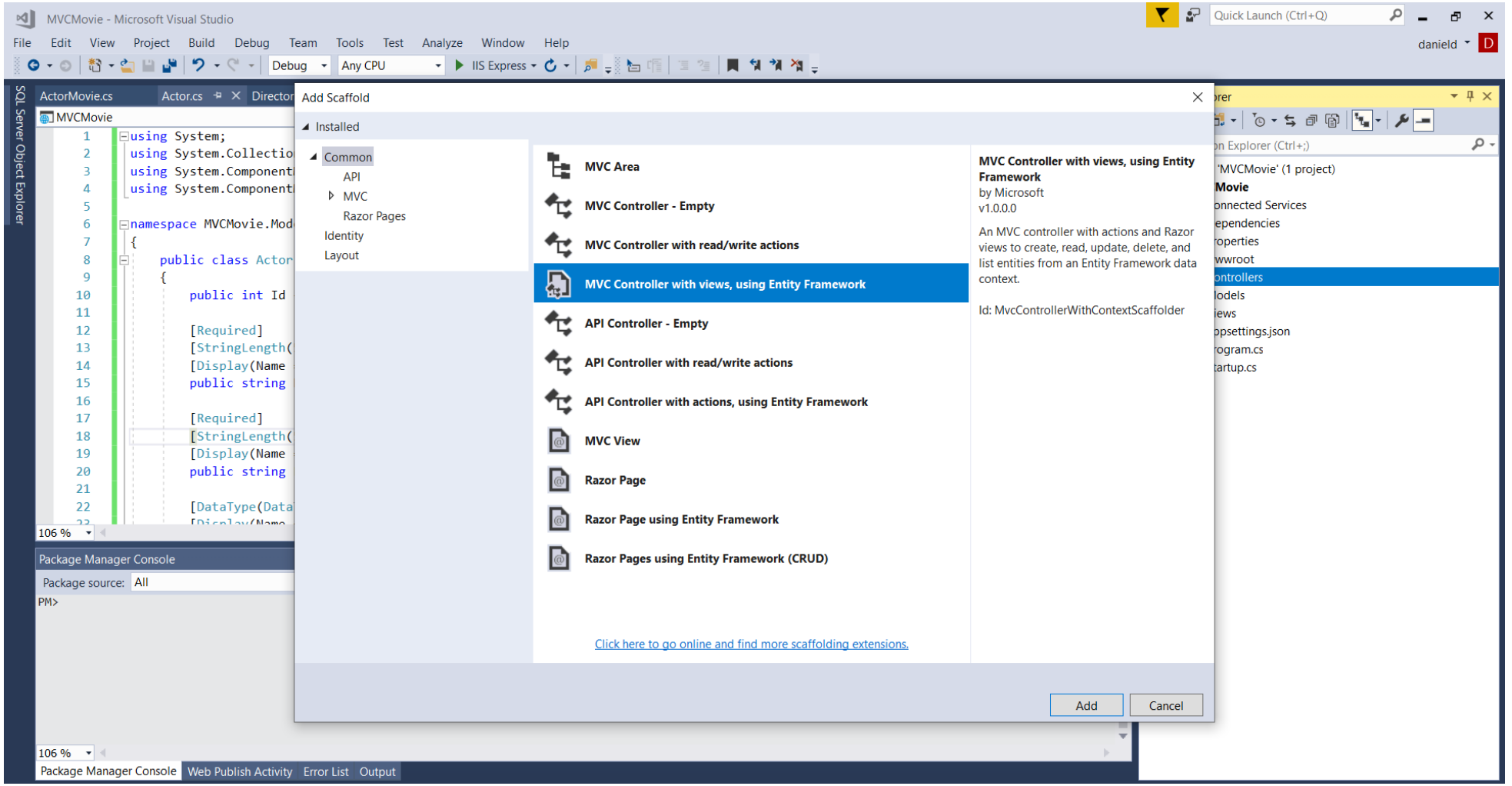
Actor.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace MVCMovie.Models
{
    public class Actor
    {
        public int Id { get; set; }

        [Required]
        [StringLength(50)]
        [Display(Name = "First Name")]
        public string FirstName { get; set; }

        [Required]
        [StringLength(50)]
        [Display(Name = "Last Name")]
        public string LastName { get; set; }

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

        [NotMapped]
        public int Age {
            get {
                TimeSpan span = DateTime.Now - BirthDate;
                double years = (double)span.TotalDays / 365.2425;
                return (int)years; }
        }

        public string FullName {
            get { return String.Format("{0} {1}", FirstName, LastName); }
        }

