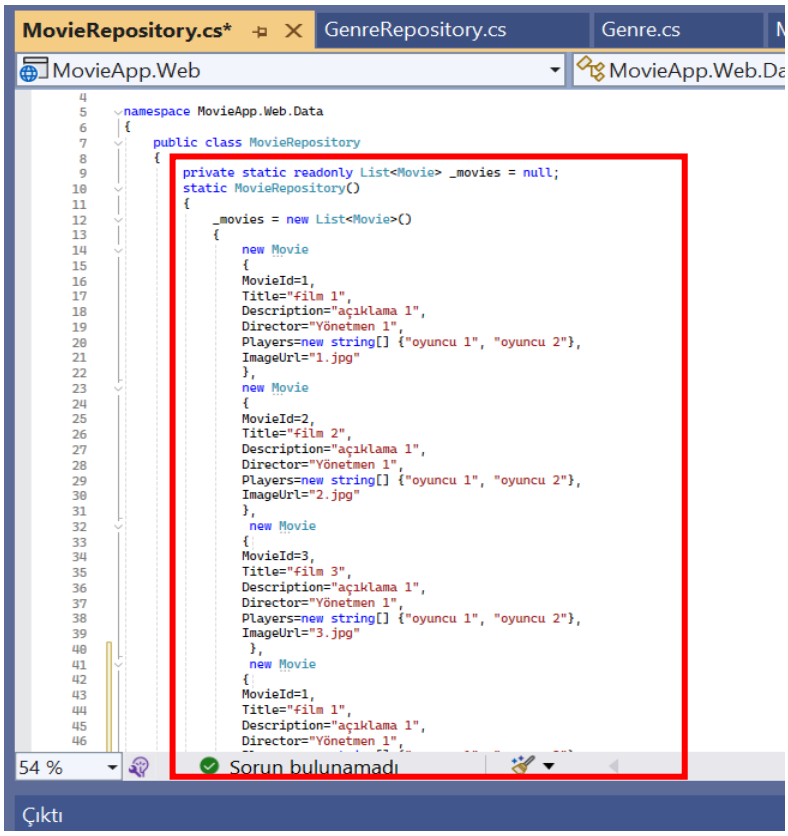
ANKARA
HACI BAYRAM VELİ ÜNİVERSİTESİ
UZAKTAN EĞİTİM UYGULAMA VE ARAŞTIRMA MERKEZİ

İNTERNET PROGRAMCILIĞI

Bu bölümde proje içinde kullanılacak olan verilerin veritabanına kaydedilmesi, verilerin listelenmesi ve detay sayfası işlemlerine yönelik uygulamalara yer verilmiştir.

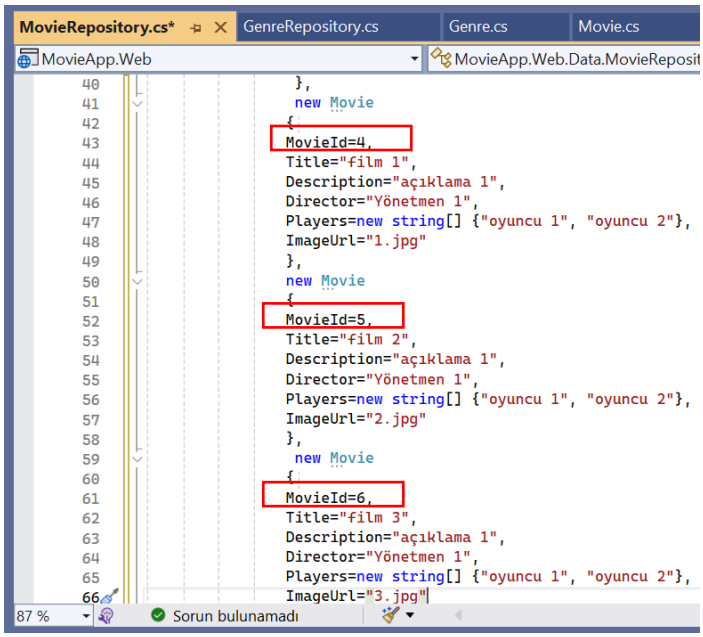
1. VERİLERİN LİSTELENMESİ

ADIM 1: MovieRepository.cs dosyasında 3 tane film bilgisi vardı, 6 film bilgisi olacak şekilde düzenleyelim. 3 lü listeyi kopyalayıp altına yeniden yapıştırıyoruz (parantez sonrası virgüle dikkat ediniz).



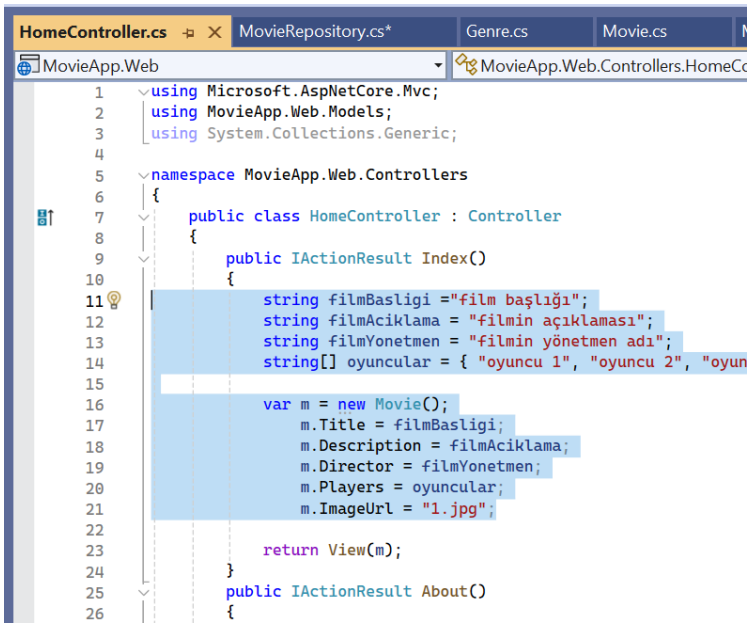

```
namespace MovieApp.Web.Data
{
    public class MovieRepository
    {
        private static readonly List<Movie> _movies = null;
        static MovieRepository()
        {
            _movies = new List<Movie>()
            {
                new Movie
                {
                MovieId=1,
                Title="film 1",
                Description="açıklama 1",
                Director="Yönetmen 1",
                Players=new string[] {"oyuncu 1", "oyuncu 2"},
                ImageUrl="1.jpg"
                },
                new Movie
                {
                MovieId=2,
                Title="film 2",
                Description="açıklama 1",
                Director="Yönetmen 1",
                Players=new string[] {"oyuncu 1", "oyuncu 2"},
                ImageUrl="2.jpg"
                },
                 new Movie
                {
                MovieId=3,
                Title="film 3",
                Description="açıklama 1",
                Director="Yönetmen 1",
                Players=new string[] {"oyuncu 1", "oyuncu 2"},
                ImageUrl="3.jpg"
                 },
                 new Movie
                {
                MovieId=1,
                Title="film 1",
                Description="açıklama 1",
                Director="Yönetmen 1",
```

ADIM 2: MovieId bilgilerini 3 ten sonra devam edecek şekilde düzenleyelim.

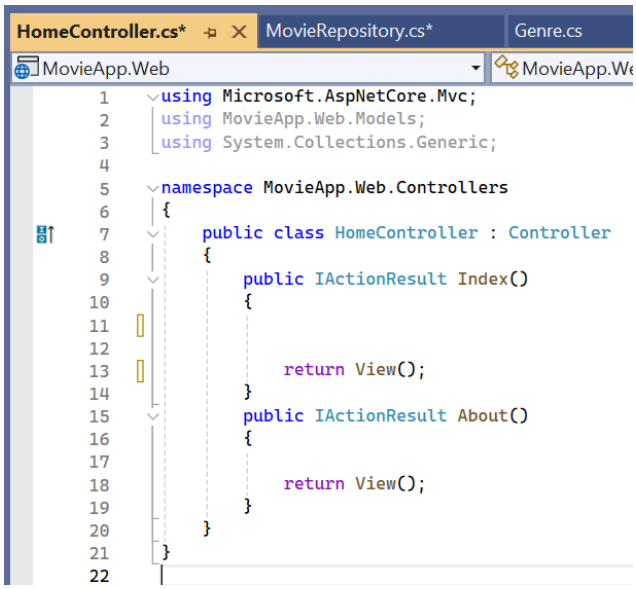


ADIM 3: Anasayfada (Home Controller aracılığıyla) tek bir film bilgisi gösteriyorduk. **HomeController** içindeki tek film ile ilgili kodları temizleyelim. Return View içini temizlemeyi unutmayınız!

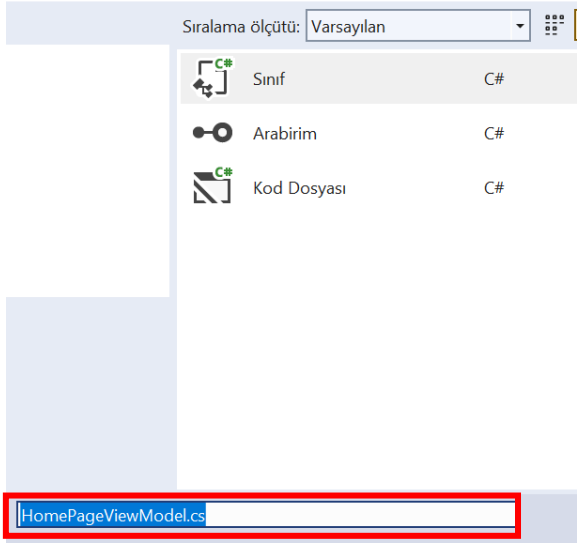
NOT: Film Bilgilerini Repository üzerinden anasayfaya aktaracağız. HomeController da film bilgileri yerine, MovieRepository.cs ye yönlendirme yapacağız.



