



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).

```

4
5 namespace MovieApp.Web.Data
6 {
7     public class MovieRepository
8     {
9         private static readonly List<Movie> _movies = null;
10        static MovieRepository()
11        {
12            _movies = new List<Movie>()
13            {
14                new Movie
15                {
16                    MovieId=1,
17                    Title="film 1",
18                    Description="açıklama 1",
19                    Director="Yönetmen 1",
20                    Players=new string[] {"oyuncu 1", "oyuncu 2"},
21                    ImageUrl="1.jpg"
22                },
23                new Movie
24                {
25                    MovieId=2,
26                    Title="film 2",
27                    Description="açıklama 1",
28                    Director="Yönetmen 1",
29                    Players=new string[] {"oyuncu 1", "oyuncu 2"},
30                    ImageUrl="2.jpg"
31                },
32                new Movie
33                {
34                    MovieId=3,
35                    Title="film 3",
36                    Description="açıklama 1",
37                    Director="Yönetmen 1",
38                    Players=new string[] {"oyuncu 1", "oyuncu 2"},
39                    ImageUrl="3.jpg"
40                },
41                new Movie
42                {
43                    MovieId=1,
44                    Title="film 1",
45                    Description="açıklama 1",
46                    Director="Yönetmen 1",
47                    Players=new string[] {"oyuncu 1", "oyuncu 2"},
48                    ImageUrl="1.jpg"
49                },
50                new Movie
51                {
52                    MovieId=2,
53                    Title="film 2",
54                    Description="açıklama 1",
55                    Director="Yönetmen 1",
56                    Players=new string[] {"oyuncu 1", "oyuncu 2"},
57                    ImageUrl="2.jpg"
58                },
59                new Movie
60                {
61                    MovieId=3,
62                    Title="film 3",
63                    Description="açıklama 1",
64                    Director="Yönetmen 1",
65                    Players=new string[] {"oyuncu 1", "oyuncu 2"},
66                    ImageUrl="3.jpg"
67                }
68            }
69        }
70    }
71 }

```

54 % Sorun bulunamadı

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.

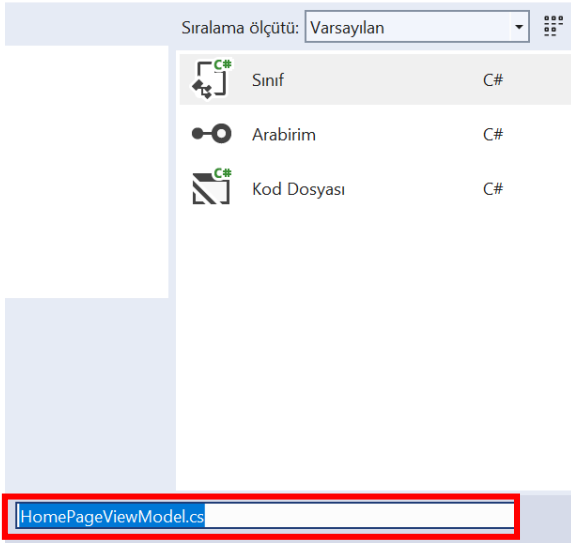
ADIM 4: Silme işleminden sonra **HomeController** görünümü aşağıdaki gibi olmalı.

```

1  using Microsoft.AspNetCore.Mvc;
2  using MovieApp.Web.Models;
3  using System.Collections.Generic;
4
5  namespace MovieApp.Web.Controllers
6  {
7      public class HomeController : Controller
8      {
9          public IActionResult Index()
10         {
11             return View();
12         }
13         public IActionResult About()
14         {
15             return View();
16         }
17     }
18 }

```

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. (Popüler filmler gibi) **HomePageViewModel.cs** sayfasına ilgili kod bloğunu şu şekilde ekleriz.

```

1  using System.Collections.Generic;
2
3  namespace MovieApp.Web.Models
4  {
5      public class HomePageViewModel
6      {
7          public List<Movie> PopularMovies { get; set; }
8      }
9  }

```

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

NOT: Popüler filmlerin bilgileri MovieRepository üzerinden gelecek şekilde düzenlendi.

```
4 using System.Collections.Generic;
5
6 namespace MovieApp.Web.Controllers
7 {
8     public class HomeController : Controller
9     {
10         public IActionResult Index()
11         {
12             var model = new HomePageViewModel
13             {
14                 PopularMovies = MovieRepository.Movies
15             };
16
17             return View(model);
18         }
19         public IActionResult About()
20         {
21             return View();
22         }
23     }
24 }
25
26
```

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

```

1  using Microsoft.AspNetCore.Mvc;
2  using MovieApp.Web.Data;
3  using MovieApp.Web.Models;
4  using System.Collections.Generic;
5
6  namespace MovieApp.Web.Controllers
7  {
8      public class HomeController : Controller
9      {
10         public IActionResult Index()
11         {
12             var model = new HomePageViewModel
13             {
14                 PopularMovies = MovieRepository.Movies
15             };
16
17             return View(model);
18         }
19         public IActionResult About()
20         {
21
22             return View();

```

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

```

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

