



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

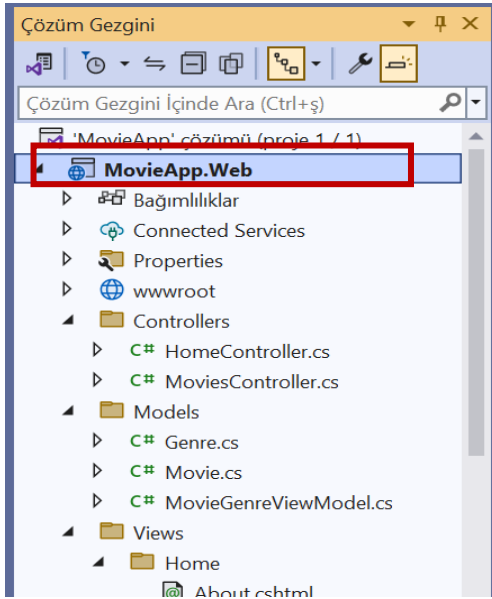
İNTERNET PROGRAMCILIĞI

Bu bölümde proje içinde kullanılacak olan View Components, Repository uygulamalarına yer verilmiştir.

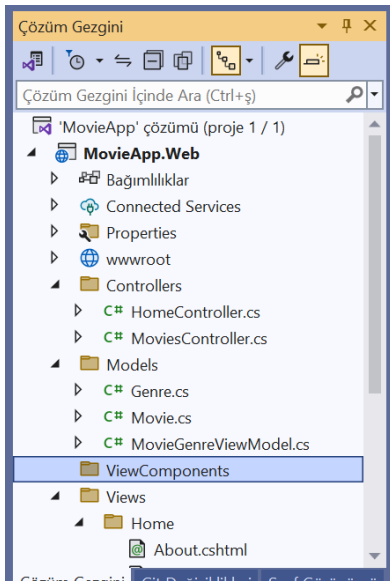
1. VIEW COMPONENTS

Shared klasörü içinde oluşturduğumuz .cshtml dosyaları çalışmak için bir modele ihtiyaç duyuyor. .cshtml lerin ihtiyaç duydukları modeli kendileri elde edebilmeleri için, bu dosyalara dışardan model göndermek zorunda kalmamak için View Components ler kullanılır.

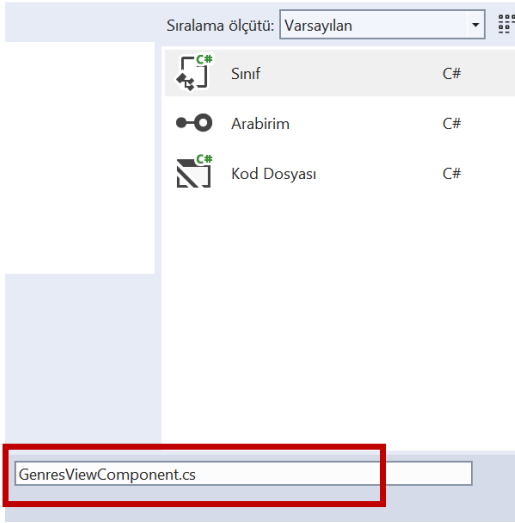
ADIM 1: Projeye sağ tıklanarak yeni klasör (new folder) eklenir.



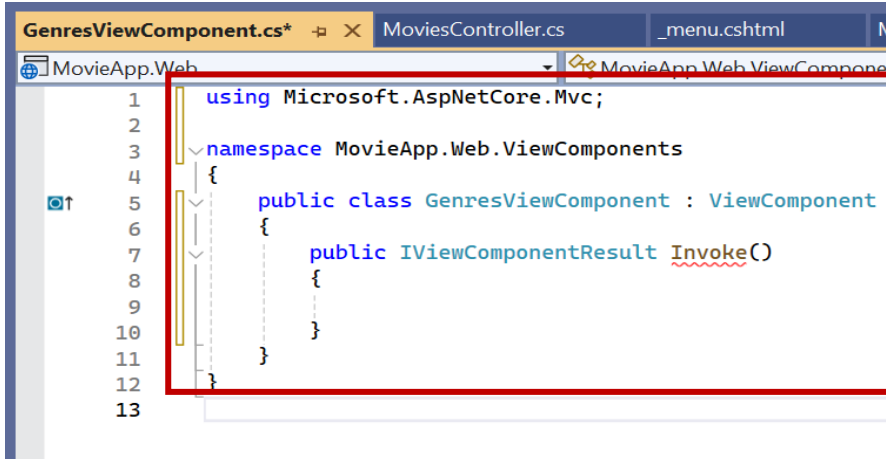
ADIM 2: Klasör **ViewComponents** olarak isimlendirilir.