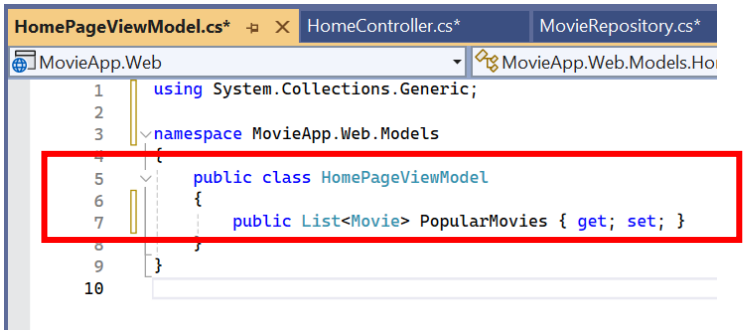
ADIM 4: Silme işleminden sonra **HomeController** görünümü aşağıdaki gibi olmalı.



ADIM 5: Repository üzerinden aktaracağımız bilgilerin görselleri için Models a sağ tıklayarak yeni bir sınıf (class) oluşturuyoruz. **HomePageViewModel.cs** olarak isimlendiriyoruz.

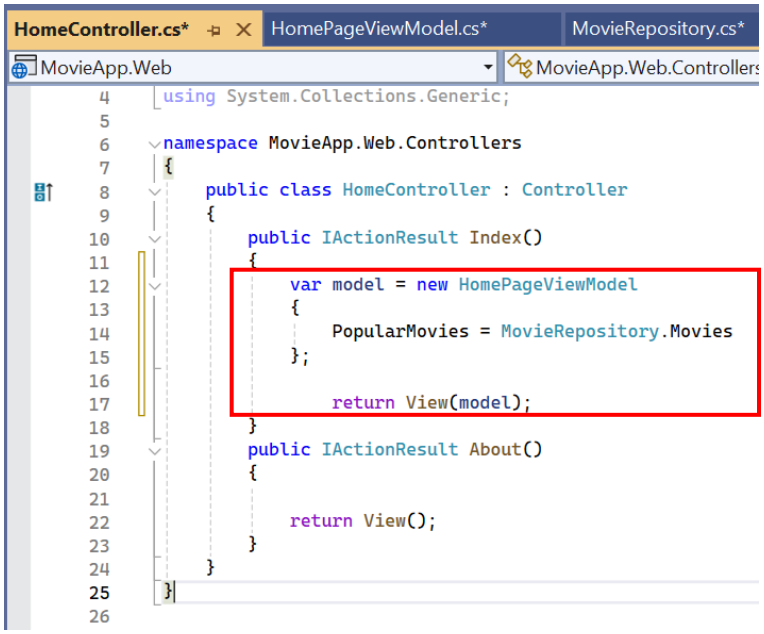


ADIM 6: Örneğin anasayfada göstermek istediğimiz film listesini filtreleyip gösteriyor olsaydık. (Populerfilmler gibi) **HomePageViewModel.cs** sayfasına ilgili kod bloğunu şu şekilde ekleriz.

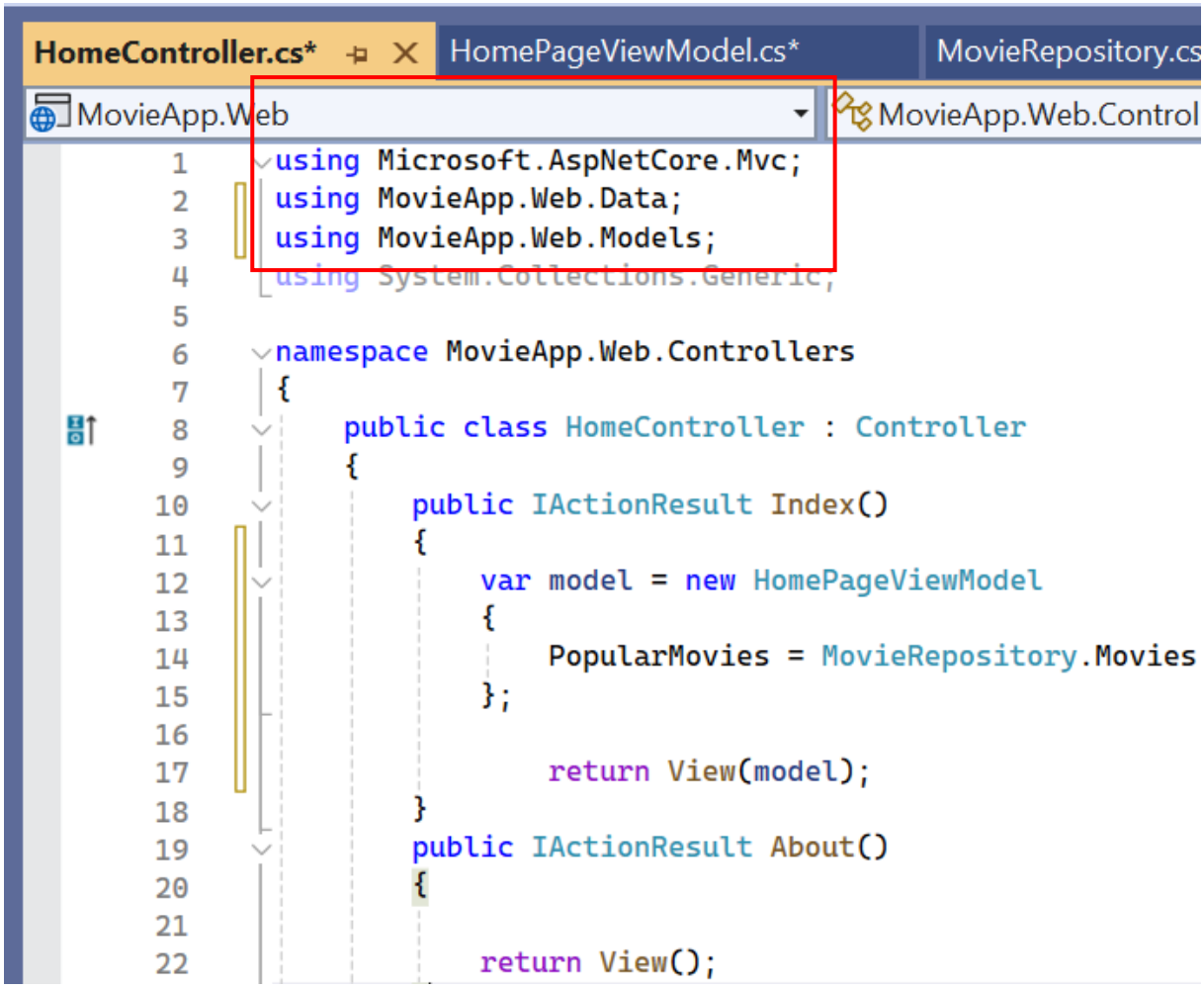


ADIM 7: HomeController.cs dosyası HomePageViewModel.cs e göre düzenlenir.

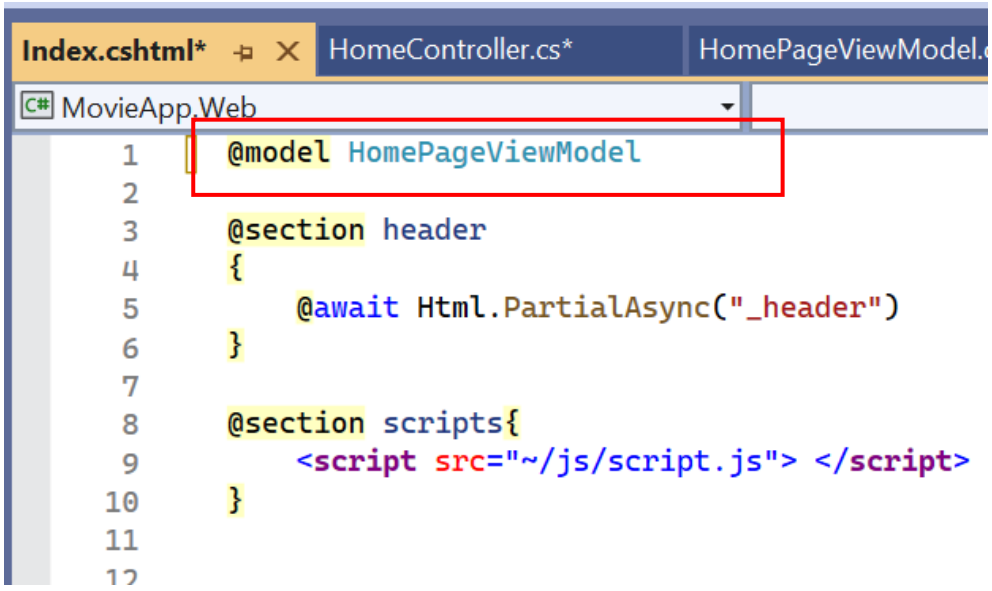
NOT: Popüler filmlerin bilgileri MovieRepository üzerinden gelecek şekilde düzenlendi.



ADIM 8: ilgili namespace lerin varlığını **HomeController** da kontrol ediniz eksik olanları tamamlayınız.



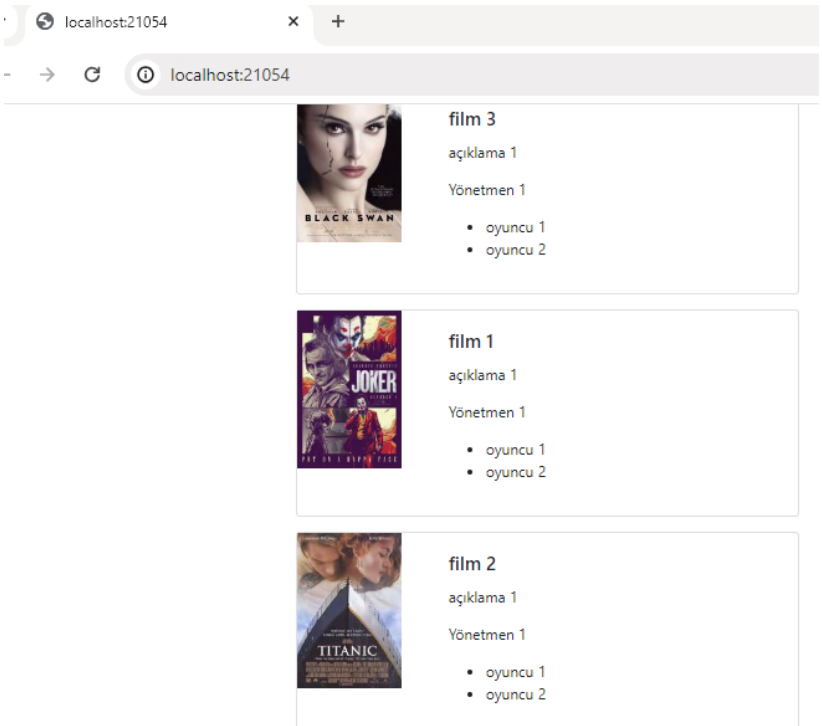
ADIM 9: HomePageViewModel i Anasayfada karşılamak için ilgili kod satırını Views Klasöründeki Home içerisindeki **Index.cshtml** de düzenliyoruz. @model Movie yerine @model HomePageViewModel yazılır.