```

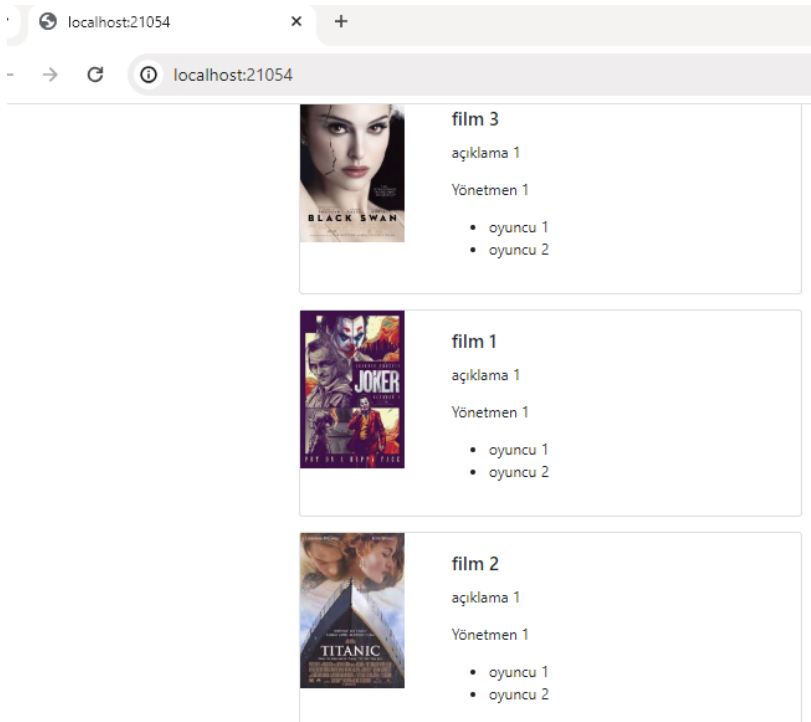
ADIM 10: **Index.cshtml** sayfasında döngü kod bloğu aşağıdaki gibi düzenlenir.

```

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 </div>
20
21

```

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



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

```

12         return View();
13     }
14
15     //localhost:****/movies/list
16     public IActionResult List()
17     {
18         var filmListesi = new List<Movie>()
19         {
20             new Movie
21             {
22                 Title="film 1",
23                 Description="açıklama 1",
24                 Director="Yönetmen 1",
25                 Players=new string[] { "oyuncu 1", "oyuncu 2"},
26                 ImageUrl="1.jpg"
27             },
28             new Movie
29             {
30                 Title="film 2",
31                 Description="açıklama 1",
32                 Director="Yönetmen 1",
33                 Players=new string[] { "oyuncu 1", "oyuncu 2"},
34                 ImageUrl="2.jpg"
35             },
36             new Movie
37             {
38                 Title="film 3",
39                 Description="açıklama 1",
40                 Director="Yönetmen 1",
41                 Players=new string[] { "oyuncu 1", "oyuncu 2"},
42                 ImageUrl="3.jpg"
43             }
44         };
45
46
47         var model = new MovieGenreViewModel()
48         {
49             Movies = filmListesi
50         }

```

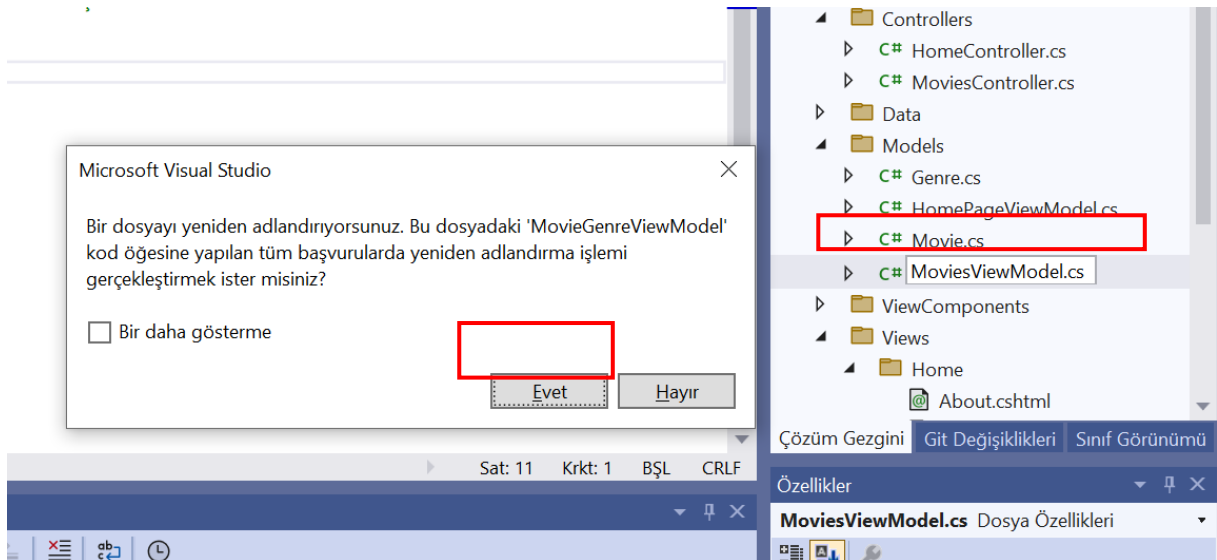
ADIM 13: Yine MoviesController.cs içindeki **MovieGenreViewModel()**, **MoviesViewModel()** olarak değiştirilir.

```

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         return View("Movies", model);
25     }
26
27     //localhost:****/movies/details
28     public string Details()
29     {
30         return "Film Detayı";
31     }
32 }
33
34
35

```

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



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

```

1  using System.Collections.Generic;
2
3  namespace MovieApp.Web.Models
4  {
5      public class MoviesViewModel
6      {
7          public List<Movie> Movies { get; set; } //film bilgileri için
8          public List<Genre> Genres { get; set; } //film türleri için
9      }
10 }
11
12