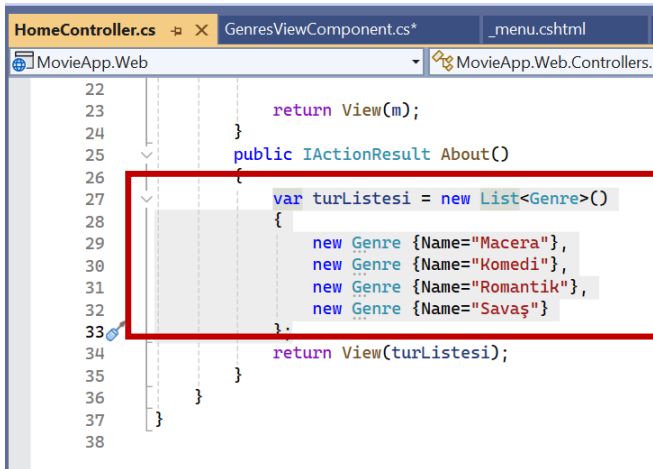
ADIM 3: ViewComponents klasörü üzerine sağ tıklanarak yeni **sınıf (class)** eklenir. **GenresViewComponent** olarak isimlendirilir.



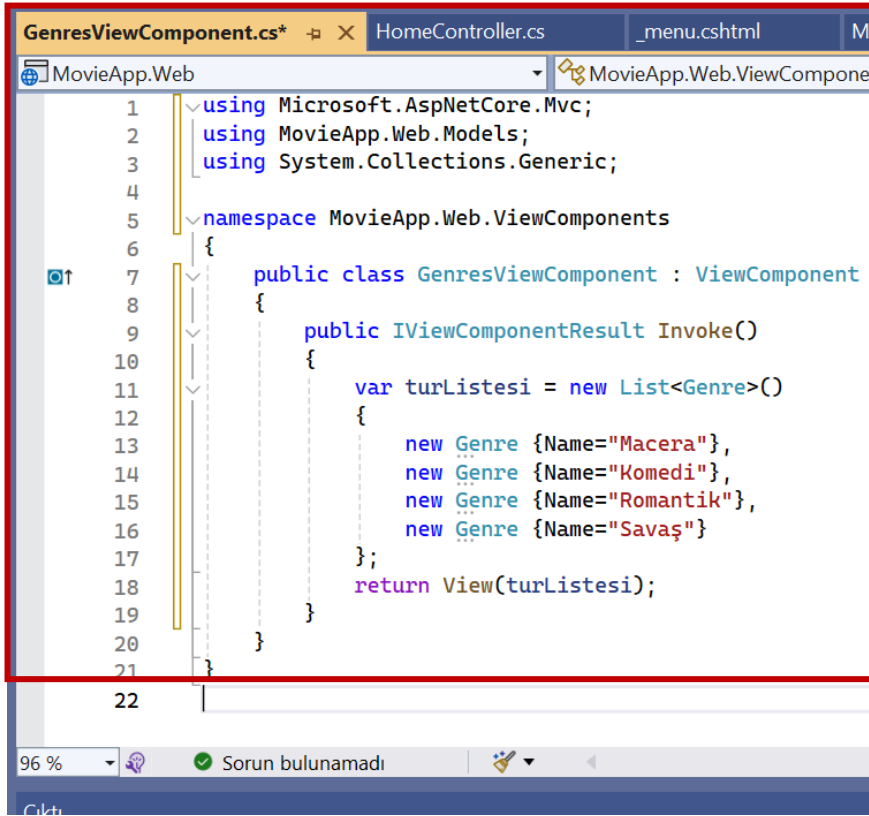
ADIM 4: GenresViewComponent.cs dosyası içine yazılması gereken kod bloğu. Bu metod içinde bir liste talep ediyor.!!!!