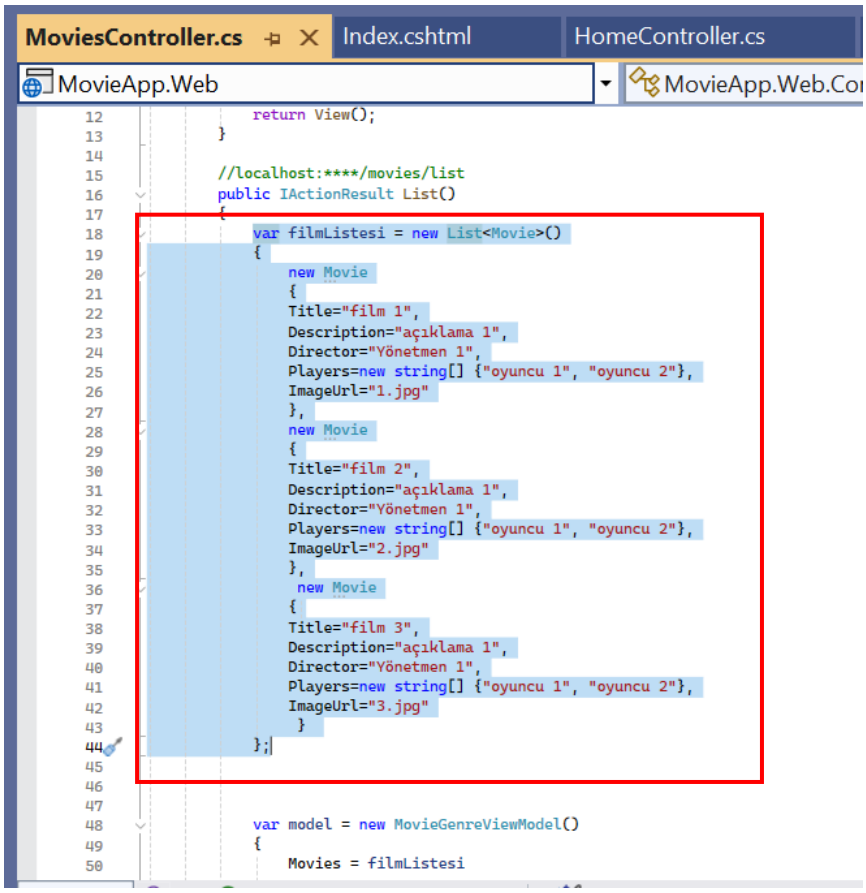
ADIM 10: Index.cshtml sayfasında döngü kod bloğu aşağıdaki gibi düzenlenir.

```
Index.cshtml*  HomeController.cs*  HomePageViewModel.cs*  Mo
MovieApp.Web
1   @model HomePageViewModel
2
3   @section header
4   {
5       @await Html.PartialAsync("_header")
6   }
7
8   @section scripts{
9       <script src="~/js/script.js"> </script>
10  }
11
12
13          <h1> Film Listesi</h1>
14          <div id="filmler">
15              @foreach (var movie in Model.PopularMovies)
16              {
17                  @await Html.PartialAsync("_movie", movie)
18              }
19
20          </div>
21
```

ADIM 11: Çalıştırıp **anasayfada** 6 tane film olup olmadığı kontrol edilir.



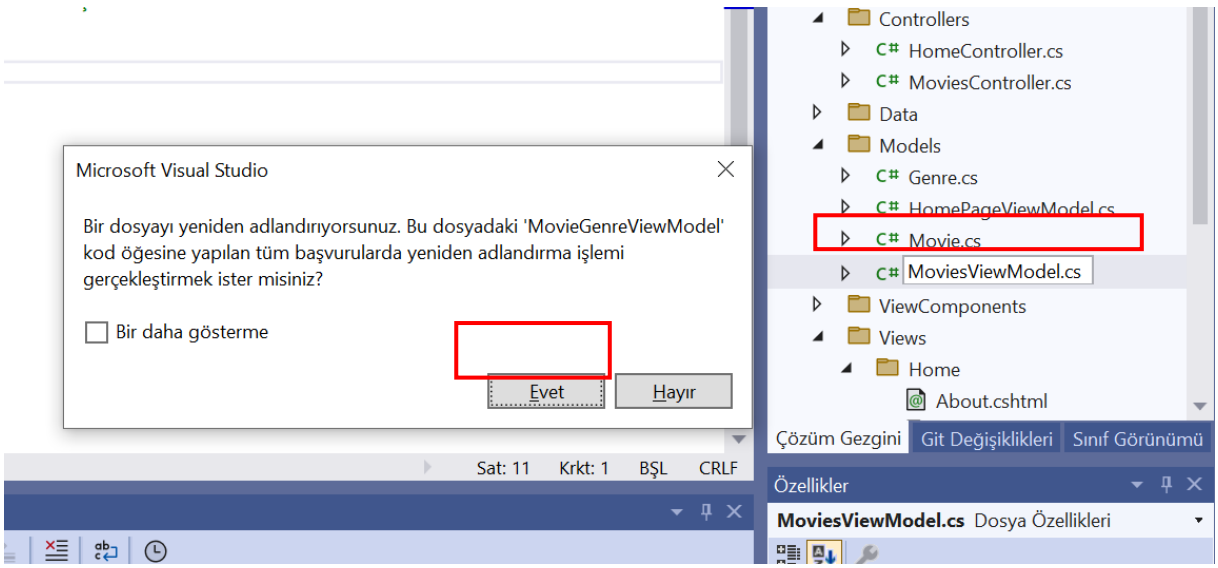
ADIM 12: Movie/list deki film bilgileri de Repository üzerinden gelecek şekilde ayarlansın. Bunun için öncelikle **MoviesController.cs** içindeki film listesi kod bloğunu silelim.



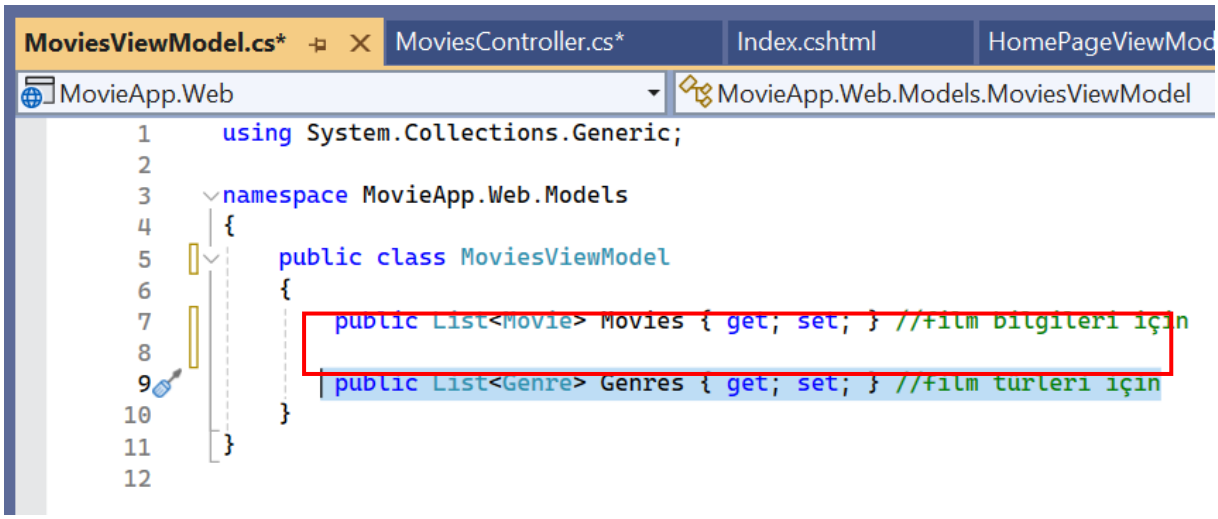
ADIM 13: Yine MoviesController.cs içindeki MovieGenreViewModel(), MoviesViewModel() olarak değiştirilir.

```
MoviesController.cs*  Index.cshtml  HomeController.cs
MovieApp.Web                        MovieApp.Web.Con
12              return View();
13          }
14
15          //localhost:****/movies/list
16          public IActionResult List()
17          {
18
19              var model = new MoviesViewModel()
20              {
21                  Movies = filmListesi
22
23              };
24
25              return View("Movies", model);
26          }
27
28          //localhost:****/movies/details
29          public string Details()
30          {
31              return "Film Detayı";
32          }
33      }
34  }
35
```

