



ANKARA
HACI BAYRAM VELİ ÜNİVERSİTESİ
**UZAKTAN EĞİTİM UYGULAMA VE
ARAŞTIRMA MERKEZİ**

İNTERNET PROGRAMCILIĞI

Bu bölümde proje içinde kullanılacak olan Veri Arama Formu Oluşturma, Veri Ekleme Formu Oluşturma konularına yönelik uygulama yapılmıştır.

1. FİLM ARAMA FORMU OLUŞTURMA

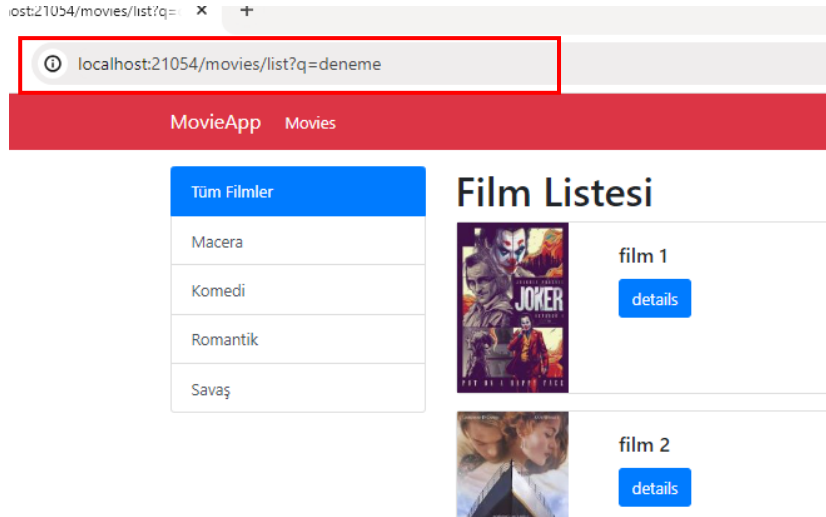
ADIM 1: _navbar.cshtml dosyasına ilgili kod bloğu eklenir.

```

_navbar.cshtml*  Default.cshtml  GenresViewComponent.cs  Movies.cshtml  MoviesController.cs  GenreReposit
MovieApp.Web
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         <form action="/movies/list" method="get" class="form-inline">
14             <input type="text" name="q" placeholder="Film Ara" class="form-control mr-2"/>
15             <button type="submit" class="btn btn-outline-light mr-0"> Bul </button>
16         </form>
17     </div>
18 </div>
19 </nav>
20
21
22

```

ADIM 2: Uygulamayı çalıştırarak film ara yerine “deneme” yazıp url yi kontrol ediyoruz.



ADIM 3: MoviesController.cs içinde url uygun olarak ilgili kod satırına ekleme yapılır.

```

10 {
11     public IActionResult Index()
12     {
13         return View();
14     }
15     //localhost:****/movies/list
16     //localhost:****/movies/list/1
17     public IActionResult List(int? id, string q)
18     {
19         var movies = MovieRepository.Movies;
20         if (id!=null)
21         {
22             movies = movies.Where(m => m.GenreId == id).ToList();
23         }
24
25         var model = new MoviesViewModel()
26         {
27             Movies = movies
28         }
29     }
30 }

```

ADIM 4: MoviesController.cs içine girilen q değerini kontrol edecek if bloğu eklenir.

```

16 //localhost:****/movies/list/1
17 public IActionResult List(int? id, string q)
18 {
19     var movies = MovieRepository.Movies;
20     if (id!=null)
21     {
22         movies = movies.Where(m => m.GenreId == id).ToList();
23     }
24
25     if(!string.IsNullOrEmpty(q))
26     {
27         movies = movies.Where(i =>
28             i.Title.ToLower().Contains(q.ToLower()) ||
29             i.Description.ToLower().Contains(q.ToLower())).ToList();
30     }
31
32     var model = new MoviesViewModel()
33     {
34         Movies = movies
35     };
36
37     return View("Movies", model);
38 }