ADIM 5: HomeController.cs içinden bir liste kod bloğu kopyalanır.



ADIM 6: GenresViewComponent.cs içinde ilgili alana yapıştırılır.

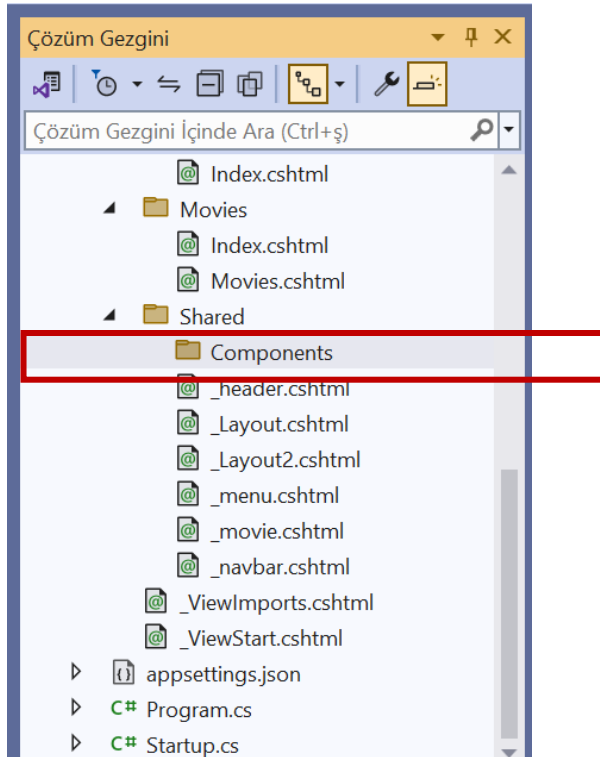


```

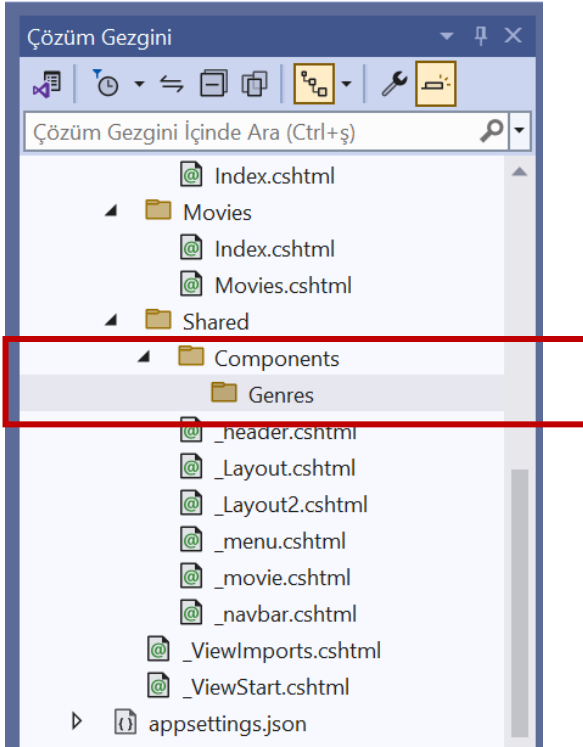
1  using Microsoft.AspNetCore.Mvc;
2  using MovieApp.Web.Models;
3  using System.Collections.Generic;
4
5  namespace MovieApp.Web.ViewComponents
6  {
7      public class GenresViewComponent : ViewComponent
8      {
9          public IViewComponentResult Invoke()
10         {
11             var turListesi = 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