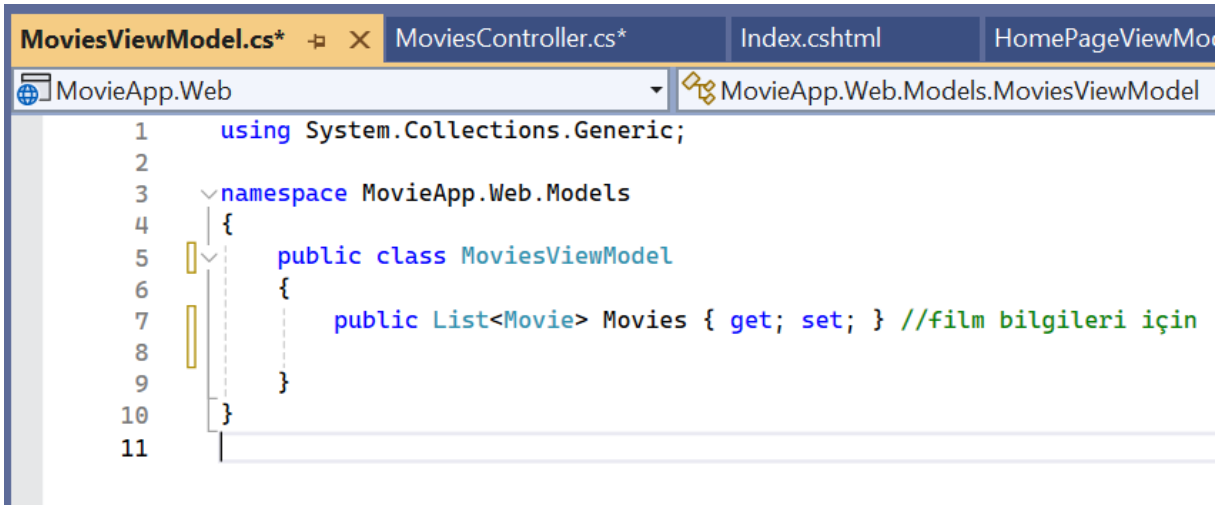
ADIM 14: Sağ Tarafta **Models** Klasörü içindeki MoviesGenreViewModel.cs dosyasının ismi MoviesViewModel.cs olarak güncellenir.



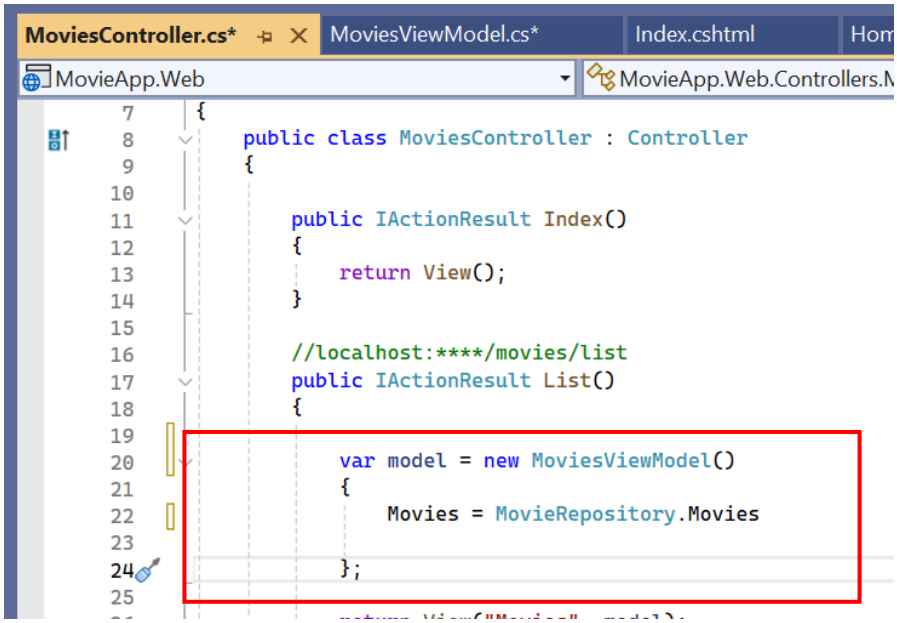
ADIM 15: MoviesViewModel.cs içinden **genre (tür)** bilgisi kod satırı silinsin.



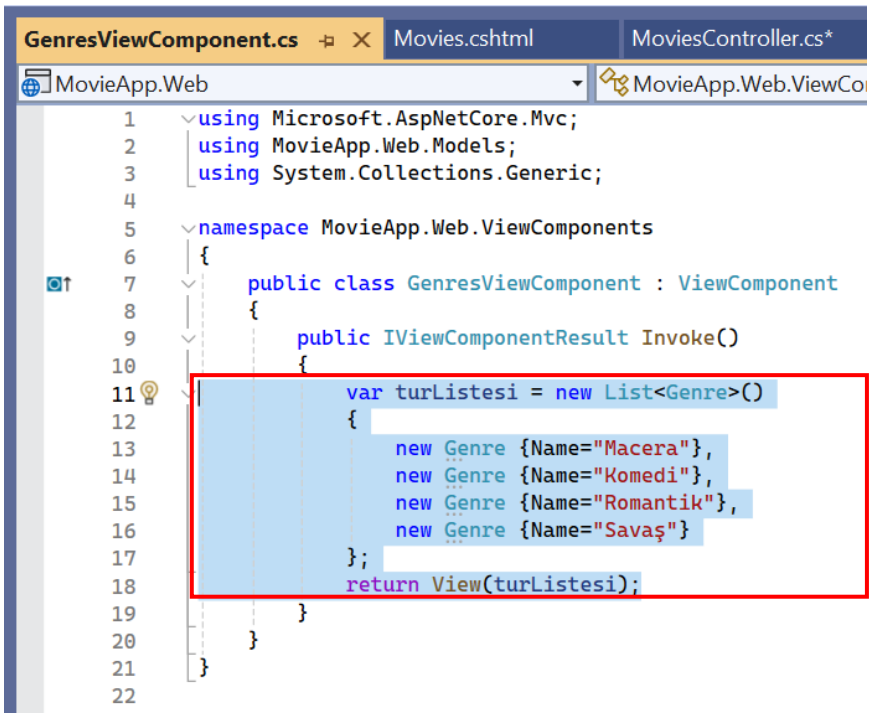
ADIM 16: MoviesViewModel.cs son görünümü.



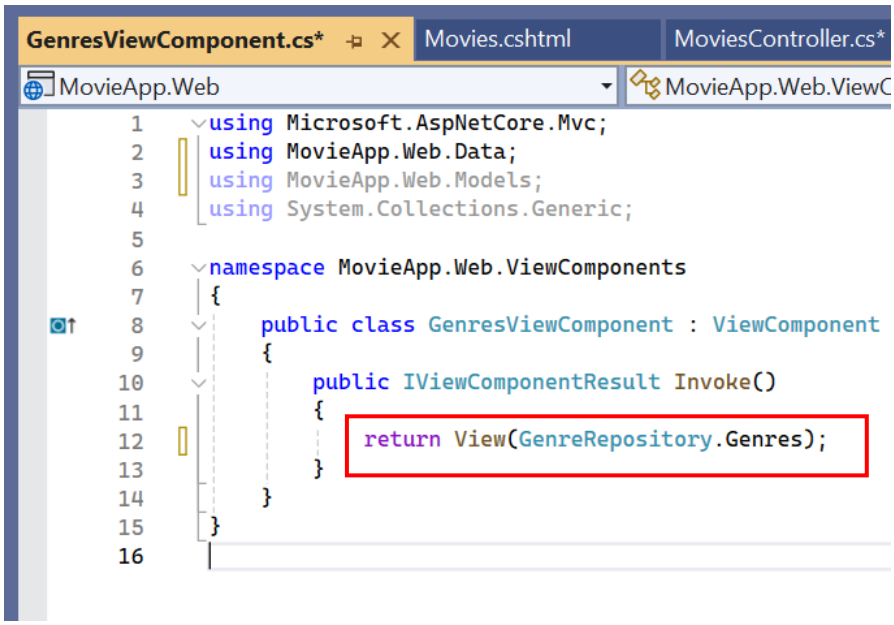
ADIM 17: MoviesController.cs içindeki kod bloğu düzenlenir.



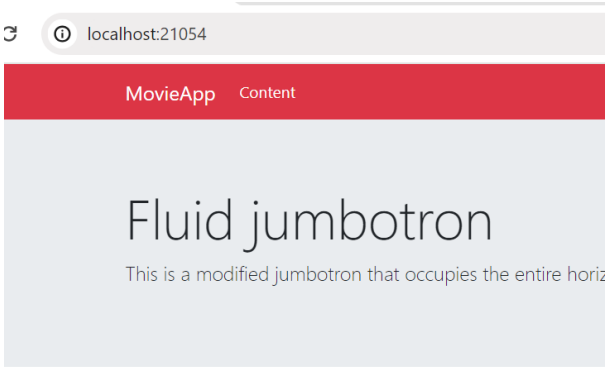
ADIM 18: GenresViewComponent.cs kod bloğu silinir.

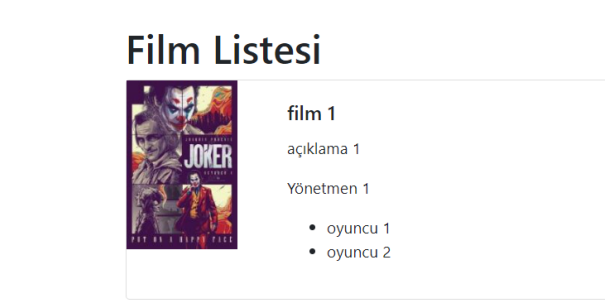


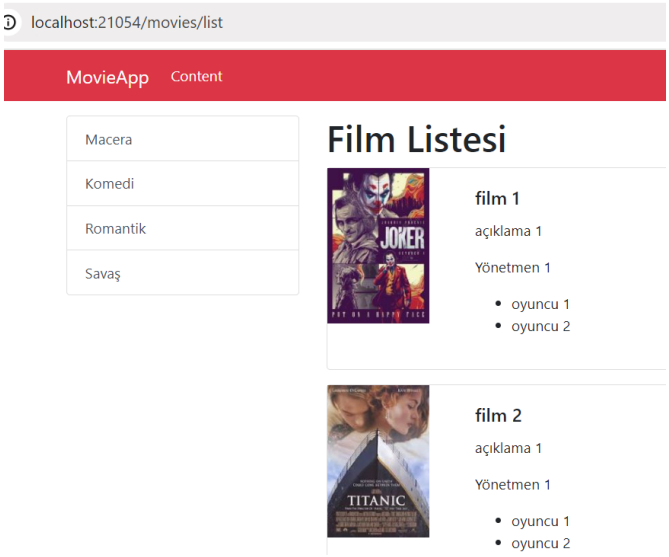
ADIM 19: Kod satırı eklenir.