```

ADIM 16: MoviesViewModel.cs son görünümü.

```

1  using System.Collections.Generic;
2
3  namespace MovieApp.Web.Models
4  {
5      public class MoviesViewModel
6      {
7          public List<Movie> Movies { get; set; } //film bilgileri için
8      }
9  }
10
11

```

ADIM 17: MoviesController.cs içindeki kod bloğu düzenlenir.

```

7  {
8      public class MoviesController : Controller
9      {
10
11          public IActionResult Index()
12          {
13              return View();
14          }
15
16          //localhost:****/movies/list
17          public IActionResult List()
18          {
19
20              var model = new MoviesViewModel()
21              {
22                  Movies = MovieRepository.Movies
23              };
24          }
25

```

ADIM 18: GenresViewComponent.cs kod bloğu silinir.

```

1  using Microsoft.AspNetCore.Mvc;
2  using MovieApp.Web.Models;
3  using System.Collections.Generic;
4
5  namespace MovieApp.Web.ViewComponents
6  {
7      public class GenresViewComponent : ViewComponent
8      {
9          public IViewComponentResult Invoke()
10         {
11             var turListesi = new List<Genre>()
12             {
13                 new Genre {Name="Macera"},
14                 new Genre {Name="Komedi"},
15                 new Genre {Name="Romantik"},
16                 new Genre {Name="Savaş"}
17             };
18             return View(turListesi);
19         }
20     }
21 }
22

```

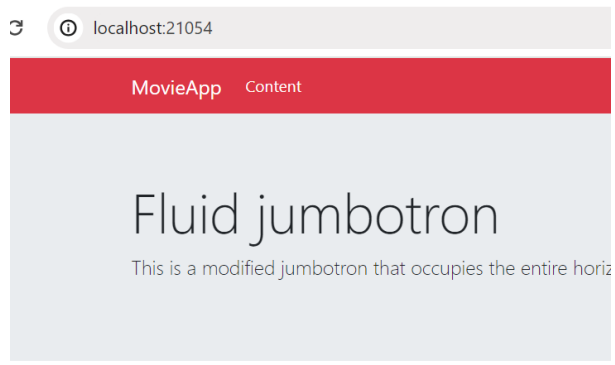
ADIM 19: Kod satırı eklenir.

```

1  using Microsoft.AspNetCore.Mvc;
2  using MovieApp.Web.Data;
3  using MovieApp.Web.Models;
4  using System.Collections.Generic;
5
6  namespace MovieApp.Web.ViewComponents
7  {
8      public class GenresViewComponent : ViewComponent
9      {
10         public IViewComponentResult Invoke()
11         {
12             return View(GenreRepository.Genres);
13         }
14     }
15 }
16

```

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



Film Listesi



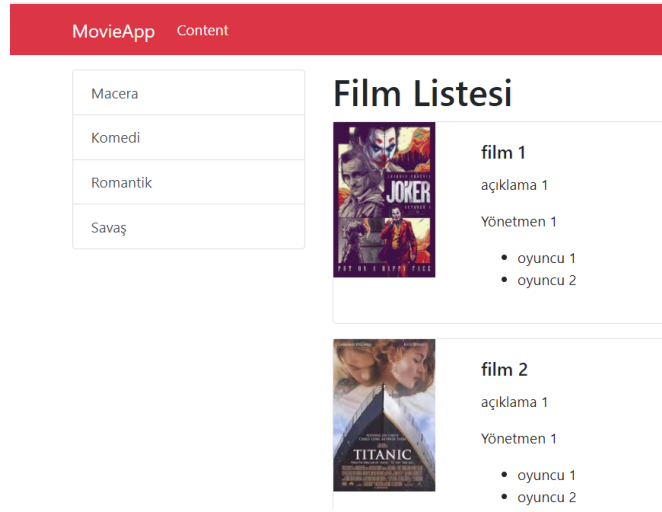
film 1

açıklama 1

Yönetmen 1

- oyuncu 1
- oyuncu 2

localhost:21054/movies/list



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

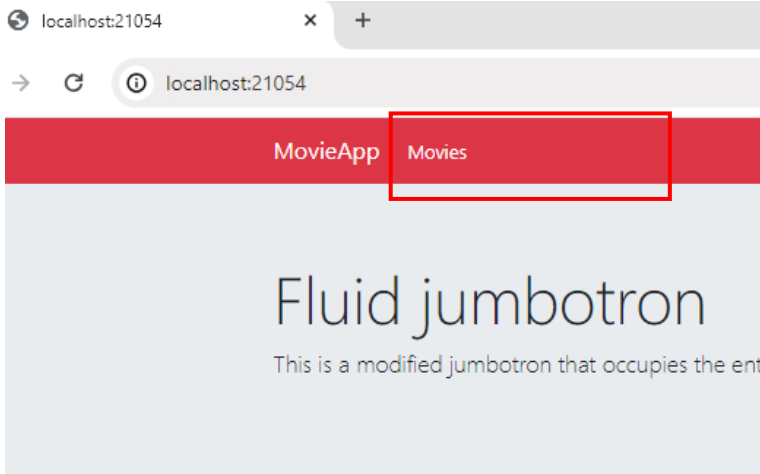
- 1- Link eklemek için # işareti / ile değiştirilir.
- 2- Movies/list e giden Content link ismi Movies olarak değiştirilir.

```

1 <nav class="navbar navbar-expand-lg navbar-dark bg-danger">
2   <div class="container">
3     <button class="navbar-toggler" type="button" data-toggle="collapse"
4       <span class="navbar-toggler-icon"></span>
5     </button>
6     <div class="collapse navbar-collapse" id="navbarToggler">
7       <a href="/" class="navbar-brand">MovieApp</a>
8       <ul class="navbar-nav mr-auto">
9         <li class="navbar-item active">
10          <a href="/movies/list" class="nav-link">Movies</a>
11        </li>
12      </ul>
13    </div>
14  </div>
15 </nav>
16

```

ADIM 22: Çalıştırılarak link kontrol edilir.