```

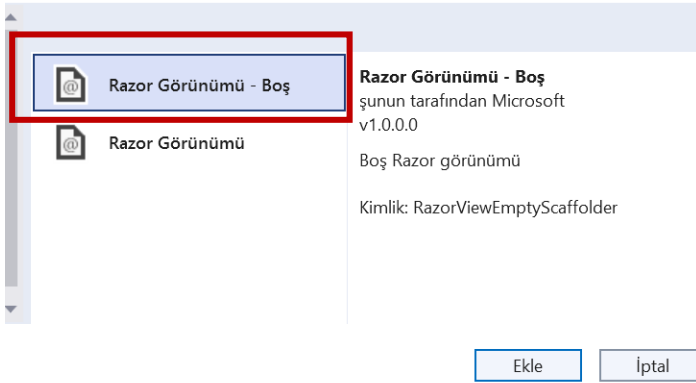
ADIM 7: Oluşturduğumuz **Model** için bir de **View** oluşturmamız gerekli. Bunun için öncelikle **Shared** klasörüne sağ tıklayarak tüm componentleri içine alabileceğimiz **Components** isiminde bir **klasör** oluşturulur.



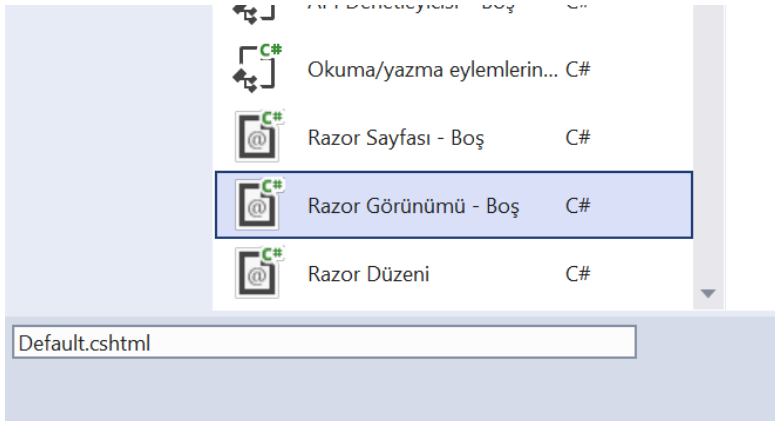
ADIM 8: Components klasörü altına da Genres adında bir klasör eklensin.



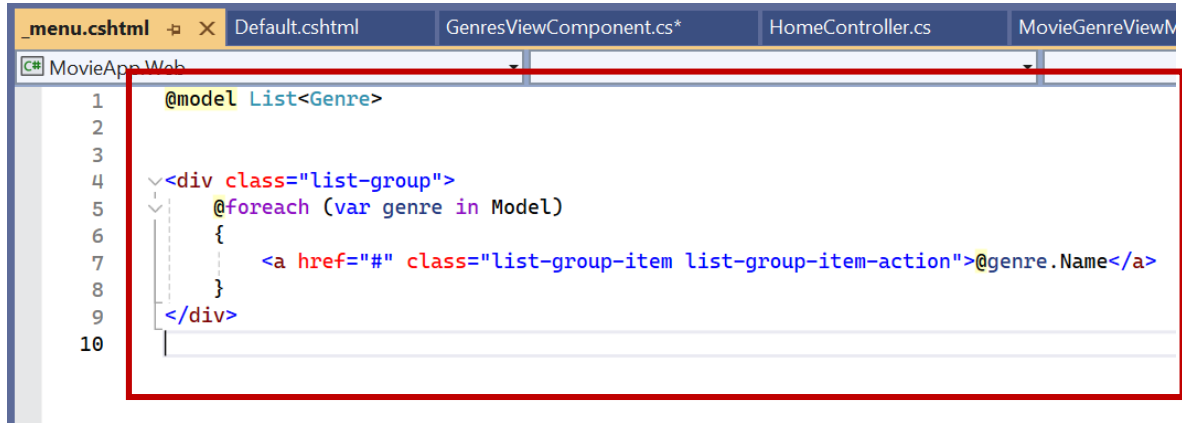
ADIM 9: Genres klasörüne sağ tıklanarak bir **Razor View** eklenir.