```

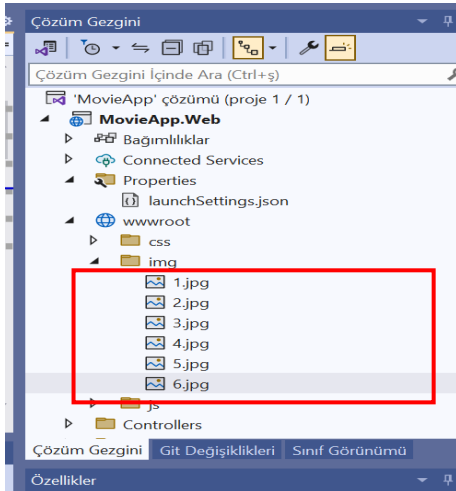
ADIM 5: MovieRepository.cs içindeki film bilgilerini doğru şekilde dolduralım.

```

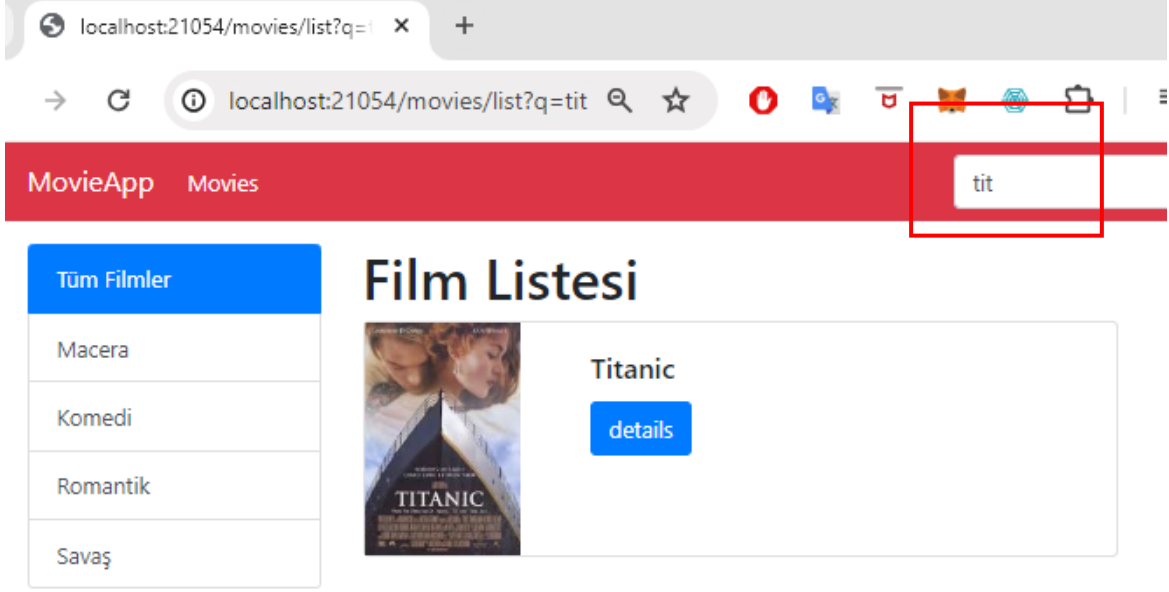
11 {
12     _movies = new List<Movie>()
13     {
14         new Movie
15         {
16             MovieId=1,
17             Title="Joker",
18             Description="Joker, 2019 yapımı Amerikan psikolojik gerilim",
19             Director="Todd Phillips",
20             Players=new string[] { "Ed Helms", "Justin Bartha" },
21             ImageUrl="1.jpg",
22             GenreId=4
23         },
24         new Movie
25         {
26             MovieId=2,
27             Title="Titanic",
28             Description="Titanik (İngilizce: Titanic), James Cameron'ın",
29             Director="James Cameron"
30         }
31     }
32 }

```

ADIM 6: Resimleri düzenleyelim



ADIM 7: Çalıştırılır. Arama kısmına herhangi bir hece yazılarak kontrol edilir.