Film Listesi



film 1

açıklama 1

Yönetmen 1

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

```

1  @model Movie
2
3  <div class="card mb-3" style="max-width: 540px;">
4    <div class="row g-0">
5      <div class="col-md-3">
6        
8      <div class="col-md-9">
9        <div class="card-body">
10         <h5 class="card-title">@Model.Title</h5>
11         <p class="card-text">@Model.Description</p>
12         <p class="card-text">@Model.Director</p>
13         <p class="card-text">
14           <ul>
15             @foreach (var player in Model.Players)
16             {
17               <li>@player</li>
18             }
19           </ul>
20         </p>
21       </div>
22     </div>
23   </div>
24 </div>

```

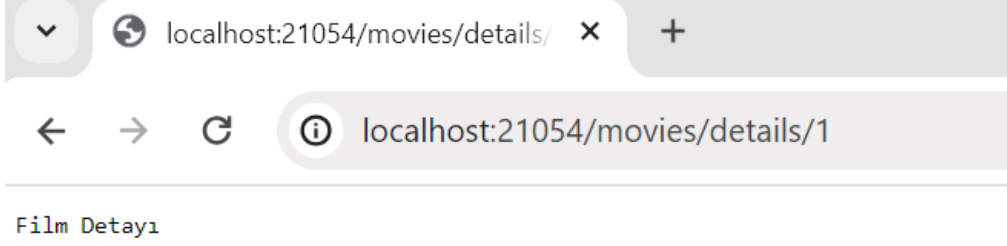
ADIM 2: Bizi film detay sayfasına götürecek bir link oluşturulur. İlgili html kod satırı ile.

```

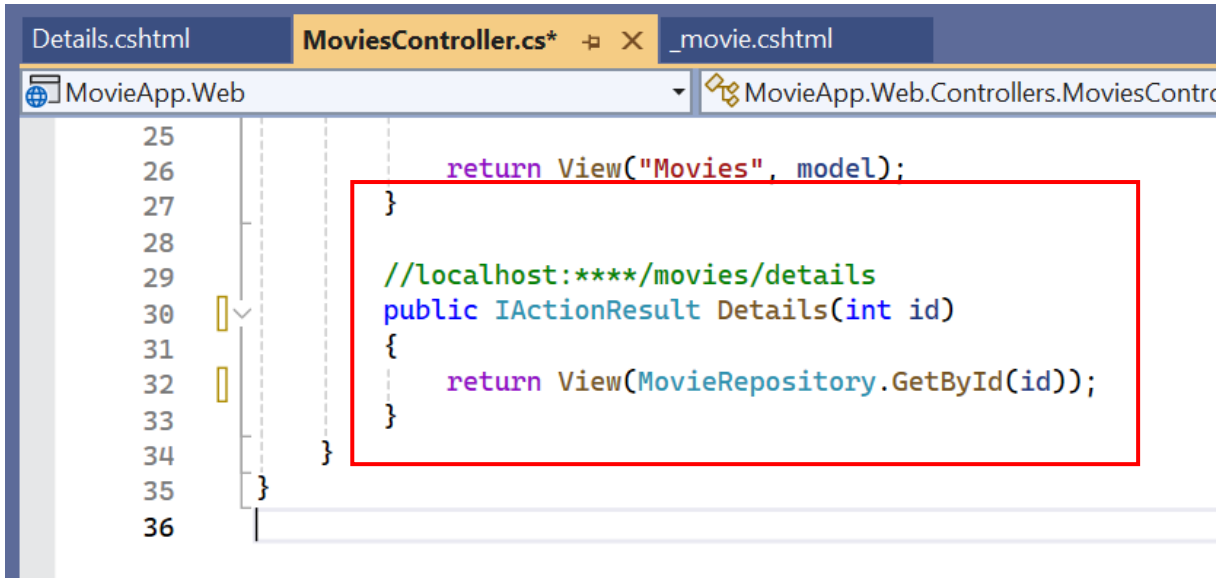
1  @model Movie
2
3  <div class="card mb-3" style="max-width: 540px;">
4    <div class="row g-0">
5      <div class="col-md-3">
6        
8      <div class="col-md-9">
9        <div class="card-body">
10         <h5 class="card-title">@Model.Title</h5>
11         <p>
12           <a class="btn btn-primary" href="/movies/details/@Model.MovieId">details</a>
13         </p>
14       </div>
15     </div>
16   </div>
17 </div>
18 </div>
19 </div>
20