        public ICollection<ActorMovie>? Movies { get; set; }
    }
}
```

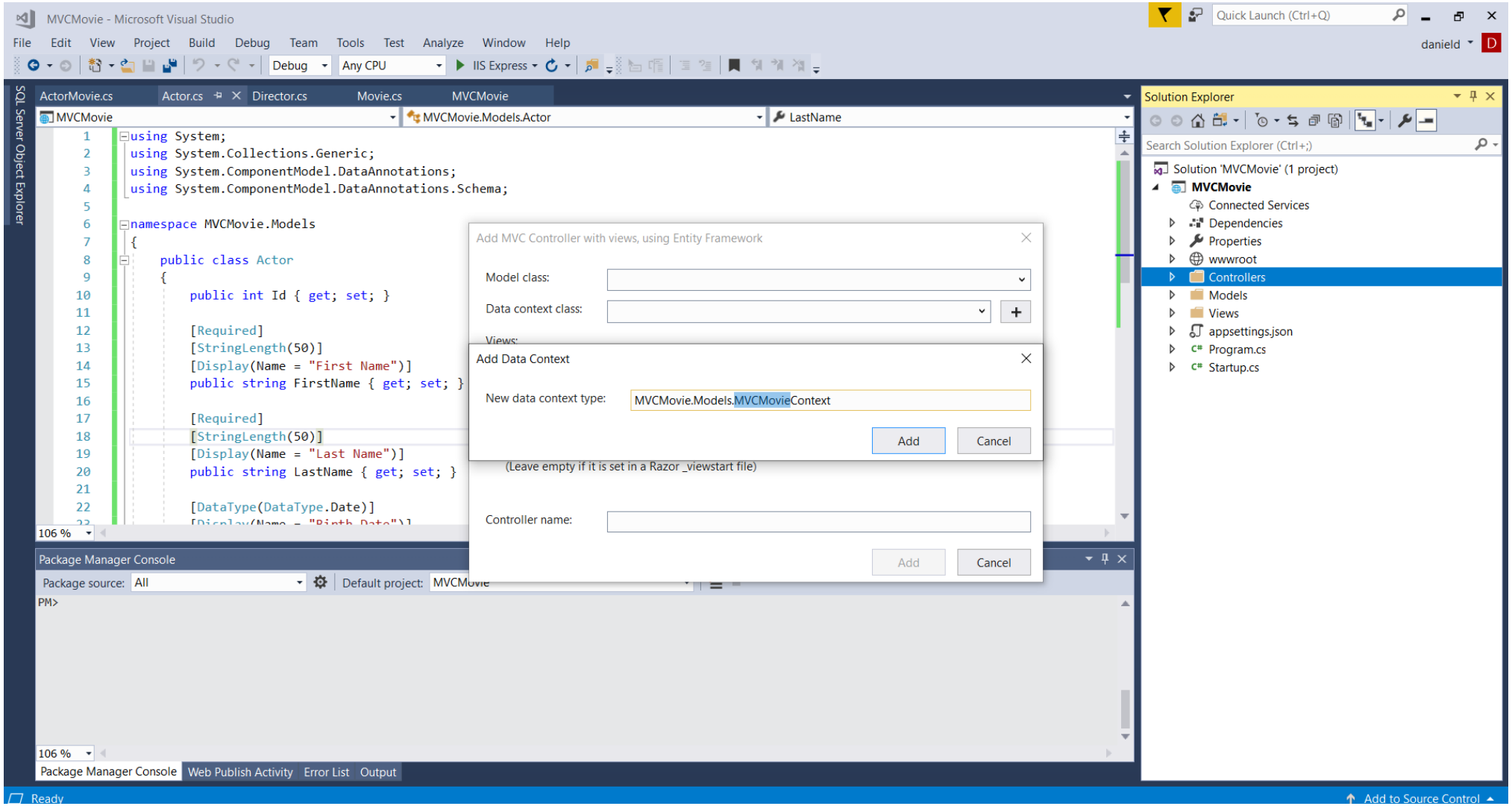
ActorMovie.cs (junction table)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace MVCMovie.Models
{
    public class ActorMovie
    {
        public int Id { get; set; }
        public int ActorId { get; set; }