ADIM 20: Çalıştırılır anasayfa ve Movie/list kontrol edilir.



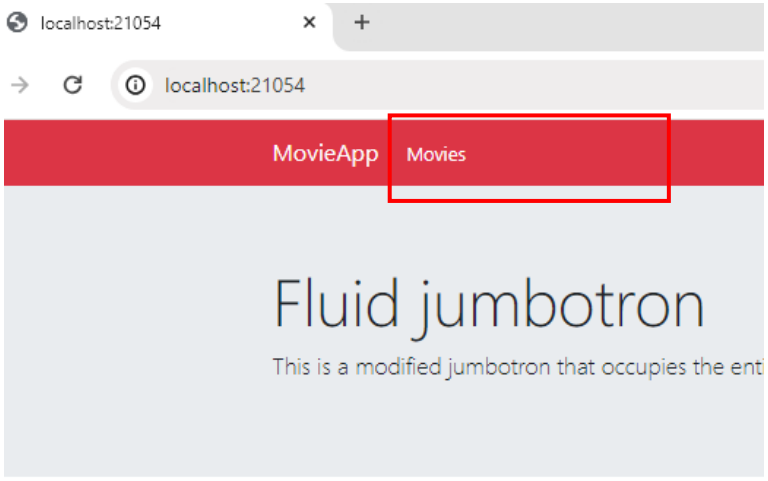




ADIM 21: _navbar.cshtml dosyası içinde html kodlarında değişiklikler yaparak görünümü düzenleyelim.

1- Link eklemek için **#** işareti **/** ile değiştirilir.
2- Movies/list e giden **Content** link ismi **Movies** olarak değiştirlir.

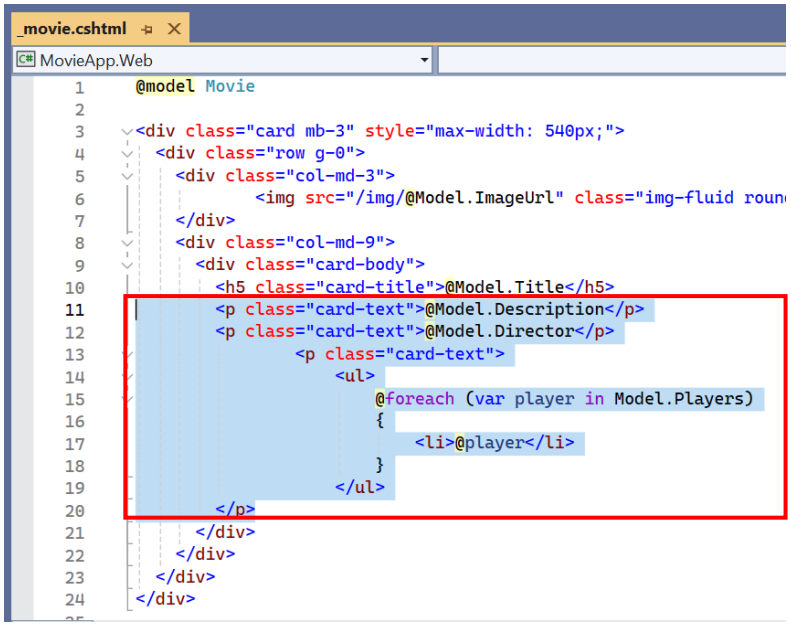
ADIM 22: Çalıştırılarak link kontrol edilir.



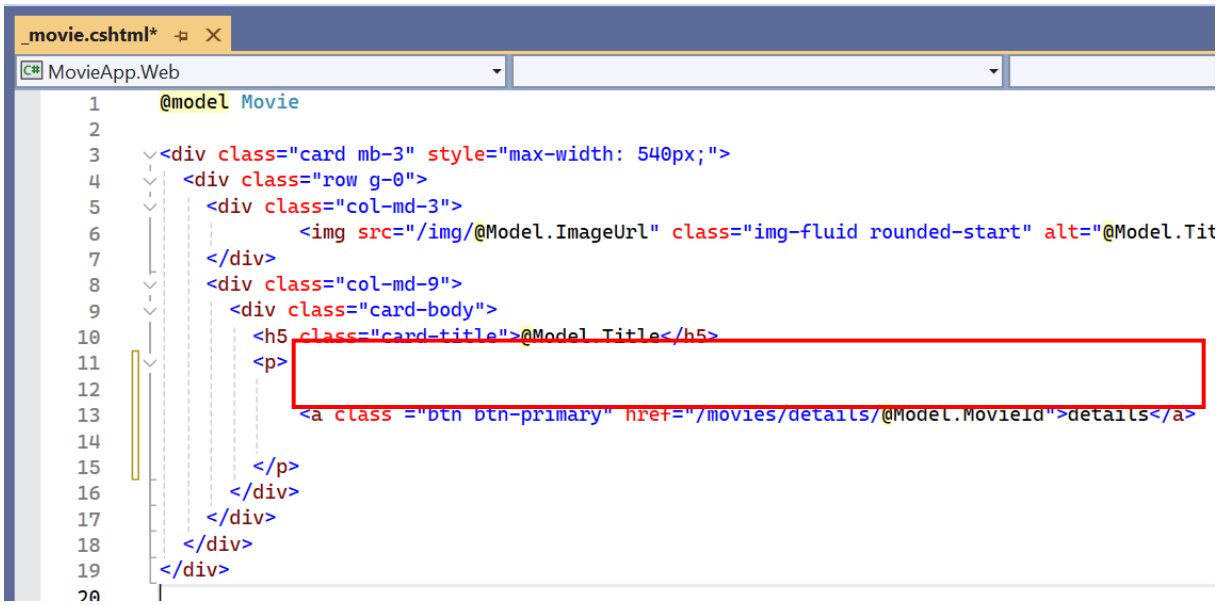
2. DETAY SAYFASI

Bu bölümde her bir filme bir link ekleyelim. Linke tıkladığımızda o filme ilişkin detay bilgilere ulaşalım. Film başlığı dışındaki tüm bilgiler detay sayfasında yer alsın.

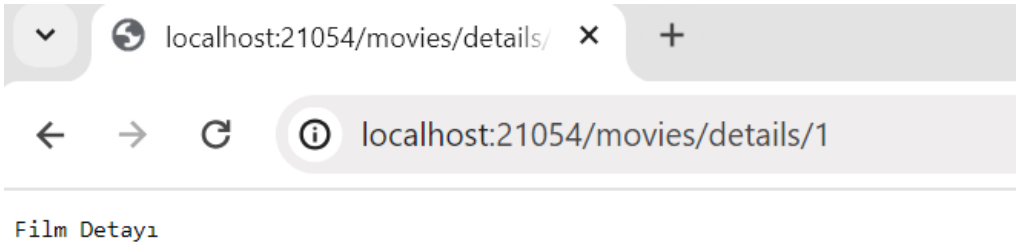
ADIM 1: Components klasörü içindeki **_movie.cshtml** dosyasında filmin başlığı dışındaki bilgileri silelim



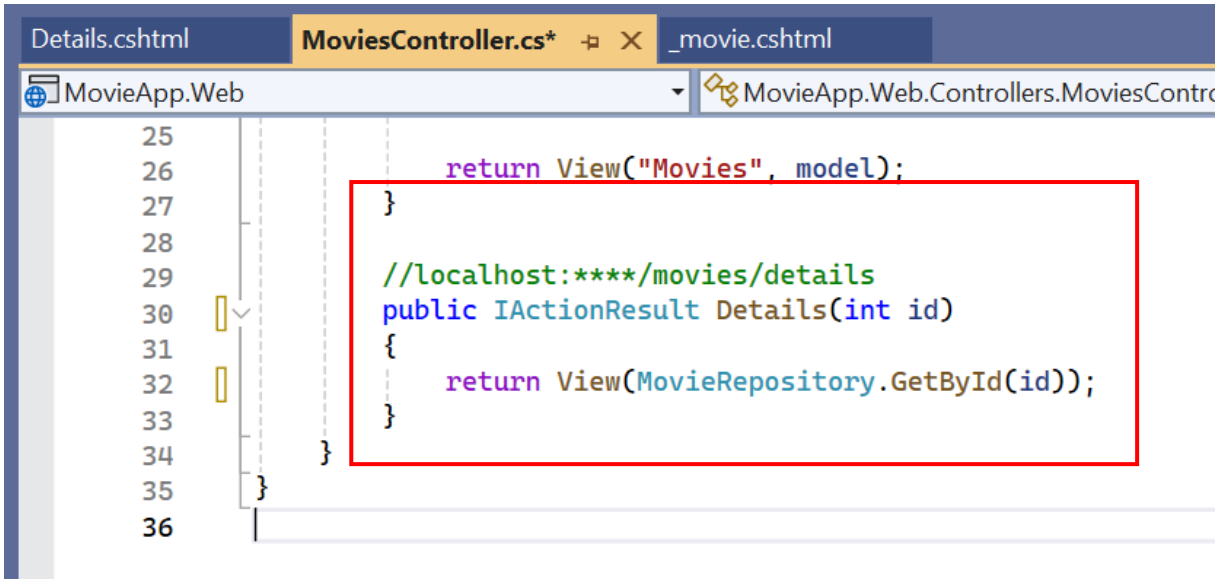
ADIM 2: Bizi film detay sayfasına götürecek bir link oluşturulur. İlgili html kod satırı ile.