2. FİLM EKLEME FORMU OLUŞTURMA

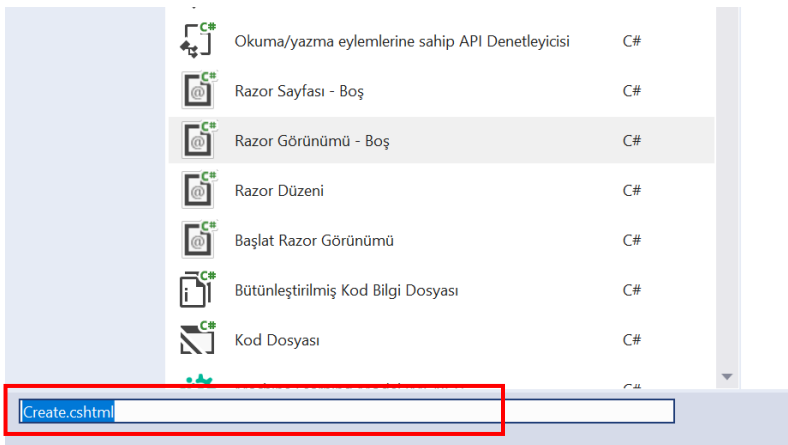
ADIM 1: Oluşturacağımız html sayfasını **MoviesController.cs** de tanımlayalım. İlgili kod bloğu eklenir.

```

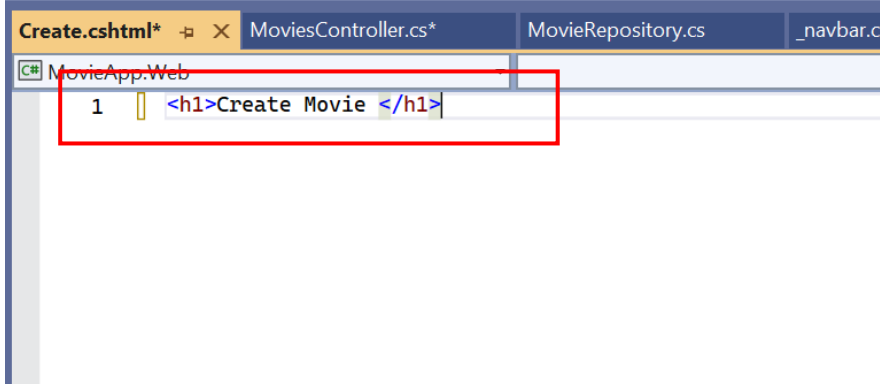
34         Movies = movies
35     };
36
37     return View("Movies", model);
38 }
39 //localhost:****/movies/details/1
40 public IActionResult Details(int id)
41 {
42     return View(MovieRepository.GetById(id));
43 }
44
45 public IActionResult Create()
46 {
47     return View();
48 }
49 }
50 }
51

```

ADIM 2: Movies Klasörü altına sağ tıklayarak **Boş Razor Görünümü (View)** eklenir. **Create.cshtml** olarak isimlendirilir.

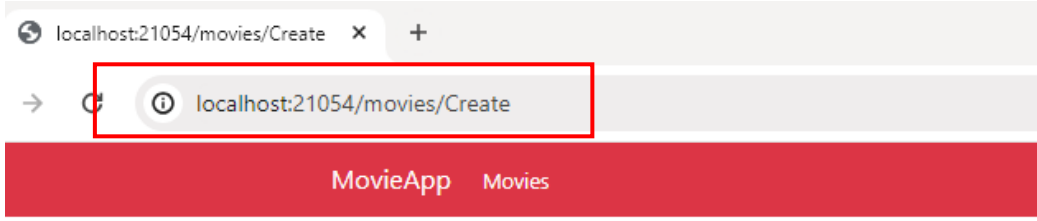


ADIM 3: Basit bir html kodu oluşturalım.



```
1 <h1>Create Movie </h1>
```

ADIM 4: Çalıştırılır kontrol edilir.



Create Movie

ADIM 5: Create.cshtml için ilgili kod bloğu yazılır.

```

1 <h1>Create Movie </h1>
2 <hr/>
3 <form>
4   <div class="form-group">
5     <label for="Title"> Title </label>
6     <input type="text" name="Title" class="form-control" />
7   </div>
8   <div class="form-group">
9     <label for="Description"> Description </label>
10    <textarea name="Description" class="form-control"> </textarea>
11  </div>
12  <div class="form-group">
13    <label for="Director"> Director </label>
14    <input type="text" name="Title" class="form-control" />
15  </div>
16  <div class="form-group">
17    <label for="ImageUrl"> ImageUrl </label>
18    <input type="text" name="ImageUrl" class="form-control" />
19  </div>
20  <div class="form-group">
21    <label for="GenreId"> GenreId </label>
22    <input type="text" name="GenreId" class="form-control" />
23  </div>
24  <button type="submit" class="btn btn-primary">Submit</button>
25 </form>