        public Actor? Actor { get; set; }
        public int MovieId { get; set; }
        public Movie? Movie { get; set; }
    }
}
```

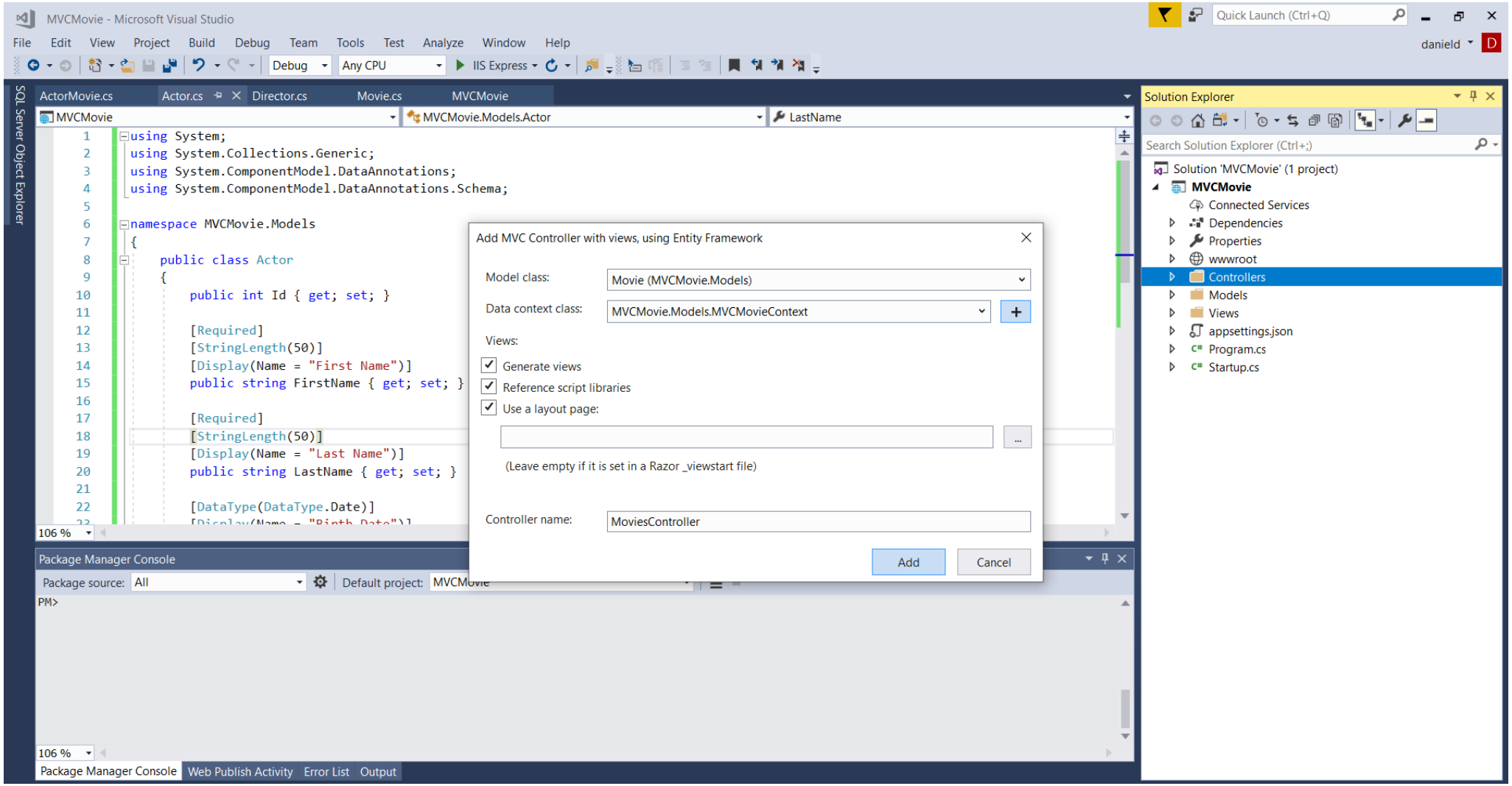
Scaffold the Models to create template Views and Controllers (Right click on Controller folder + Add new scaffolded item)