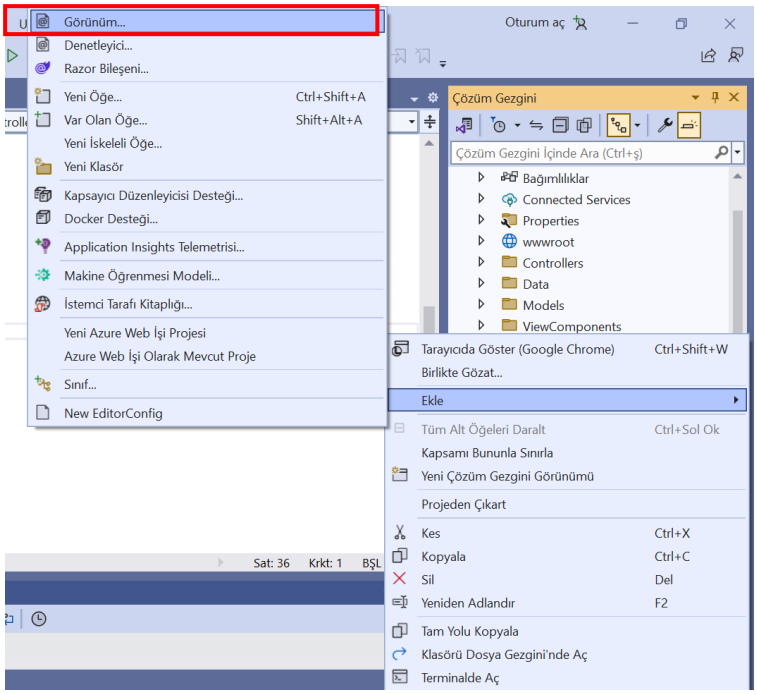
ADIM 3: Çalıştırılır kontrol edilir. Butona tıkladığımızda açılan sayfa MoviesController altındaki string değer tanımladığımız sayfa.



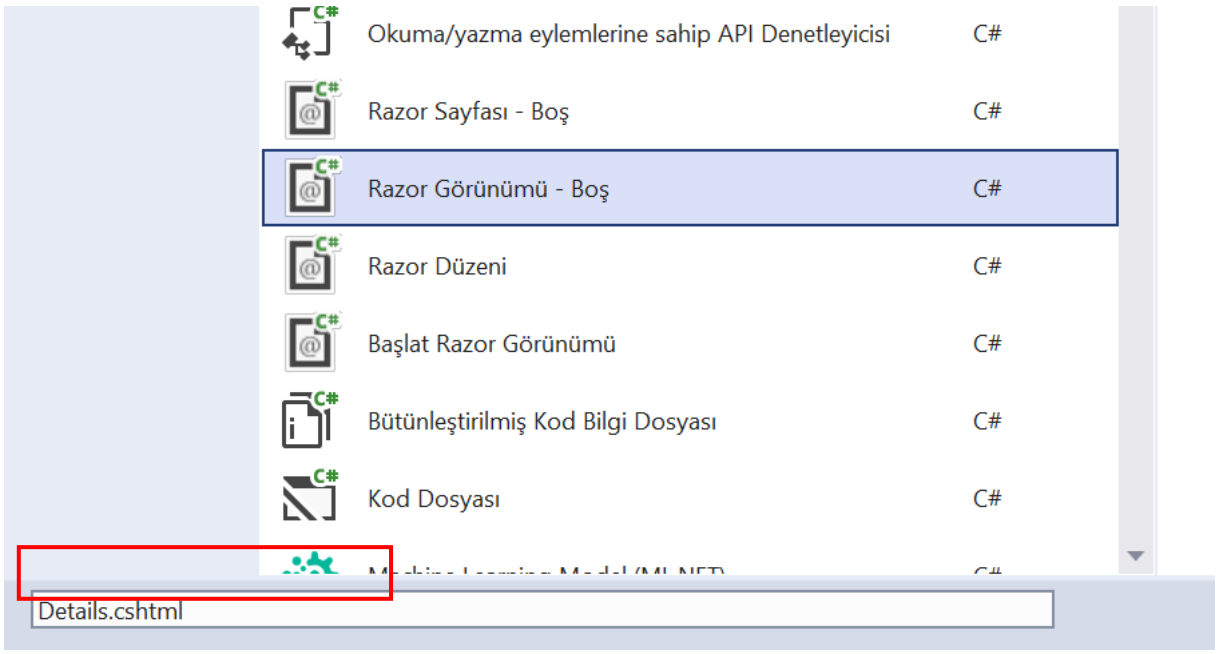
ADIM 4: MoviesController.cs altındaki kod bloğundaki değişiklik yapılır.



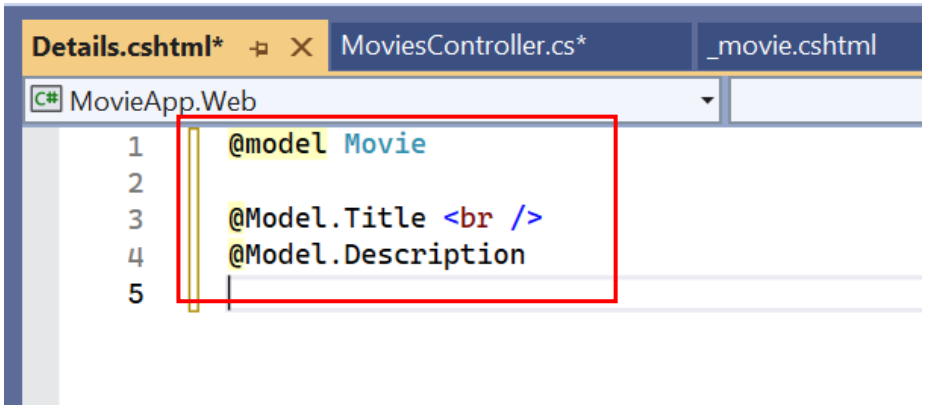
ADIM 5: Bir önceki aşamada yer alan string değeri bir Razor View oluşturarak değiştirelim. Razor View oluşturmak için; **Movies** klasörüne sağ tıkla Görünüm (View) seçilir.



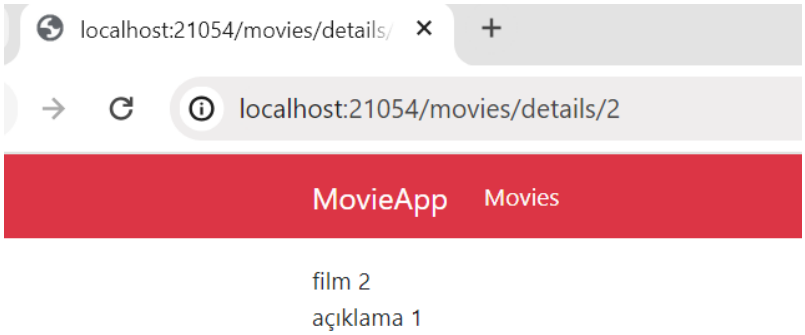
ADIM 6: Details.cshtml olarak isimlendirilir.



ADIM 7: Details.cshtml içerisine ilgili kod bloğu yazılır.



ADIM 8: Uygulama çalıştırılarak kontrol edilir. Örneğin film 2 ye tıkladığımızda film 2 nin detay sayfasına erişebiliyor muyuz?



ADIM 9: Details.cshtml içerisindeki Title ve Description satırlarını silelim ve html kodlarını görünümü daha güzel hale getirecek şekilde düzenleyelim.

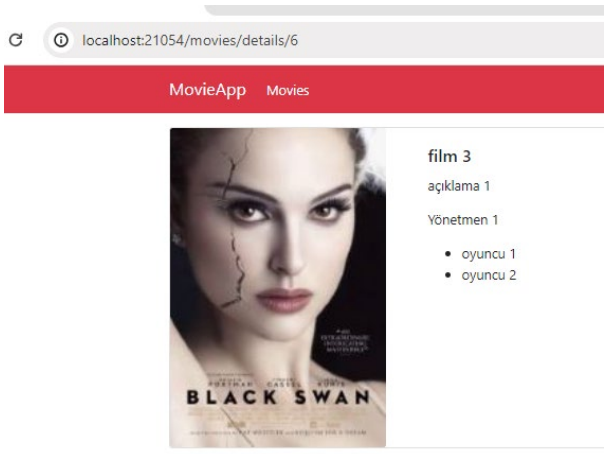
Details.cshtml* | MoviesController.cs | _movie.cshtml

MovieApp.Web

```
@model Movie

<div class="card">
    <div class="row">
        <div class="col-md-3">
            <img src="/img/@Model.ImageUrl" class="card-img"/>
        </div>
        <div class="col-md-9">
            <div class="card-body">
                <h5 class="card-title">@Model.Title</h5>
                <p class="card-text">@Model.Description</p>
                <p class="card-text">@Model.Director</p>
                <p>
                    <ul>
                        @foreach (var item in Model.Players)
                        {
                            <li>@item</li>
                        }
                    </ul>
                </p>
            </div>
        </div>
    </div>
</div>
```

ADIM 9: Anasayfada farklı filmlerin detay butonuna tıklayarak açılan film detay sayfalarını kontrol edelim. Örneğin film 6.



Daha önce oluşturmuş olduğumuz Film Türlerine göre Film Bilgilerini filtrelemek için yapılması gereken adımlar. Neler olabilir?

Hatırlayalım Filmler için **film bilgileri (Movie.cs)** ve **tür bilgileri (Genre.cs)** olmak üzere iki ayrı sınıfımız vardı.

Seçtiğimiz türe ait film bilgilerini nasıl listeleyebiliriz?