ADIM 10: Default.cshtml olarak isimlendirilir.



ADIM 11: _menu.cshtml deki kodlar kopyalanır.



```

1  @model List<Genre>
2
3
4  <div class="list-group">
5      @foreach (var genre in Model)
6      {
7          <a href="#" class="list-group-item list-group-item-action">@genre.Name</a>
8      }
9  </div>
10

```

ADIM 12: Default.cshtml içine yapıştırılır.

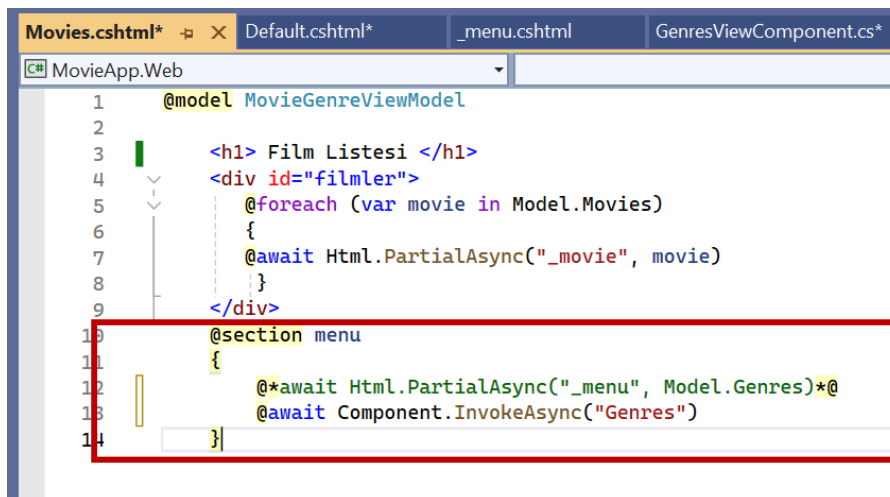


```

1  @model List<Genre>
2
3
4  <div class="list-group">
5      @foreach (var genre in Model)
6      {
7          <a href="#" class="list-group-item list-group-item-action">@genre.Name</a>
8      }
9  </div>
10

```

ADIM 13: Movies.cshtml içindeki kod bloğunun ilk satırı yorum satırı olarak kapatılır, altına satır eklenir.



```

1  @model MovieGenreViewModel
2
3  <h1> Film Listesi </h1>
4  <div id="filmler">
5      @foreach (var movie in Model.Movies)
6      {
7          @await Html.PartialAsync("_movie", movie)
8      }
9  </div>
10
11  @section menu
12  {
13      @*await Html.PartialAsync("_menu", Model.Genres)*@
14      @await Component.InvokeAsync("Genres")
15  }