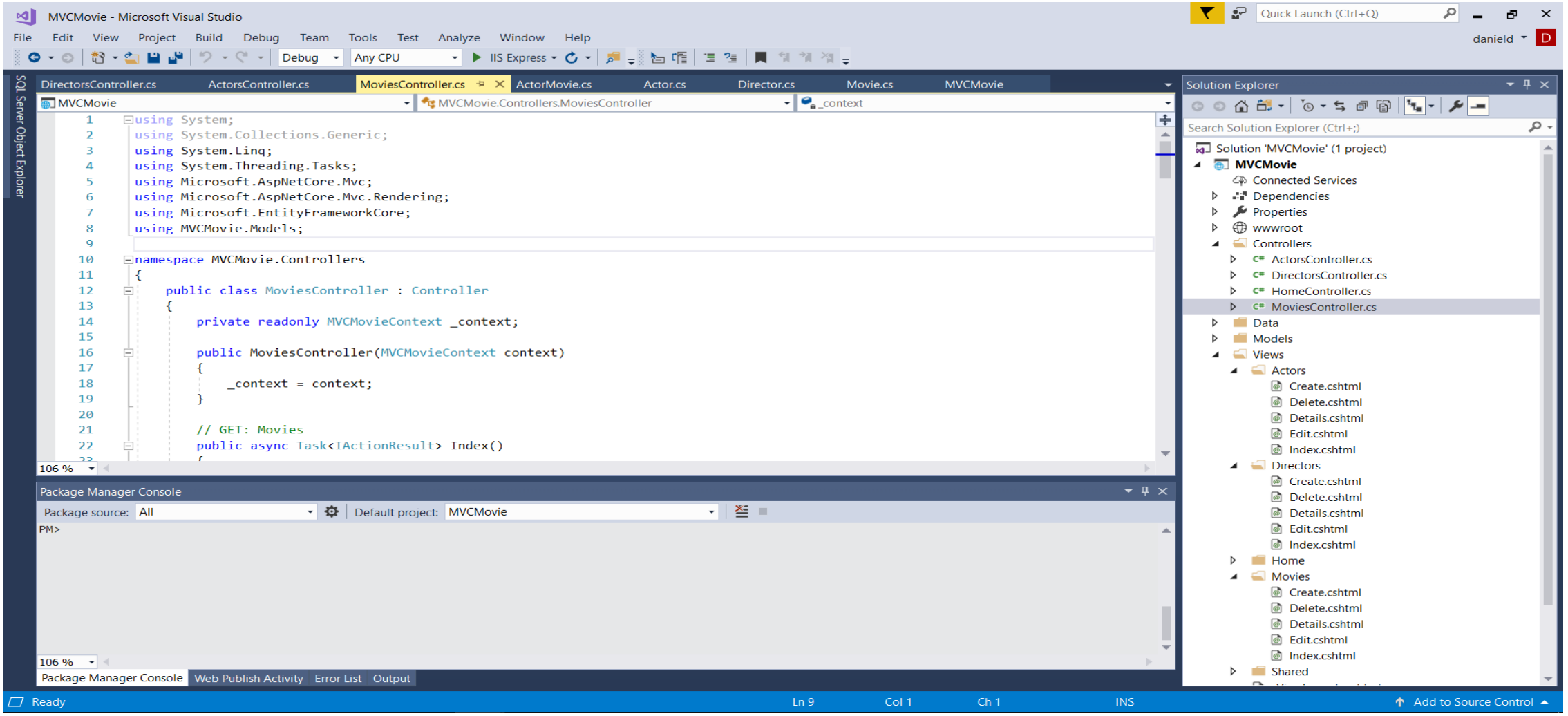
Add the data context in between



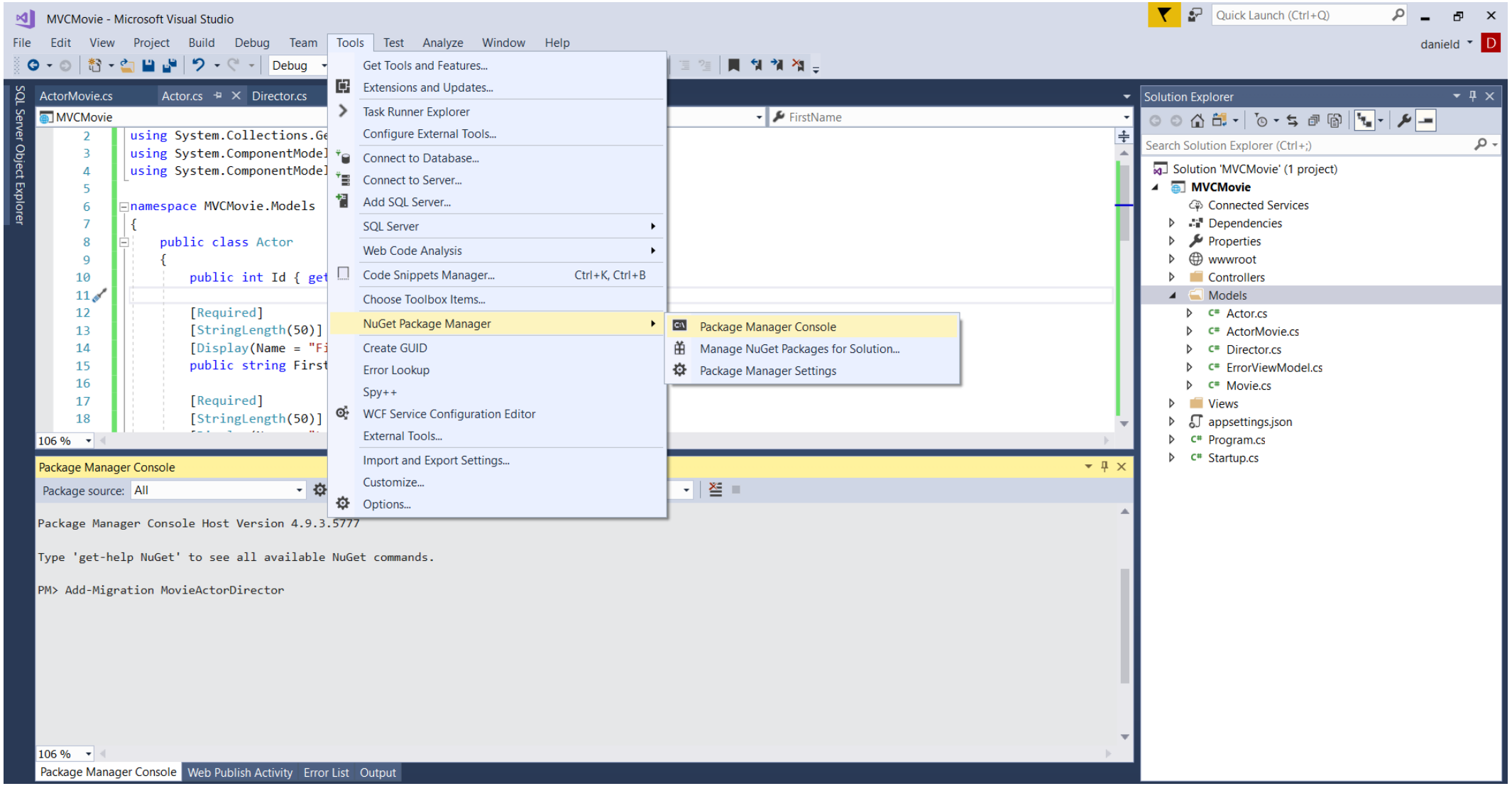
Scaffold the Movie model and then do the same for Actor and Director