```

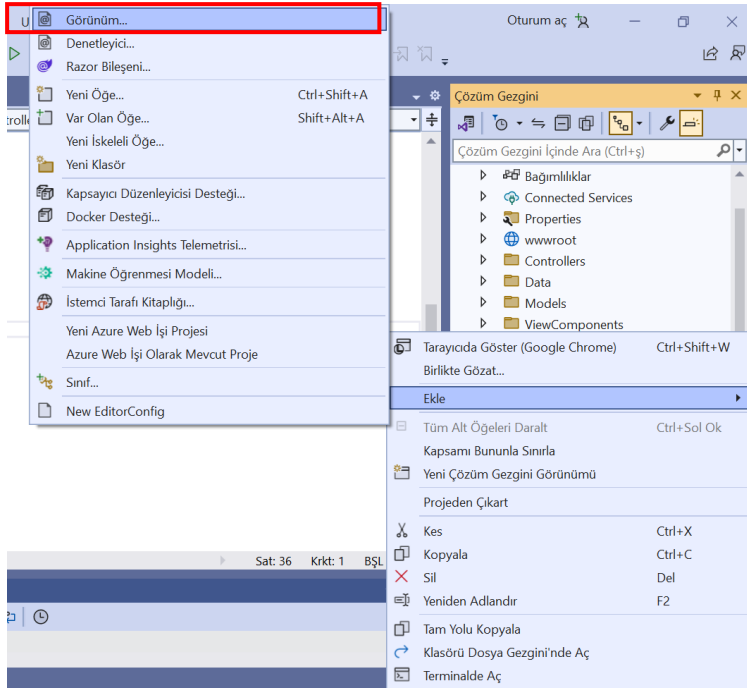
ADIM 3: Çalıştırılır kontrol edilir. Butona tıkladığımızda açılan sayfa MoviesController altındaki string değer tanımladığımız sayfa.



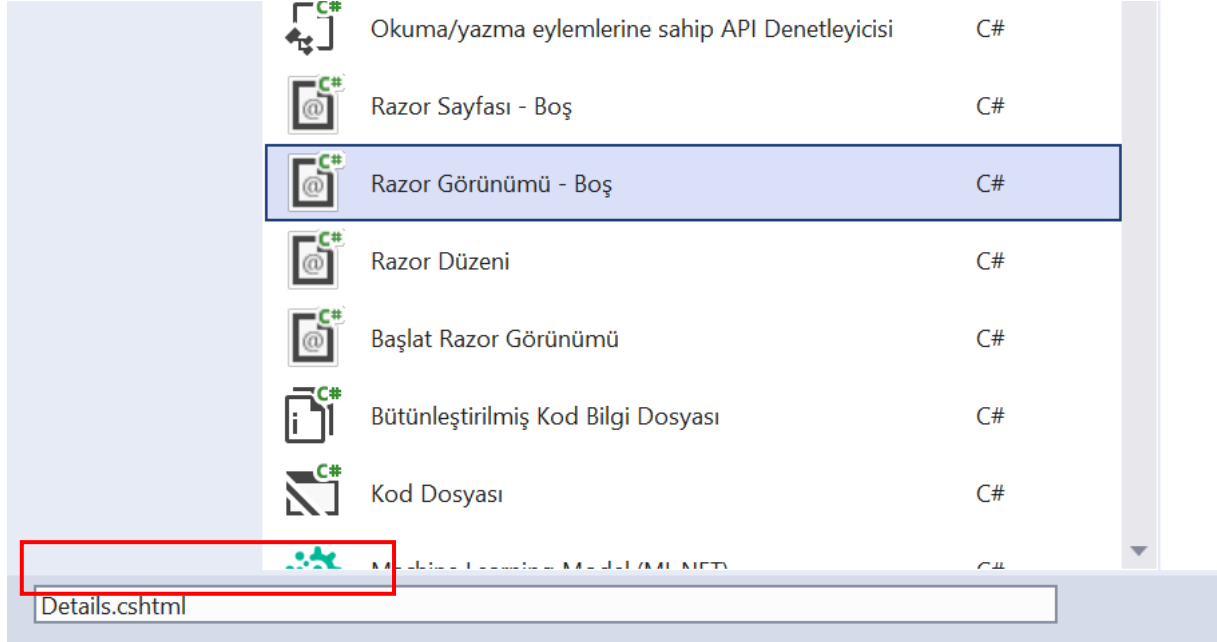
ADIM 4: MoviesController.cs altındaki kod bloğundaki değişiklik yapılır.



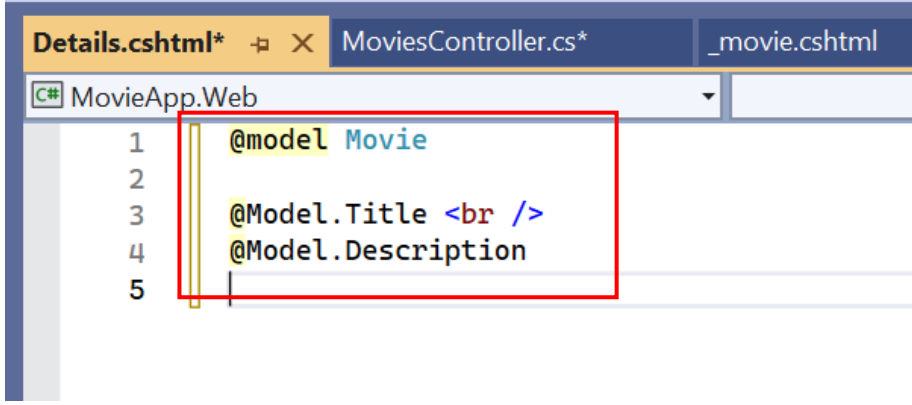
ADIM 5: Bir önceki aşamada yer alan string değeri bir Razor View oluşturarak değiştirelim. Razor View oluşturmak için; **Movies** klasörüne sağ tıkla Görünüm (View) seçilir.