```

ADIM 6: _navbar.cshtml dosyasına ilgili kod bloğu düzenlenir ve eklenir.

```

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" data-target="#navbarToggler"
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">
10          <a href="/movies/list" class="nav-link">Movies</a>
11        </li>
12        <li class="navbar-item">
13          <a href="/movies/create" class="nav-link">Create Movie</a>
14        </li>
15      </ul>
16    </div>
17    <form action="/movies/list" method="get" class="form-inline">
18      <input type="text" name="q" placeholder="Film Ara" class="form-control mr-2"/>
19      <button type="submit" class="btn btn-outline-light mr-0"> Bul </button>
20    </form>
21  </div>
22 </nav>

```

ADIM 7: Çalıştırılarak kontrol edilir

MovieApp Movies Create Movie Film A

Create Movie

Title

Description

Director

ImageUrl

Genrelid

Submit

3. FORM BİLGİLERİNİN POST İLE GÖNDERİLMESİ

ADIM 1: Server tarafından bir kaynak talebinde bulunabilmek için [HttpGet] kod satırı ilgili yerlere eklenir.

```

MoviesController.cs* _navbar.cshtml Create.cshtml MovieRepository.cs
MovieApp.Web MovieApp.Web.Controllers.MoviesController
31
32     var model = new MoviesViewModel()
33     {
34         Movies = movies
35     };
36
37     return View("Movies", model);
38 }
39
40 [HttpGet]
41 //localhost:****/movies/details/1
42 public IActionResult Details(int id)
43 {
44     return View(MovieRepository.GetById(id));
45 }
46 [HttpGet]
47 public IActionResult Create()
48 {
49     return View();
50 }
51

```

ADIM 2: Server tarafından bir kaynak talebinde bulunabilmek için [HttpGet] kod satırı ilgili yerlere eklenir.

```

7 namespace MovieApp.Web.Controllers
8 {
9     public class MoviesController : Controller
10    {
11        public IActionResult Index()
12        {
13            return View();
14        }
15        //localhost:****/movies/list
16        //localhost:****/movies/list/1
17        [HttpGet]
18        public IActionResult List(int? id, string q)
19        {
20            var movies = MovieRepository.Movies;
21            if (id!=null)
22            {
23                movies = movies.Where(m => m.GenreId == id).ToList
24            }
25
26            if(!string.IsNullOrEmpty(q))
27            {
28                movies = movies.Where(i =>
29                    i.Title.ToLower().Contains(q.ToLower()) ||

```

ADIM 3: Yeni girilecek film bilgilerini post aracılığı ile göndermek istediğimiz için Create.cshtml de ilgili kod satırı düzenlenir.

```

1 <h1>Create Movie </h1>
2 <hr/>
3 <form method="post" action="/movies/create">
4     <div class="form-group">
5         <label for="Title"> Title </label>
6         <input type="text" name="Title" class="form-control" />
7     </div>
8     <div class="form-group">
9         <label for="Description"> Description </label>
10        <textarea name="Description" class="form-control"> </textarea>
11    </div>
12    <div class="form-group">
13        <label for="Director"> Director </label>
14        <input type="text" name="Title" class="form-control" />
15    </div>
16    <div class="form-group">
17        <label for="ImageUrl"> ImageUrl </label>
18        <input type="text" name="ImageUrl" class="form-control" />
19    </div>
20    <div class="form-group">

```

ADIM 4: Create.cshtml de post action metodu kullandığımız için **MoviesController.cs** de action metodunu **post** olarak tanımlıyoruz.

```

37     };
38
39     return View("Movies", model);
40 }
41
42 [HttpGet]
43 //localhost:****/movies/details/1
44 public IActionResult Details(int id)
45 {
46     return View(MovieRepository.GetById(id));
47 }
48 [HttpGet]
49 public IActionResult Create()
50 {
51     return View();
52 }
53 [HttpPost]
54 public IActionResult Create(string Title, string Description, string Director, string ImageUrl, int GenreId)
55 {
56     return View();
57 }
58
59

```

4. FİLM BİLGİLERİNİN KAYDEDİLMESİ

Daha önce **MovieRepository** de **Add** metodu tanımlamıştık. Bu metod aracılığı ile eklenen film yine **MovieRepository** içindeki film listesine aktarılmış olacak.