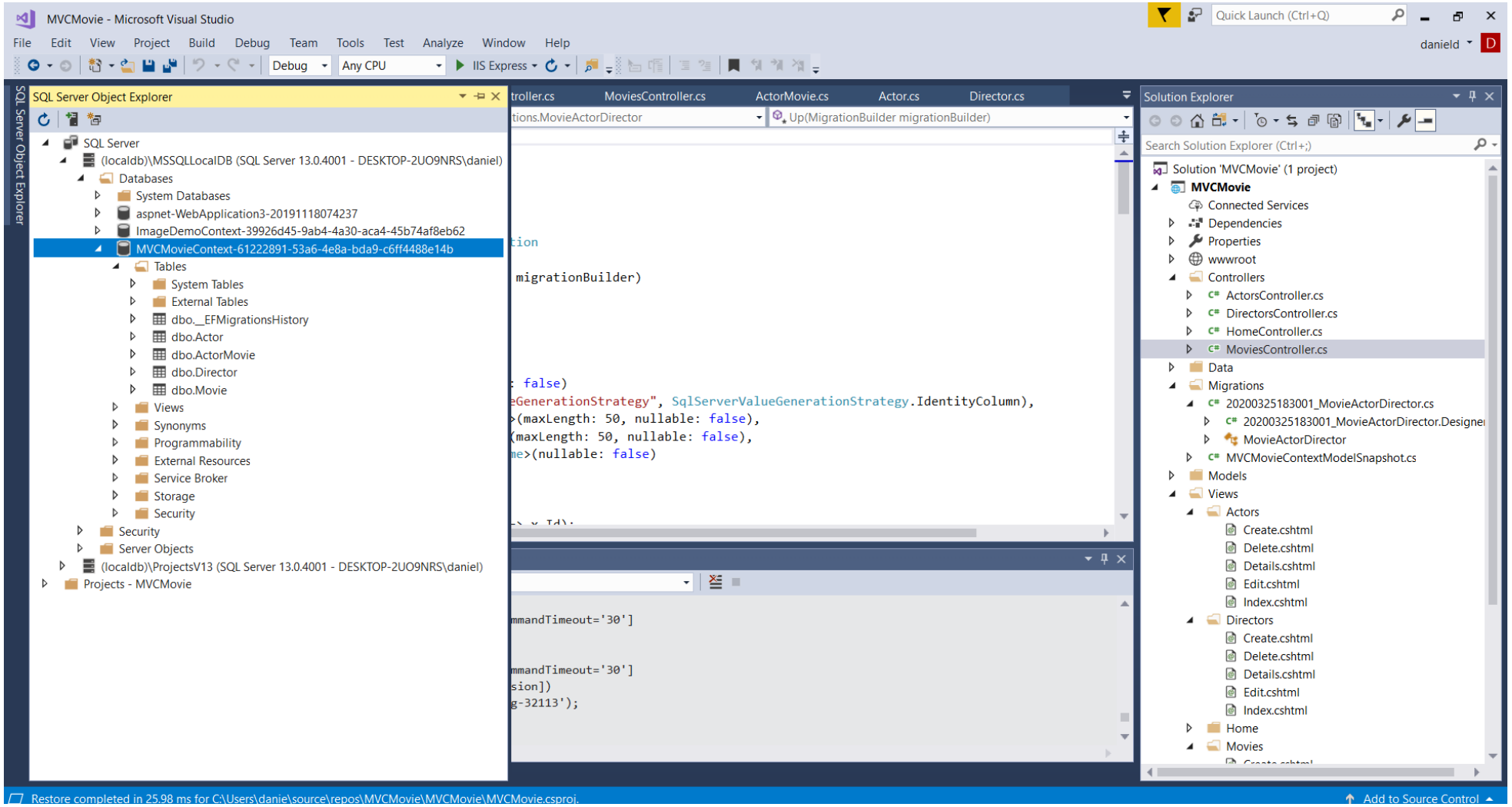
The project structure after the Scaffolding



1. Start the Package Manager Console (PMC)
2. Type “Add-Migration Initial”



1. Type “Update-Database” in PMC to create the DB
2. Check if DB created in SQL Server Object Explorer



Add the OnModelCreating method in Data\MVCMovieContext.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using MVCMovie.Models;

namespace MVCMovie.Models
{
    public class MVCMovieContext : DbContext
    {
        public MVCMovieContext (DbContextOptions<MVCMovieContext> options)
            : base(options)
        {
        }

        public DbSet<Movie> Movie { get; set; }
        public DbSet<Actor> Actor { get; set; }
        public DbSet<Director> Director { get; set; }
        public DbSet<ActorMovie> ActorMovie { get; set; }

        protected override void OnModelCreating(ModelBuilder builder)
        {
            builder.Entity<ActorMovie>()
                .HasOne<Actor>(p => p.Actor)
                .WithMany(p => p.Movies)
                .HasForeignKey(p => p.ActorId);
            //.HasPrincipalKey(p => p.Id);

            builder.Entity<ActorMovie>()
                .HasOne<Movie>(p => p.Movie)
                .WithMany(p => p.Actors)
                .HasForeignKey(p => p.MovieId);
            //.HasPrincipalKey(p => p.Id);

            builder.Entity<Movie>()
                .HasOne<Director>(p => p.Director)
                .WithMany(p => p.Movies)
                .HasForeignKey(p => p.DirectorId);
            //.HasPrincipalKey(p => p.Id);
        }
    }
}
```

Update the database

1. Open the Package Manager Console and type
   - Add-Migration MovieActorDirector
   - Update-Database

2. Add the SeedData.cs class in Models folder

SeedData.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

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

                context.Director.AddRange(
                    new Director { /*Id = 1, */FirstName = "Rob", LastName = "Reiner", BirthDate = DateTime.Parse("1947-3-6") },
                    new Director { /*Id = 2, */FirstName = "Ivan", LastName = "Reitman", BirthDate = DateTime.Parse("1946-11-27") },
                    new Director { /*Id = 3, */FirstName = "Howard", LastName = "Hawks", BirthDate = DateTime.Parse("1896-5-30") }
                );
                context.SaveChanges();

                context.Actor.AddRange(
                    new Actor { /*Id = 1, */FirstName = "Billy", LastName = "Crystal", BirthDate = DateTime.Parse("1948-3-14") },
                    new Actor { /*Id = 2, */FirstName = "Meg", LastName = "Ryan", BirthDate = DateTime.Parse("1961-11-19") },
                    new Actor { /*Id = 3, */FirstName = "Carrie", LastName = "Fisher", BirthDate = DateTime.Parse("1956-10-21") },
                    new Actor { /*Id = 4, */FirstName = "Bill", LastName = "Murray", BirthDate = DateTime.Parse("1950-9-21") },
                    new Actor { /*Id = 5, */FirstName = "Dan", LastName = "Aykroyd", BirthDate = DateTime.Parse("1952-7-1") },
                    new Actor { /*Id = 6, */FirstName = "Sigourney", LastName = "Weaver", BirthDate = DateTime.Parse("1949-11-8") },
                    new Actor { /*Id = 7, */FirstName = "John", LastName = "Wayne", BirthDate = DateTime.Parse("1907-5-26") },
                    new Actor { /*Id = 8, */FirstName = "Dean", LastName = "Martin", BirthDate = DateTime.Parse("1917-6-7") }
                );
                context.SaveChanges();

                context.Movie.AddRange(
                    new Movie
                    {
                        //Id = 1,
                        Title = "When Harry Met Sally",
                        ReleaseDate = DateTime.Parse("1989-2-12"),
                        Genre = "Romantic Comedy",
                        Rating = 5,
                        Price = 7.99M,
                        DirectorId = context.Director.Single(d => d.FirstName == "Rob" && d.LastName == "Reiner").Id
                    },
```