ADIM 6: Details.cshtml olarak isimlendirilir.

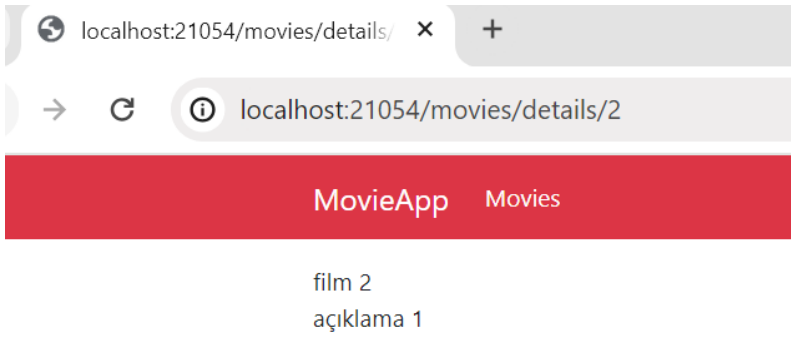


ADIM 7: Details.cshtml içerisine ilgili kod bloğu yazılır.



```
1 @model Movie
2
3 @Model.Title <br />
4 @Model.Description
5
```

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



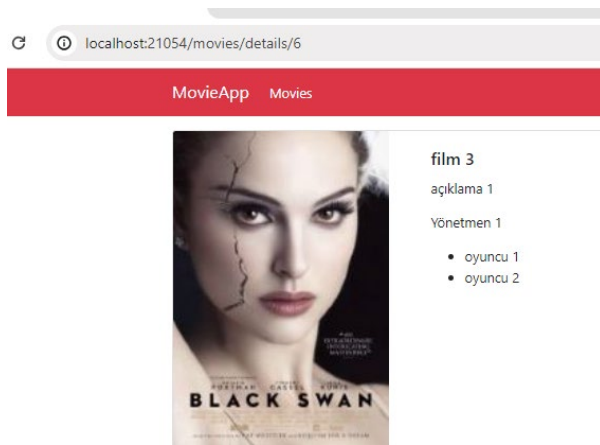
ADIM 9: Details.cshtml içerisindeki Title ve Description satırlarını silelim ve html kodlarını görünümü daha güzel hale getirecek şekilde düzenleyelim.

```

Details.cshtml* MoviesController.cs _movie.cshtml
C# MovieApp.Web
1 @model Movie
2
3 <div class="card">
4     <div class="row">
5         <div class="col-md-3">
6             
7         </div>
8         <div class="col-md-9">
9             <div class="card-body">
10                <h5 class="card-title">@Model.Title</h5>
11                <p class="card-text">@Model.Description</p>
12                <p class="card-text">@Model.Director</p>
13                <p>
14                    <ul>
15                        @foreach (var item in Model.Players)
16                        {
17                            <li>@item</li>
18                        }
19                    </ul>
20                </p>
21            </div>
22        </div>
23    </div>
24 </div>
25

```

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 FİMLERİN FİLTRELENMESİ

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

```

1 namespace MovieApp.Web.Models
2 {
3     public class Movie
4     {
5         public int MovieId { get; set; }
6         public string Title { get; set; }
7         public string Description { get; set; }
8         public string Director { get; set; }
9         public string [] Players { get; set; }
10        public string ImageUrl { get; set; }
11        public int GenreId { get; set; }
12    }
13 }
14

```

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

```

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33

```

```

{
    _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",
            GenreId=4
        },
        new Movie
        {
            MovieId=2,
            Title="film 2",
            Description="açıklama 1",
            Director="Yönetmen 1",
            Players=new string[] { "oyuncu 1", "oyuncu 2"},
            ImageUrl="2.jpg",
            GenreId=3
        },
    }
}

```

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

```

Default.cshtml*  MovieRepository.cs*  GenreRepository.cs  Genre.cs
C# MovieApp.Web
1  @model List<Genre>
2
3
4  <div class="list-group">
5      @foreach (var genre in Model)
6      {
7          <a href="/movies/list/@genre.GenreId" class="list-group-item">
8      }
9  </div>
10

```

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

```

Default.cshtml*  MoviesController.cs*  MovieRepository.cs*  GenreRepository.cs  Genre.cs  Movie
C# MovieApp.Web
1  @model List<Genre>
2
3
4  <div class="list-group">
5      <a href="/movies/list" class="list-group-item list-group-item-action">Tüm Filmler</a>
6
7      @foreach (var genre in Model)
8      {
9          <a href="/movies/list/@genre.GenreId" class="list-group-item list-group-item-action"
10     }
11 </div>
12

```

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

```

MoviesController.cs*  Default.cshtml*  MovieRepository.cs*  Genre
C# MovieApp.Web  MovieApp.Web.Controllers.Mov
15
16 //localhost:****/movies/list
17 public IActionResult List(int? id)
18 {
19
20     var model = new MoviesViewModel()
21     {
22         Movies = MovieRepository.Movies
23     };
24
25     return View("Movies", model);
26 }
27
28 //localhost:****/movies/details/1
29 public IActionResult Details(int id)
30 {
31     return View(MovieRepository.GetById(id));
32 }
33
34