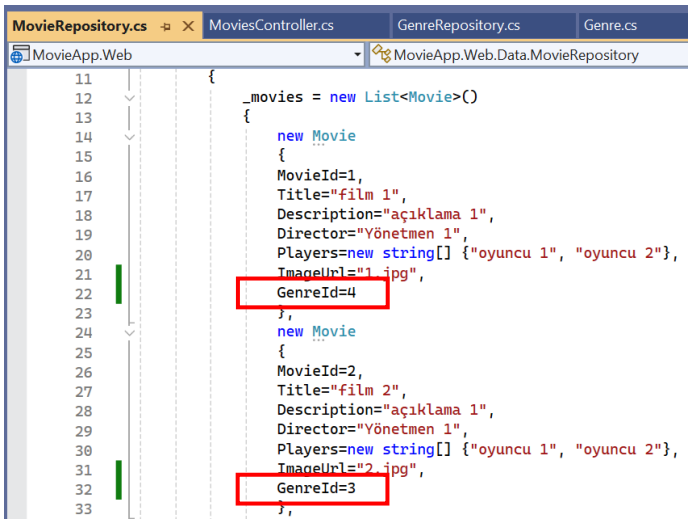
Her filmin yalnızca bir türe ait olduğunu varsayarsak. Adımlarımız şu şekilde ilerler;

3. TÜRE GÖRE FİLMLERİN FİLTRELENMESİ

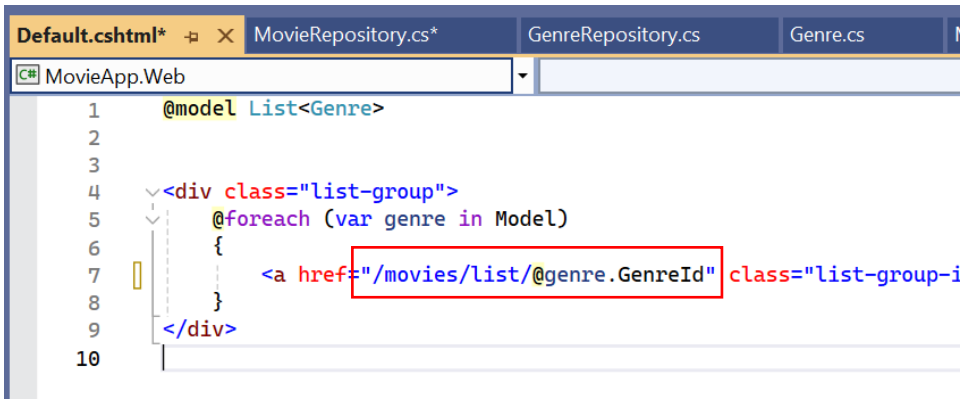
ADIM 1: Genre.cs dosyasındaki GenreId bilgisi Movie.cs dosyasına eklenir.

ADIM 2: GenreRepository.cs dosyası içerisinde daha önce film tür bilgilerine GenreId atamıştık. Oradaki Id bilgilerine göre; **MovieRepository.cs** dosyasındaki filmlerin her birine tür id leri (GenreId) eklenir.

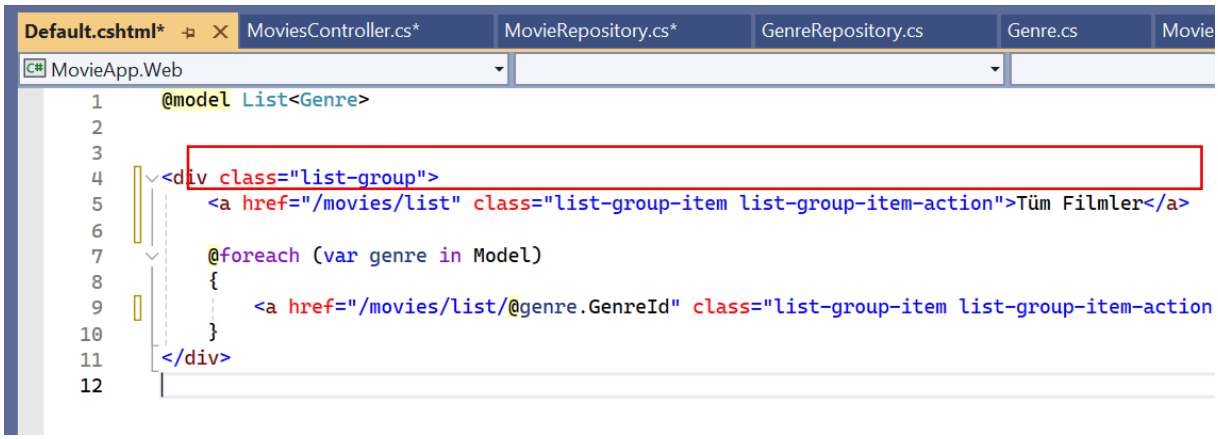
NOT: Örneğin id si 2 olan komedi türünü herhangi bir filmle eşleştirmeyelim ve sayfanın boş geldiğini kontrol edelim



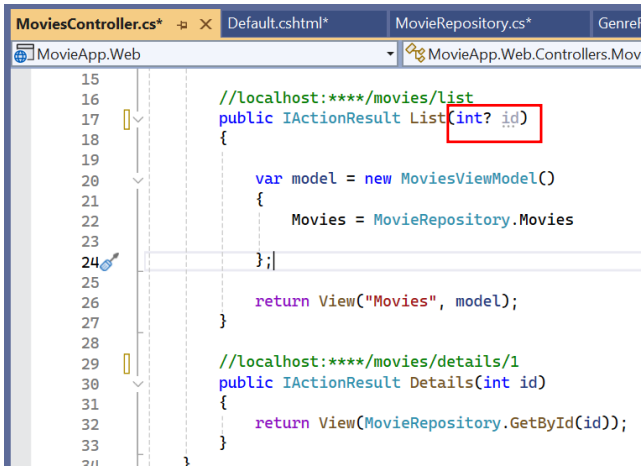
ADIM 3: Components Klasörü içindeki Genres klasörü içerisinde daha önce oluşturmuş olduğumuz **Default.cshtml** dosyasına ilgili değişiklikler yapılır.



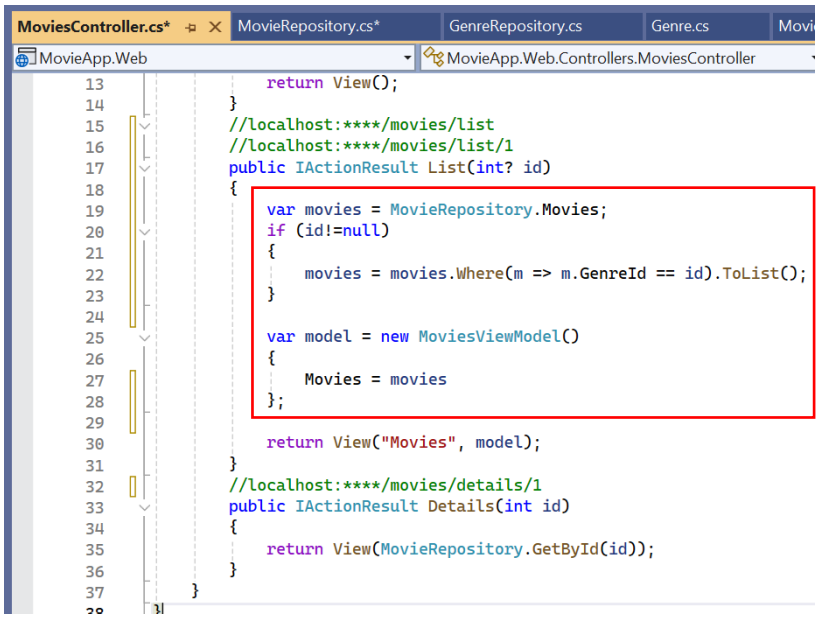
ADIM 4: Kategori filtrelemesi olmadan da url in düzgün çalışabilmesi için **Default.cshtml** içine aşağıdaki kod satırı eklenir.



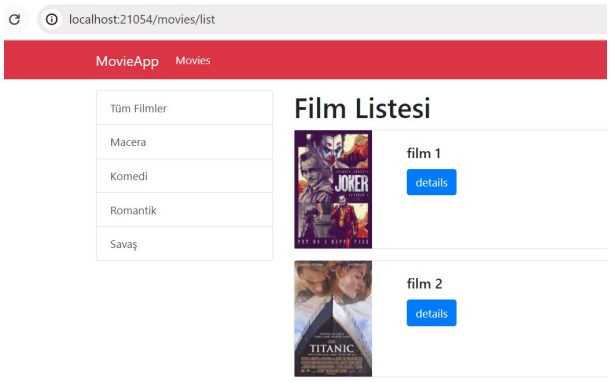
ADIM 5: Tür bilgilerinin id lerine göre url i düzenleyebilmek için **MoviesController.cs** dosyasında ilgili değişiklik yapılır. (**İnt? Ne işe yarar:** Eğer bir parametre göndermediysek **id=null** olsun)

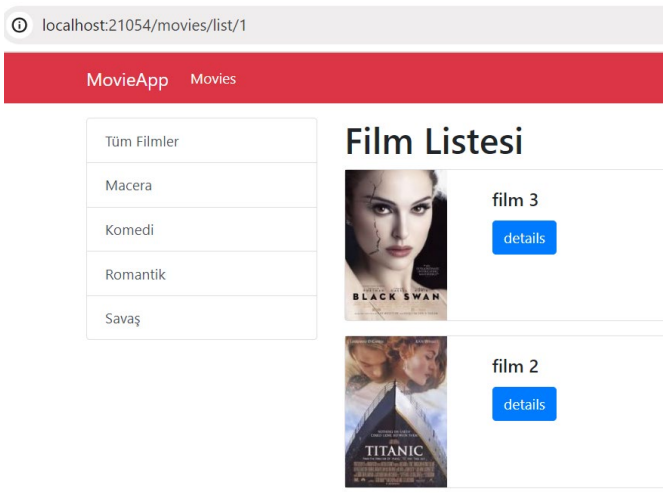


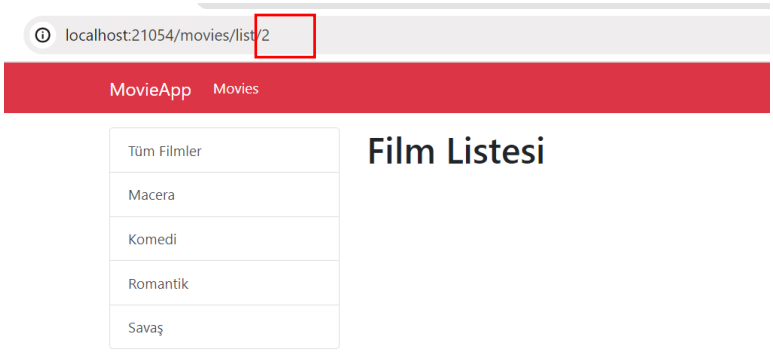
ADIM 6: Tür bilgisine göre bir filtreleme yapılıp yapılmayacağı url e gelen id bilgisi ile kontrol edilmelidir. Bunun için **MoviesController.cs** dosyasına ilgili if kod bloğu yazılır ve şu şekilde düzenlenir.