```

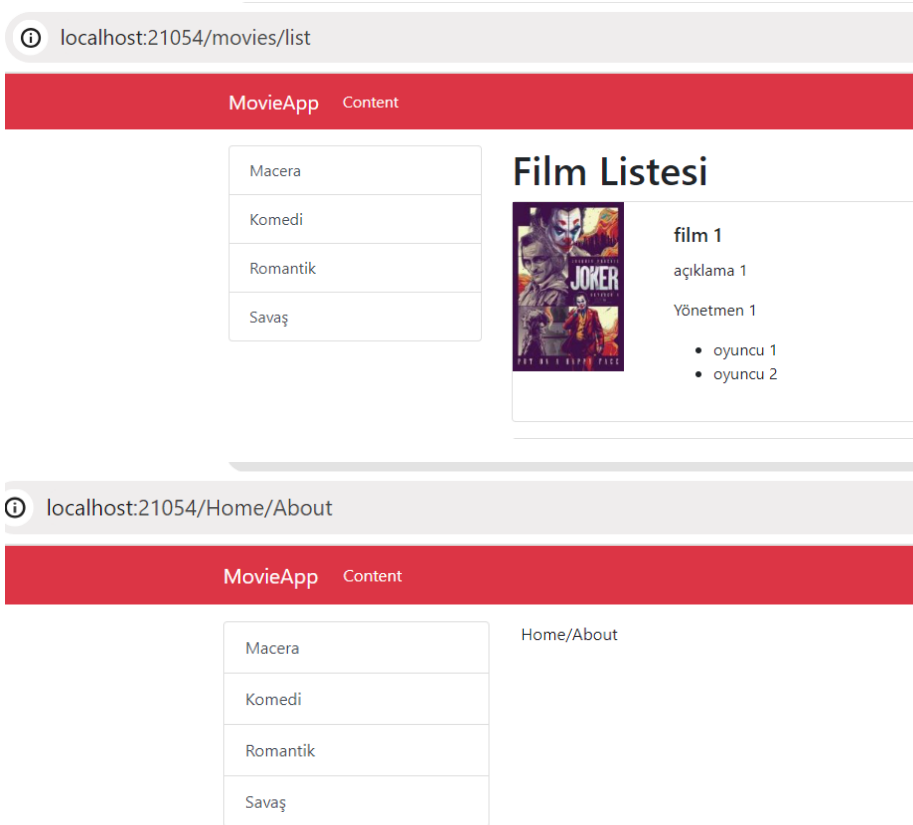
ADIM 14: Aynı işlem **About.cshtml** için de yapılır.

```

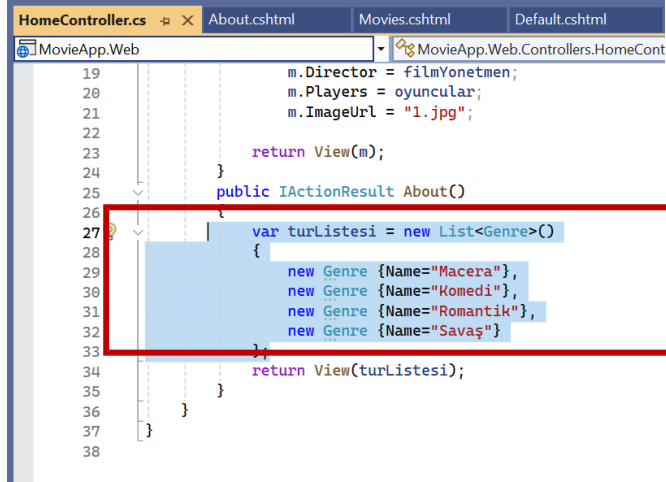
1  @model List<Genre>
2  <p>
3      Home/About
4  </p>
5  @section menu
6  {
7      @*await Html.PartialAsync("_menu", Model.Genres)*@
8      @await Component.InvokeAsync("Genres")
9  }

```

ADIM 15: Uygulama çalıştırılır. home/about ve movie/list kontrol edilir.



ADIM 16: HomeController.cs de ilgili tür listesine ihtiyaç kalmadı. İşaretili kod bloğu silinir.

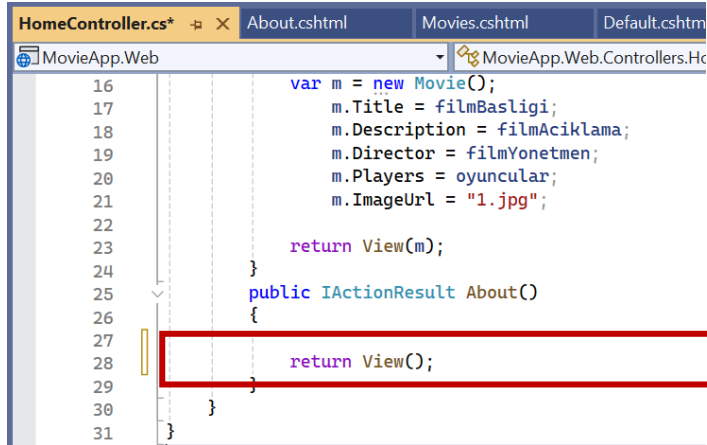


```

19      m.Director = filmYonetmen;
20      m.Players = oyuncular;
21      m.ImageUrl = "1.jpg";
22
23      return View(m);
24  }
25  public IActionResult About()
26  {
27      var turListesi = new List<Genre>()
28      {
29          new Genre {Name="Macera"},
30          new Genre {Name="Komedisi"},
31          new Genre {Name="Romantik"},
32          new Genre {Name="Savaş"}
33      }
34      return View(turListesi);
35  }
36  }
37  }
38  }