ADIM 1: Bunun için öncelikle **MoviesController.cs** içerisinde bir m objesi için bir döngü oluşturacağız ve formdan gelen bilgilere göre döngüyü düzenleyeceğiz. **MoviesController.cs** de ilgili kod bloğu oluşturulur.

```

49 public IActionResult Create()
50 {
51     return View();
52 }
53 [HttpPost]
54 public IActionResult Create(string Title, string Description, string Director)
55 {
56     var m = new Movie()
57     {
58         Title = Title,
59         Description = Description,
60         Director = Director,
61         ImageUrl = ImageUrl,
62         GenreId = GenreId
63     };
64     MovieRepository.Add(m);
65     return View();
66 }
67 }
68 }
69

```

ADIM 2: Eklemiş olduğumuz filmlere otomatik olarak bir MovieId atamak için MovieRepository.cs deki Add metodu içine ilgili kod satırı yazılır.

```

79 get
80 {
81     return _movies;
82 }
83 }
84 public static void Add(Movie movie)
85 {
86     movie.MovieId = _movies.Count() + 1;
87     _movies.Add(movie);
88 }
89 }
90 public static Movie GetById (int id)
91 {
92     // ilgili liste üzerinde bütün elemanlara tek tek döngü şeklinde bakar
93     // m yukarıdaki listenin her bir elemanını ifade eder
94     return _movies.FirstOrDefault(m => m.MovieId == id);
95 }
96 }
97 }
98

```

ADIM 3: Oyuncu bilgilerine ilişkin herhangi bir işlem yapmadık, kullanıcının oyuncu bilgilerini boş bırakabilmesini istiyorsak, **Details.cshtml** de daha önce players (oyuncular) için oluşturmuş olduğumuz döngünün oyuncu bilgileri boş işe çalıştırılmaması için players döngü kodları bir if bloğu içine alınır.

```

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      @if (Model.Players!=null)
14      {
15        <p>
16          <ul>
17            @foreach (var item in Model.Players)
18            {
19              <li>@item</li>
20            }
21          </ul>
22        </p>
23      }
24    </div>

```

ADIM 4: Çalıştırılarak film ekleme işlemi aşağıdaki bilgilerle kontrol edilir.