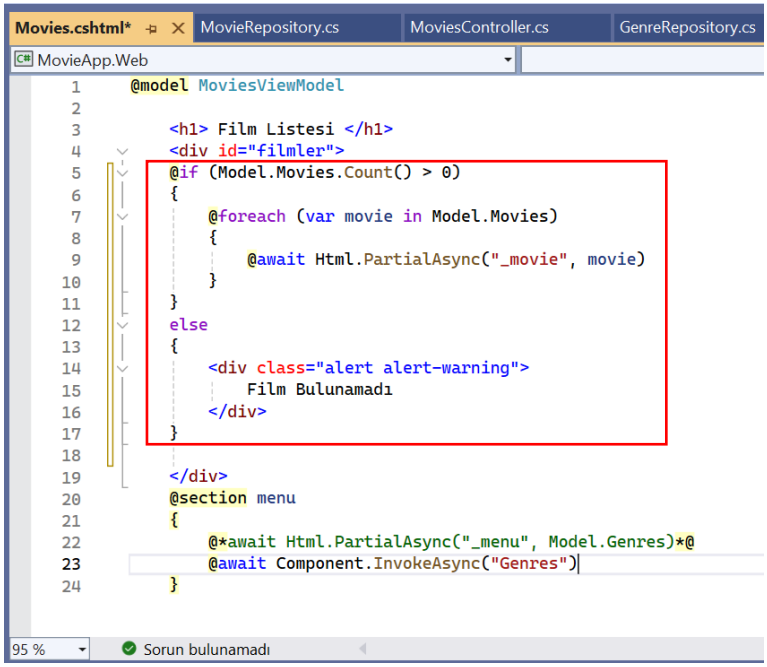
ADIM 7: Çalıştırılarak her bir tür kontrol edilir. Id si 2 olan Komedi türünün id sini herhangi bir filmle eşleştirmemiş ve boş bırakmıştık. Kontrol edelim.



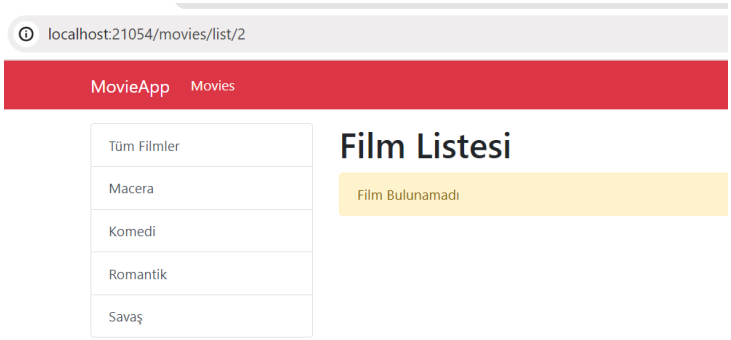




ADIM 8: Seçilen türde herhangi bir film olmaması durumunda sayfanın kullanıcıya boş gelmemesi ve bir uyarı vermesi için. Movies.cshtml dosyasında bir if bloğu oluşturarak kontrol edelim. Bunun için **Movies.cshtml** dosyasında ilgili kod bloğunu düzenleyelim.

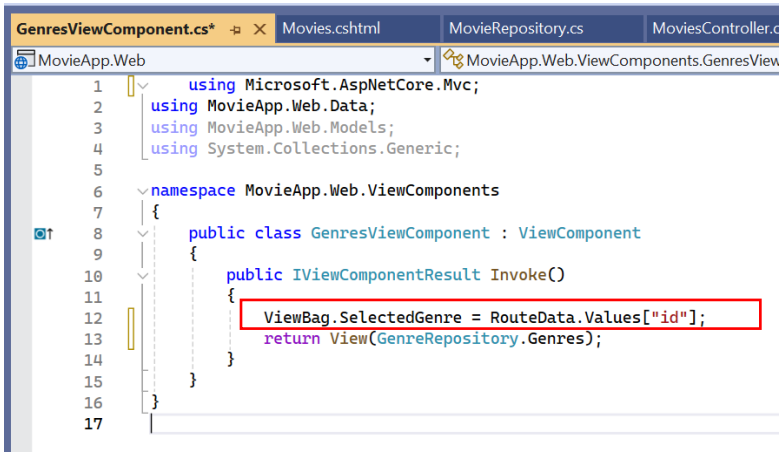


ADIM 9: Çalıştırarak boş olan sayfa kontrol edilir.

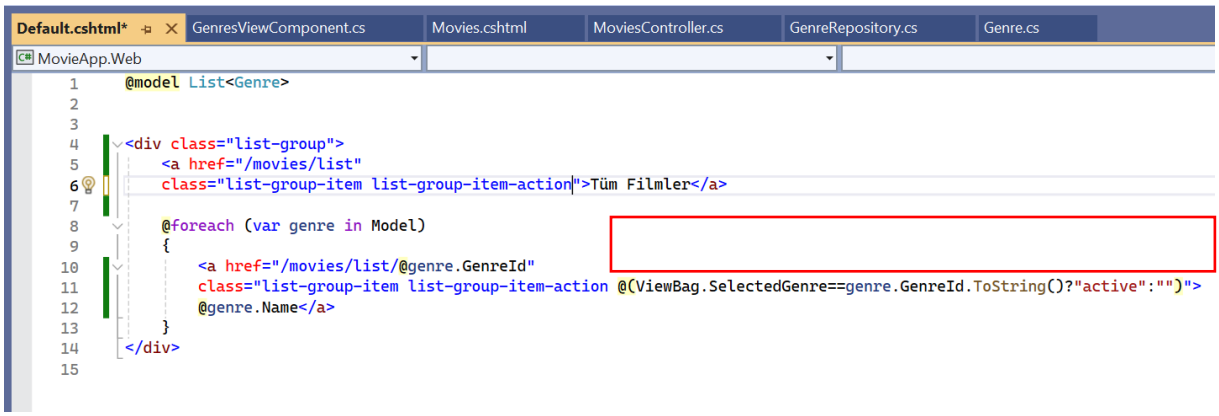


4. SEÇİLİ TÜR BİLGİSİNİN MENÜDE İŞARETLİ KALMASI

ADIM 1: GenresViewComponent.cs dosyasına id bilgisini tutmaya yarayan ilgili kod satırı eklenir.

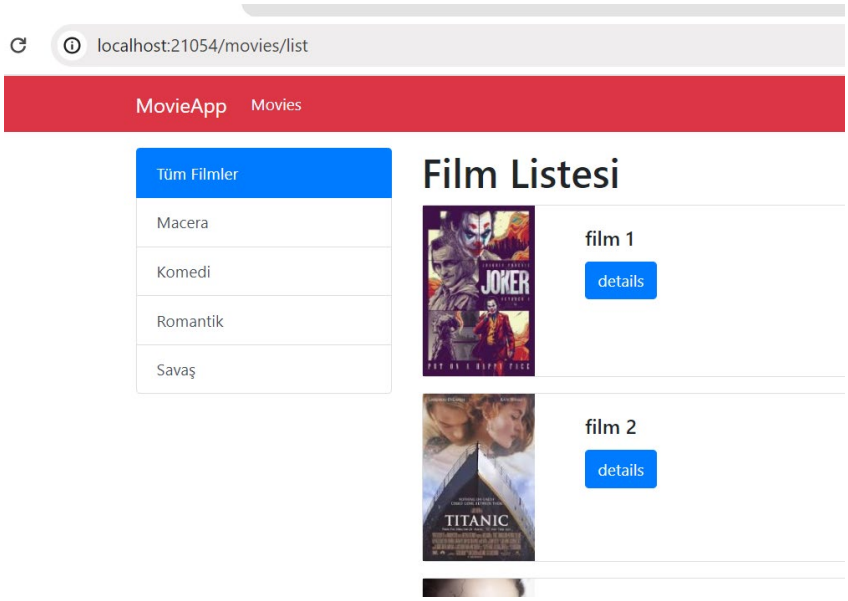


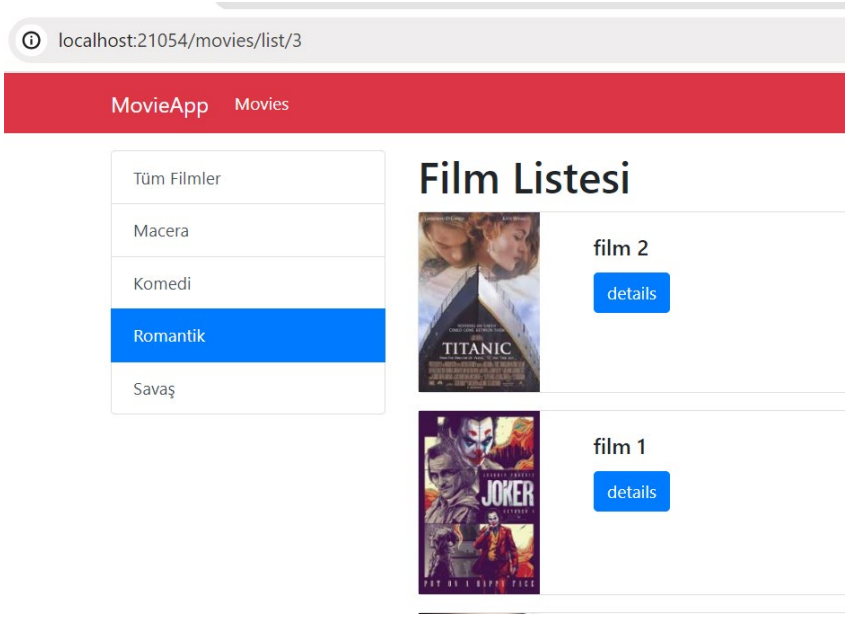
ADIM 2: Default.cshtml dosyasında ilgili değişiklikler şu şekildedir.



ADIM 3: Benzer kod satırı tüm filmler seçili olduğu duruma da eklenir.

ADIM 4: Çalıştırarak kontrol edilir.





ÇALIŞMA SORUSU

Invoke metodunun uygulamada kullanım amacı nedir açıklayınız.