```

ADIM 17: return içi düzeltilir.



```

16      var m = new Movie();
17      m.Title = filmBasligi;
18      m.Description = filmAciklama;
19      m.Director = filmYonetmen;
20      m.Players = oyuncular;
21      m.ImageUrl = "1.jpg";
22
23      return View(m);
24  }
25  public IActionResult About()
26  {
27      return View();
28  }
29  }
30  }
31  }

```

ADIM 18: MoviesController.cs de de artık bir tür listesine ihtiyaç kalmadı. İlgili kod bloğu buradan da silinir.


```

41 Players=new string[] { "oyuncu 1", "oyuncu 2" };
42 ImageUrl="3.jpg";
43 } };
44
45 var turListesi = new List<Genre>()
46 {
47     new Genre {Name="Macera"},
48     new Genre {Name="Komedi"},
49     new Genre {Name="Romantik"},
50     new Genre {Name="Savaş"}
51 };
52
53 var model = new MovieGenreViewModel()
54 {
55     Movies = filmListesi,
56     Genres = turListesi
57 };
58
59 return View("Movies", model);

```

ADIM 19: Alt kod bloğundaki **Genres=turListesi** satırı da silinir.

```

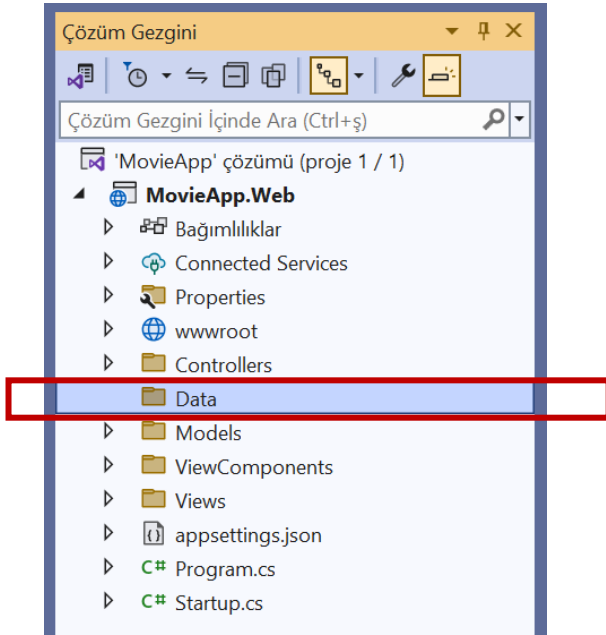
41 Players=new string[] { "oyuncu 1", "oyuncu 2" };
42 ImageUrl="3.jpg";
43 } };
44
45
46
47 var model = new MovieGenreViewModel()
48 {
49     Movies = filmListesi,
50     Genres = turListesi
51 };
52
53 return View("Movies", model);
54 }
55
56 //localhost:***:/movies/details
57 public string Details()
58 {
59 }

```

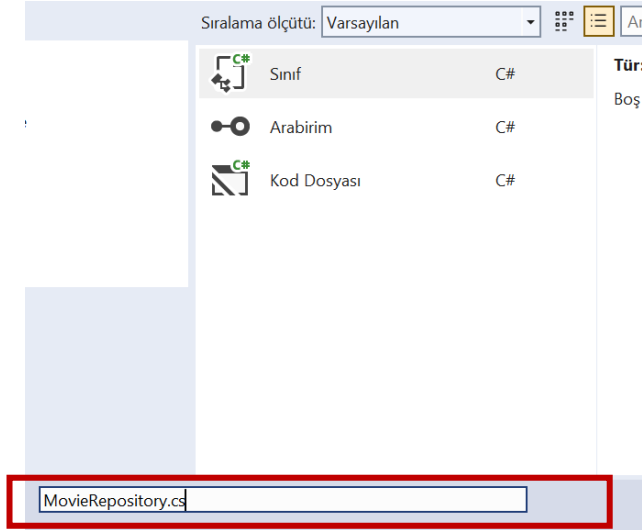
Bu derste dışarıdan gönderilen bir PartialAsync oluşturmak yerine, kendi başına çalışan bir yapı oluşturduk ve artık ilgili tür listesinin HomeController.cs de bulunmasına gerek kalmadı. Çünkü ilgili component için gelen model bilgileri **GenresViewComponent.cs** kendi içerisinde hazırlayıp kullanabiliyor.