```

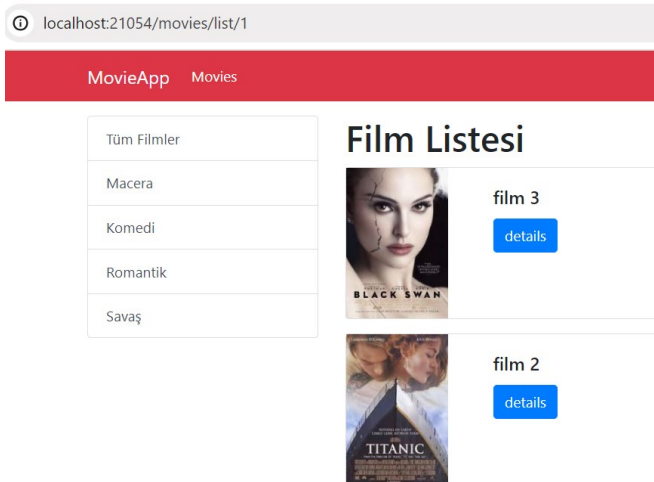
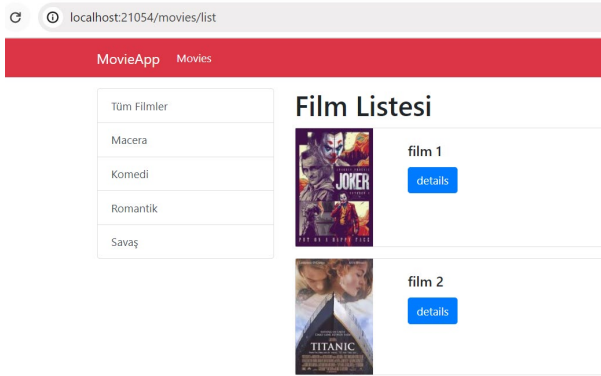
ADIM 6: Tür bilgisine göre bir filtreleme yapıp yapılmayacağı url e gelen id bilgisi ile kontrol edilmelidir. Bunun için **MoviesController.cs** dosyasına ilgili if kod bloğu yazılır ve şu şekilde düzenlenir.

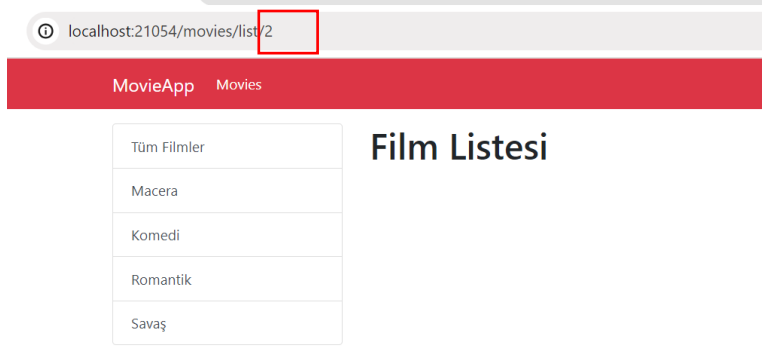
```

13         return View();
14     }
15     //localhost:****/movies/list
16     //localhost:****/movies/list/1
17     public IActionResult List(int? id)
18     {
19         var movies = MovieRepository.Movies;
20         if (id != null)
21         {
22             movies = movies.Where(m => m.GenreId == id).ToList();
23         }
24         var model = new MoviesViewModel()
25         {
26             Movies = movies
27         };
28         return View("Movies", model);
29     }
30     //localhost:****/movies/details/1
31     public IActionResult Details(int id)
32     {
33         return View(MovieRepository.GetById(id));
34     }
35 }

```

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.





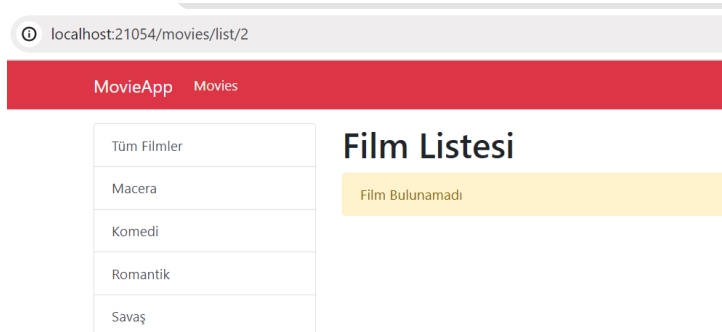
ADIM 8: Seçilen türde herhangi bir film olmaması durumunda sayfanın kullanıcıya boş gelmemesi ve bir uyarı vermesi için. Movies.cshtml dosyasında bir if bloğu oluşturarak kontrol edelim. Bunun için **Movies.cshtml** dosyasında ilgili kod bloğunu düzenleyelim.

```

1  @model MoviesViewModel
2
3  <h1> Film Listesi </h1>
4  <div id="filmler">
5      @if (Model.Movies.Count() > 0)
6      {
7          @foreach (var movie in Model.Movies)
8          {
9              @await Html.PartialAsync("_movie", movie)
10             }
11         }
12     else
13     {
14         <div class="alert alert-warning">
15             Film Bulunamadı
16         </div>
17     }
18 }
19 </div>
20 @section menu
21 {
22     @*await Html.PartialAsync("_menu", Model.Genres)*@
23     @await Component.InvokeAsync("Genres")
24 }