localhost:21054/movies/create

MovieApp Movies Create Movie

Create Movie

Title
Yeni Film

Description
Açıklama

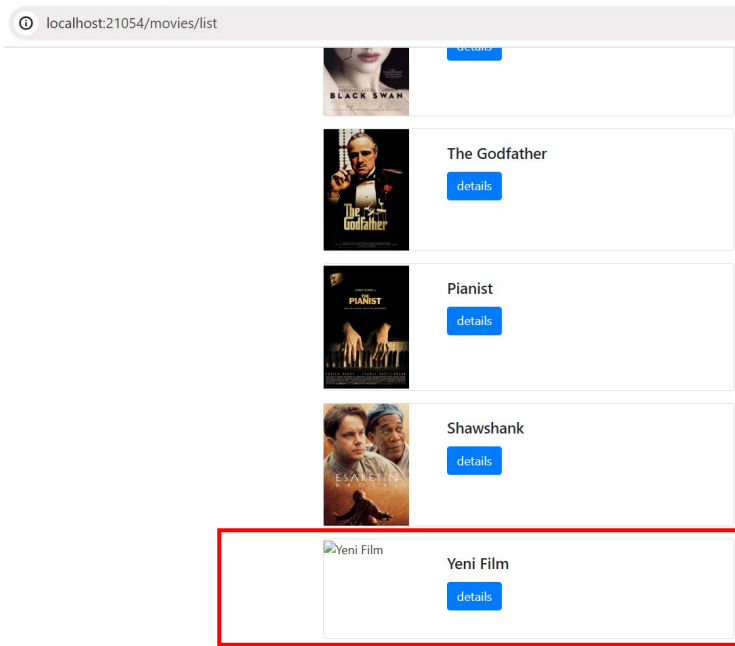
Director
Nuri Bilge Ceylan

ImageUrl
1

GenreId
1

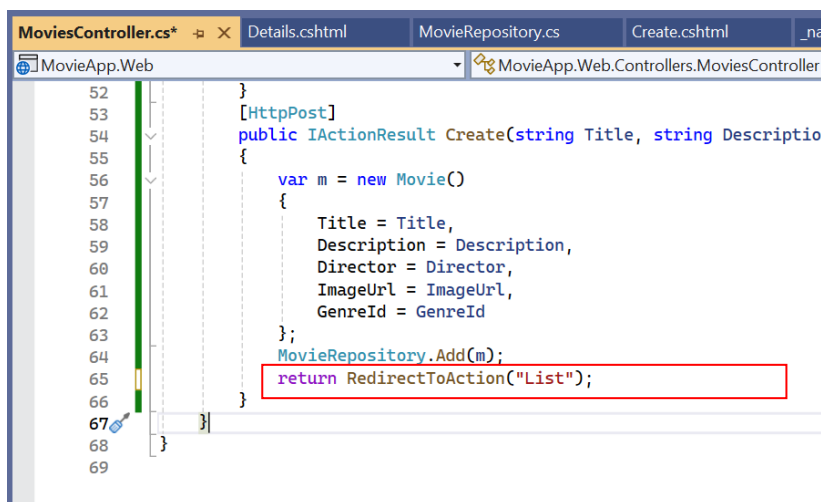
Submit

ADIM 5: Submit tuşuna bastıktan sonra, Movies/list e dönerek eklenen yeni filmi kontrol edelim.



ADIM 6: Ekleme işlemi yapıp Submit ettikten sonra karşımızda film ekleme formu boş şekilde kalıyor. Bunun yerine ekleme işlemi yaptıktan sonra form a değil film listesi sayfasına yönlendirelim kullanıcıyı.

Bunun için **MoviesController.cs** sayfasında ilgili kod satırı düzenlenir. Yine MoviesController.cs içinde list action metoduna gitmesini istediğimiz için "List") olarak düzenliyoruz.



ADIM 7: Çalıştırarak yeni film eklemesi yapalım. Submit butonuna basalım.

localhost:21054/movies/create

MovieApp Movies Create Movie

Create Movie

Title

Yeni Film

Description

Açıklama

Director

Yönetmen

ImageUrl

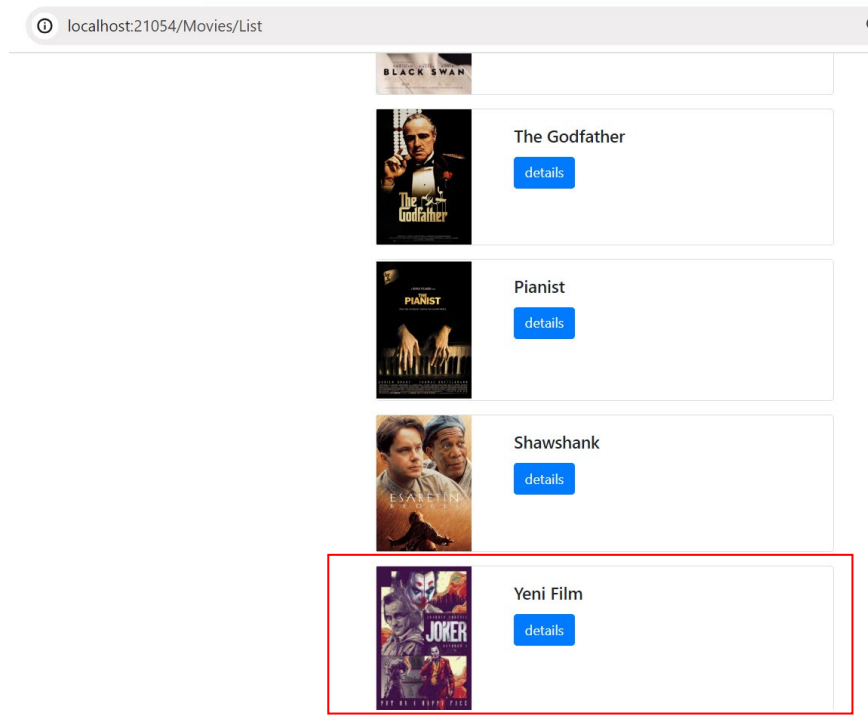
1.jpg

Genrelid

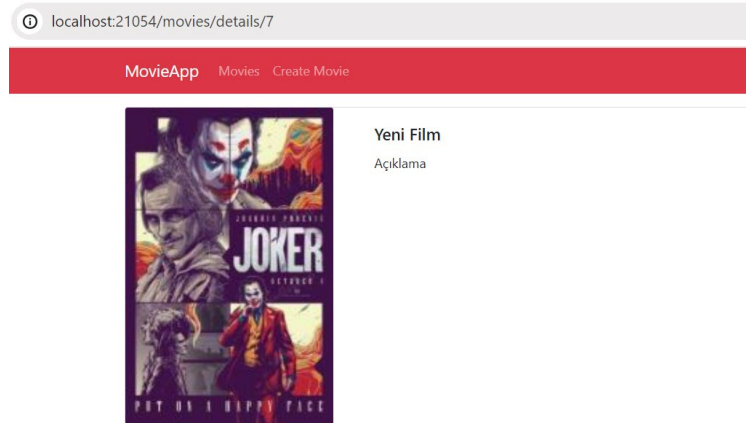
1

Submit

ADIM 8: Ekleme işlemi sonrası karşımıza doğrudan film listesi geldiğini ve yeni filmin eklendiğini kontrol edelim.