SeedData.cs (cont.)

```
                new Movie
                {
                    //Id = 2,
                    Title = "Ghostbusters",
                    ReleaseDate = DateTime.Parse("1984-3-13"),
                    Genre = "Comedy",
                    Rating = 6,
                    Price = 8.99M,
                    DirectorId = context.Director.Single(d => d.FirstName == "Ivan" && d.LastName == "Reitman").Id
                },
                new Movie
                {
                    //Id = 3,
                    Title = "Ghostbusters 2",
                    ReleaseDate = DateTime.Parse("1986-2-23"),
                    Genre = "Comedy",
                    Rating = 7,
                    Price = 9.99M,
                    DirectorId = context.Director.Single(d => d.FirstName == "Ivan" && d.LastName == "Reitman").Id
                },
                new Movie
                {
                    //Id = 4,
                    Title = "Rio Bravo",
                    ReleaseDate = DateTime.Parse("1959-4-15"),
                    Genre = "Western",
                    Rating = 4,
                    Price = 3.99M,
                    DirectorId = context.Director.Single(d => d.FirstName == "Howard" && d.LastName == "Hawks").Id
                }
            );
            context.SaveChanges();

            context.ActorMovie.AddRange(
                new ActorMovie { ActorId = 1, MovieId = 1 },
                new ActorMovie { ActorId = 2, MovieId = 1 },
                new ActorMovie { ActorId = 3, MovieId = 1 },
                new ActorMovie { ActorId = 4, MovieId = 2 },
                new ActorMovie { ActorId = 5, MovieId = 2 },
                new ActorMovie { ActorId = 6, MovieId = 2 },
                new ActorMovie { ActorId = 4, MovieId = 3 },
                new ActorMovie { ActorId = 5, MovieId = 3 },
                new ActorMovie { ActorId = 6, MovieId = 3 },
                new ActorMovie { ActorId = 7, MovieId = 4 },
                new ActorMovie { ActorId = 8, MovieId = 4 }
            );

            context.SaveChanges();
        }
    }
}
```

```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using MVCMovie.Data;
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<MVCMovieContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("MVCMovieContext")));

// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

using (var scope = app.Services.CreateScope())
{
    var services = scope.ServiceProvider;
    SeedData.Initialize(services);
}

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

Modify Program.cs to seed data into the database

Modify Views\Shared_Layout.cshtml to provide navigation to Movies, Actors and Directors

```
<div class="navbar-header">
    <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
    </button>
    <a asp-area="" asp-controller="Movies" asp-action="Index" class="navbar-brand">Movies List</a>
</div>
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        <li><a asp-area="" asp-controller="Movies" asp-action="Index">Movies</a></li>
        <li><a asp-area="" asp-controller="Directors" asp-action="Index">Directors</a></li>
        <li><a asp-area="" asp-controller="Actors" asp-action="Index">Authors</a></li>
        <li><a asp-area="" asp-controller="Home" asp-action="About">About</a></li>
        <li><a asp-area="" asp-controller="Home" asp-action="Contact">Contact</a></li>
    </ul>
</div>
```

Change routing in Startup.cs to navigate to Movies controller by default

```
app.UseMvc(routes =>
            {
                routes.MapRoute(
                    name: "default",
                    template: "{controller=Movies}/{action=Index}/{id?}");
            });
```

Change the select input in Views\Movies\Create.cshtml and Edit.cshtml to provide empty input

```
<select asp-for="DirectorId" class="form-control" asp-items="ViewBag.DirectorId">
        <option value="">Select</option>
</select>
```

Modify all occurrences of *either* of the following lines in MoviesController.cs, so that Director FullName is shown on the views

```
ViewData["DirectorId"] = new SelectList(_context.Set<Director>(), "Id", "FirstName""FullName");
ViewData["DirectorId"] = new SelectList(_context.Set<Director>(), "Id", "FirstName""FullName", movie.DirectorId);
```

Modify all occurrences of *either* of the following codes in Views\Movies\Index.cshtml, Details.cshtml, Delete.cshtml

```
<td>
    @Html.DisplayFor(model => modelItem.Director.FirstName)
    @if (item.DirectorId != null)
    {
        <a asp-controller="Directors" asp-action="Details" asp-route-id="@item.DirectorId"> @Html.DisplayFor(modelItem => item.Director.FullName)</a>
    }
</td>

<dd>
    @Html.DisplayFor(model => model.Director.FirstName)
    @if (Model.DirectorId != null)
    {
        <a asp-controller="Directors" asp-action="Details" asp-route-id="@Model.DirectorId">@Html.DisplayFor(model => model.Director.FullName)</a>
    }
</dd>
```

Modify all occurrences of *either* of the following lines in MoviesController.cs, so that Actors are loaded as related data