```

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.

```

GenresViewComponent.cs*  Movies.cshtml  MovieRepository.cs  MoviesController.cs
MovieApp.Web
1  using Microsoft.AspNetCore.Mvc;
2  using MovieApp.Web.Data;
3  using MovieApp.Web.Models;
4  using System.Collections.Generic;
5
6  namespace MovieApp.Web.ViewComponents
7  {
8      public class GenresViewComponent : ViewComponent
9      {
10         public IActionResult Invoke()
11         {
12             ViewBag.SelectedGenre = RouteData.Values["id"];
13             return View(GenreRepository.Genres);
14         }
15     }
16 }
17

```

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

```

Default.cshtml*  GenresViewComponent.cs  Movies.cshtml  MoviesController.cs  GenreRepository.cs  Genre.cs
MovieApp.Web
1  @model List<Genre>
2
3
4  <div class="list-group">
5      <a href="/movies/list"
6      class="list-group-item list-group-item-action">Tüm Filmler</a>
7
8      @foreach (var genre in Model)
9      {
10         <a href="/movies/list/@genre.GenreId"
11         class="list-group-item list-group-item-action @(ViewBag.SelectedGenre==genre.GenreId.ToString()?"active":"")">
12             @genre.Name</a>
13     }
14 </div>
15

```

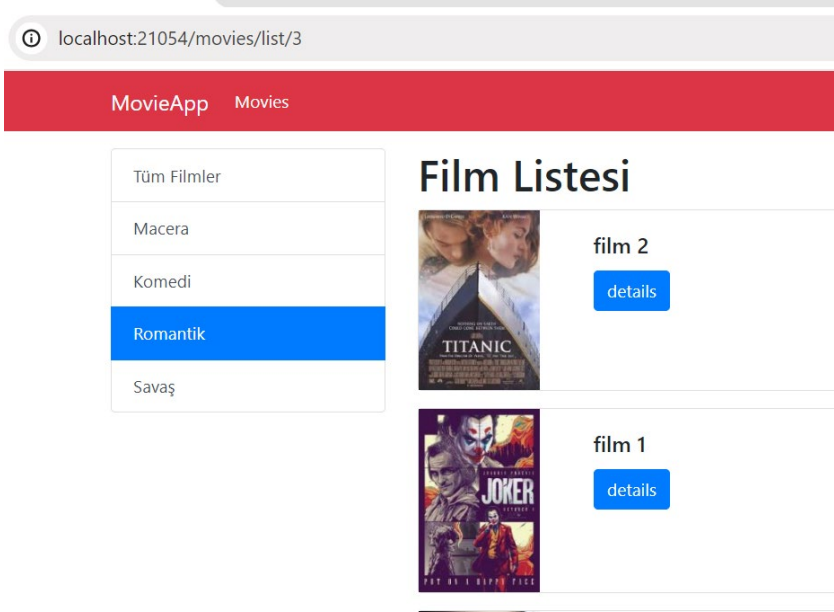
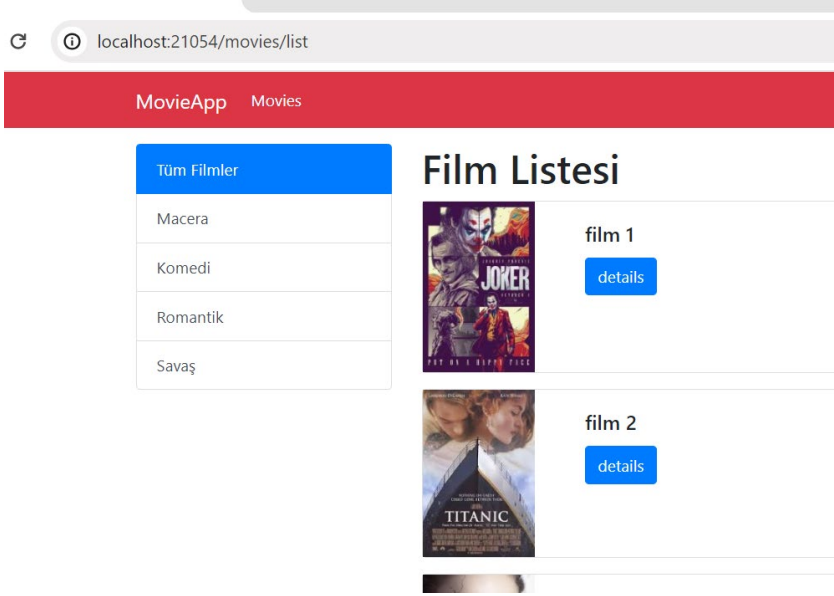
ADIM 3: Benzer kod satırı tüm filmler seçili olduğu duruma da eklenir.

```

Default.cshtml*  GenresViewComponent.cs  Movies.cshtml  MoviesController.cs  GenreRepository.cs  Genre.cs
MovieApp.Web
1  @model List<Genre>
2
3
4  <div class="list-group">
5      <a href="/movies/list"
6      class="list-group-item list-group-item-action @(ViewBag.SelectedGenre==null?"active":"")">Tüm Filmler</a>
7
8      @foreach (var genre in Model)
9      {
10         <a href="/movies/list/@genre.GenreId"
11         class="list-group-item list-group-item-action @(ViewBag.SelectedGenre==genre.GenreId.ToString()?"active":"")">
12             @genre.Name</a>
13     }
14 </div>
15

```

ADIM 4: Çalıştırarak kontrol edilir.



ÇALIŞMA SORUSU

Invoke metodunun uygulamada kullanım amacı nedir açıklayınız.