ADIM 9: Yeni eklenen filmin detay sayfasına giderek oyuncu bilgileri ile ilgili yazdığımız if bloğunda hata alıp almadığımızı kontrol edelim.



BONUS ADIM 10: **MoviesController** da yazmış olduğumuz parametreler (Title, Description, Director, ImageUrl, GenrelId) ile **Create.cshtml** dosyasındaki parametreler aynı. Bu durumda

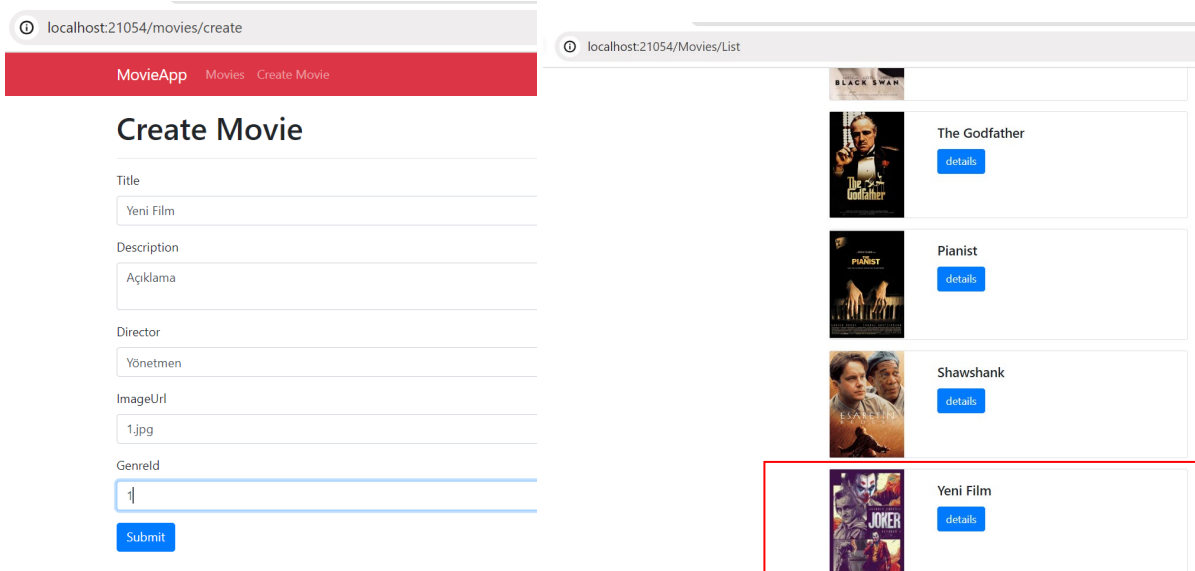
MoviesController.cs deki kod bloğunu şu şekilde düzenleseydik de metod hatasız çalışırdı. (Alttaki kod bloğu yorum satırı olarak düzenlendi.)

```

43 //localhost:****/movies/details/1
44 public IActionResult Details(int id)
45 {
46     return View(MovieRepository.GetById(id));
47 }
48 [HttpGet]
49 public IActionResult Create()
50 {
51     return View();
52 }
53 [HttpPost]
54 public IActionResult Create(Movie m)
55 {
56     // var m = new Movie()
57     // {
58     //     Title = Title,
59     //     Description = Description,
60     //     Director = Director,
61     //     ImageUrl = ImageUrl,
62     //     GenreId = GenreId
63     // };
64     MovieRepository.Add(m);
65     return RedirectToAction("List");
66 }
67 }
68

```

ADIM 11: Çalıştırılarak kontrol edilir.



localhost:21054/movies/details/7

MovieApp Movies Create Movie



Yeni Film

Açıklama

ÇALIŞMA SORUSU

Navbar kullanım amaçları nelerdir. Bootstrap kütüphanesinden farklı bir navbar kod bloğu elde ederek çalıştırınız.