```
var mVCMovieContext = _context.Movie.Include(m => m.Director).Include(m => m.Actors).ThenInclude(m => m.Actor);
var movie = await _context.Movie
                .Include(m => m.Director)
                .Include(m => m.Actors).ThenInclude(m => m.Actor)
                .FirstOrDefaultAsync(m => m.Id == id);
```

Modify Views\Movies\Details.cshtml, Delete.cshtml, add the following code in the respective place to list the authors

```
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Actors)
</dt>
<dd class="col-sm-10">
    @foreach (var actor in Model.Actors)
    {
        <a asp-controller="Actors" asp-action="Details" asp-route-id="@actor.ActorId">
@actor.Actor.FullName </a> <br />
    }
</dd>
```

Title	When Harry Met Sally
Release Date	1989-02-12
Genre	Romantic Comedy
Price	$7.99
Rating	5.00
Director	Rob Reiner
Actors	Billy Crystal
	Meg Ryan
	Carrie Fisher

Modify Views\Movies\Index.cshtml, add the following codes in the respective places in the table

```
<th>
   @Html.DisplayNameFor(model => model.Actors)
</th>
...
<td>
   @foreach (var actor in item.Actors)
   {
      <a asp-controller="Actors" asp-action="Details" asp-route-id="@actor.ActorId">  @actor.Actor.FullName </a> <br/>
   }
</td>
```

Title	Release Date	Genre	Price	Rating	Director	Actors	
When Harry Met Sally	1989-02-12	Romantic Comedy	$7.99	5.00	Rob Reiner	Billy Crystal Meg Ryan Carrie Fisher	Edit | Details | Delete
Ghostbusters	1984-03-13	Comedy	$8.99	6.00	Ivan Reitman	Bill Murray Dan Aykroyd Sigourney Weaver	Edit | Details | Delete
Ghostbusters 2	1986-02-23	Comedy	$9.99	7.00	Ivan Reitman	Bill Murray Dan Aykroyd Sigourney Weaver	Edit | Details | Delete
Rio Bravo	1959-04-15	Western	$3.99	4.00	Howard Hawks	John Wayne Dean Martin	Edit | Details | Delete

Add age field in Director and Actor views in the respective places

```
<th>
    @Html.DisplayNameFor(model => model.Age)
</th>
...
<td>
    @Html.DisplayFor(modelItem => item.Age)
</td>
```

← Index.cshtml

First Name	Last Name	Birth Date	Age	
Rob	Reiner	1947-03-06	**73**	Edit | Details | Delete
Ivan	Reitman	1946-11-27	**73**	Edit | Details | Delete
Howard	Hawks	1896-05-30	**123**	Edit | Details | Delete

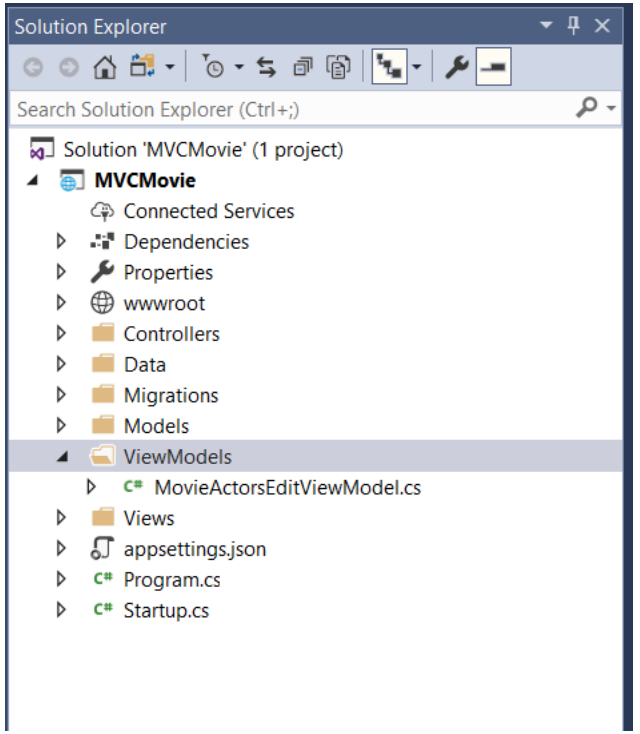
```
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Age)
</dt>
<dd class="col-sm-10">
    @Html.DisplayFor(model => model.Age)
</dd>
```

← Details.cshtml and Delete.cshmtl

First Name	Rob
Last Name	Reiner
Birth Date	1947-03-06
Age	73

Add a ViewModels folder in the project and add a class MovieActorsEditViewModel.cs in the folder

```
using Microsoft.AspNetCore.Mvc.Rendering;
using MVCMovie.Models;
using System.Collections.Generic;

namespace MVCMovie.ViewModels
{
    public class MovieActorsEditViewModel
    {
        public Movie Movie { get; set; }
        public IEnumerable<int>? SelectedActors { get; set; }
        public IEnumerable<SelectListItem>? ActorList { get; set; }
    }
}
```



Change the HttpGet Edit action in the MoviesController to use the new viewmodel

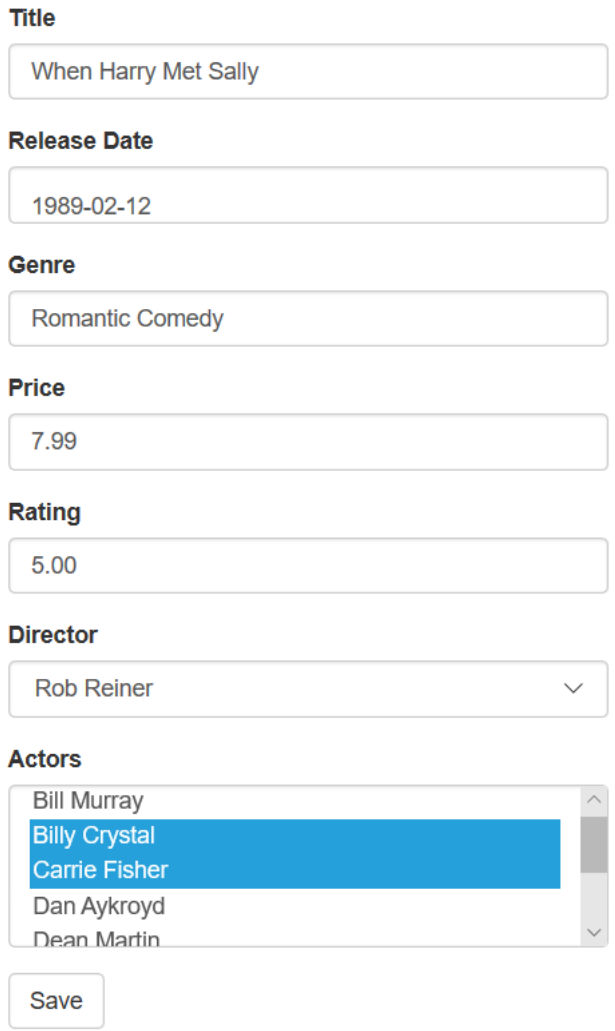
```
using MVCMovie.ViewModels;
...
// GET: Movies/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var movie = _context.Movie.Where(m => m.Id == id).Include(m => m.Actors).First();

    if (movie == null)
    {
        return NotFound();
    }

    var actors = _context.Actor.AsEnumerable();
    actors = actors.OrderBy(s => s.FullName);

    MovieActorsEditViewModel viewmodel = new MovieActorsEditViewModel
    {
        Movie = movie,
        ActorList = new MultiSelectList(actors, "Id", "FullName"),
        SelectedActors = movie.Actors.Select(sa => sa.ActorId)
    };

    ViewData["DirectorId"] = new SelectList(_context.Director, "Id", "FullName", movie.DirectorId);
    return View(viewmodel);
}
```

Change the HttpPost Edit action in the MoviesController to use the new viewmodel

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, MovieActorsEditViewModel viewmodel)
{
    if (id != viewmodel.Movie.Id) { return NotFound(); }

    if (ModelState.IsValid) {
        try {
            _context.Update(viewmodel.Movie);
            await _context.SaveChangesAsync();

            IEnumerable<int> newActorList = viewmodel.SelectedActors;
            IEnumerable<int> prevActorList = _context.ActorMovie.Where(s => s.MovieId == id).Select(s => s.ActorId);
            IQueryable<ActorMovie> toBeRemoved = _context.ActorMovie.Where(s => s.MovieId == id);
            if (newActorList != null) {
                toBeRemoved = toBeRemoved.Where(s => !newActorList.Contains(s.ActorId));
                foreach (int actorId in newActorList)
                {
                    if (!prevActorList.Any(s => s == actorId)) {
                        _context.ActorMovie.Add(new ActorMovie { ActorId = actorId, MovieId = id });
                    }
                }
            }
            _context.ActorMovie.RemoveRange(toBeRemoved);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException) {
            if (!MovieExists(viewmodel.Movie.Id)) { return NotFound(); }
            else { throw; }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["DirectorId"] = new SelectList(_context.Director, "Id", "FullName", viewmodel.Movie.DirectorId);
    return View(viewmodel);
```

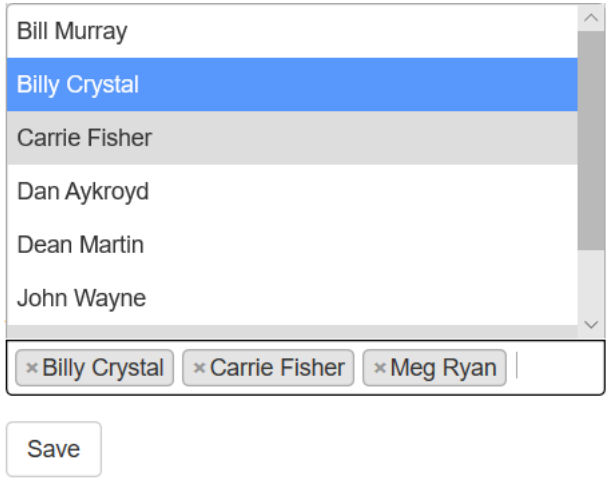
Modify Views\Movies\Edit.cshtml to use the new viewmodel and list Actors in multiselect list

```
@model MVCMovie.ViewModels.MovieActorsEditViewModel

...

<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="Movie.Id" />
            <div class="form-group">
                <label asp-for="Movie.Title" class="control-label"></label>
                <input asp-for="Movie.Title" class="form-control" />
                <span asp-validation-for="Movie.Title" class="text-danger"></span>
            </div>
...
            <div class="form-group">
                <label class="control-label">Actors</label>
                <select asp-for="SelectedActors" class="form-control js-example-basic-multiple"
asp-items="Model.ActorList"> </select>
                <span asp-validation-for="SelectedActors" class="text-danger"></span>
            </div>
...
        </form>
    </div>
</div>
...
```



Add better styling for the multiselect list in Views\Movies\Edit.cshtml

```
...
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js"></script>
    <script>
        $(document).ready(function () {
            $("#SelectedActors").select2();
        });
    </script>
}
```



Load the css style in Views\Shared_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/css/select2.min.css" rel="stylesheet" />
    <title>@ViewData["Title"] - MVCMovie</title>
```