2. REPOSITORY

ADIM 1: Proje üzerine sağ tıklayarak yeni bir klasör oluşturulur. **Data** olarak isimlendirilir.



ADIM 2: Data klasörü içine yeni bir **Class** eklenir (Bu class daha sonra Genre ve Movie Modellerine veri sağlayacak şekilde kullanılacaktır). **MovieRepository.cs** olarak isimlendirilir.



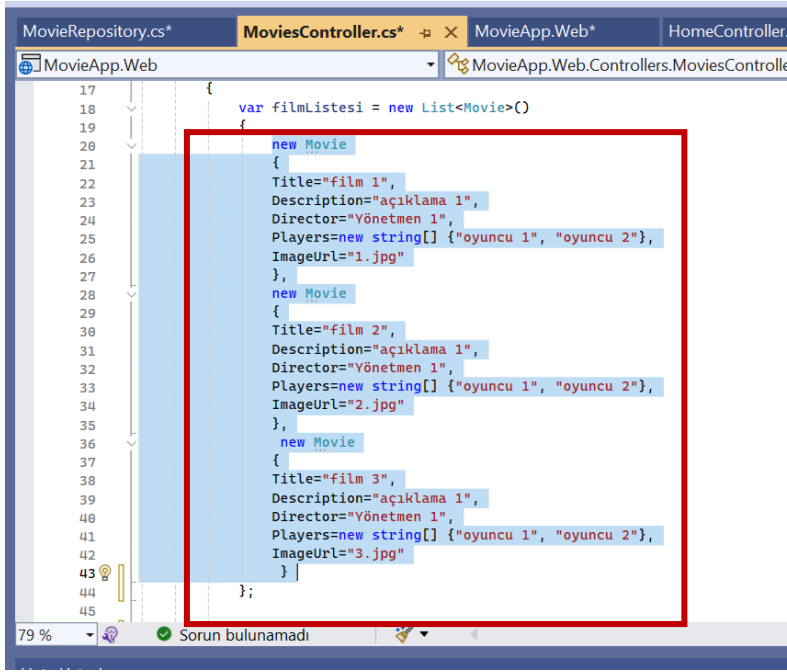
ADIM 3: Class içine sadece okunabilir dışarıdan ulaşılamayan bir liste ve ilgili kütüphaneler tanımlanır.

```

1  using System.Collections.Generic;
2  using MovieApp.Web.Models;
3
4  namespace MovieApp.Web.Data
5  {
6      public class MovieRepository
7      {
8          private static readonly List<Movie> _movies = null;
9          static MovieRepository()
10         {
11             _movies = new List<Movie>()
12             {
13             };
14         }
15     }
16 }

```

ADIM 4: MoviesController.cs içerisinde ilgili liste bilgileri kopyalanır.



```
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 }
```

ADIM 5: MovieRepository.cs içinde ilgili alana eklenir.

```

6      public class MovieRepository
7      {
8          private static readonly List<Movie> _movies = null;
9          static MovieRepository()
10         {
11             _movies = new List<Movie>()
12             {
13                 new Movie
14                 {
15                     Title="film 1",
16                     Description="açıklama 1",
17                     Director="Yönetmen 1",
18                     Players=new string[] {"oyuncu 1", "oyuncu 2"},
19                     ImageUrl="1.jpg"
20                 },
21                 new Movie
22                 {
23                     Title="film 2",
24                     Description="açıklama 1",
25                     Director="Yönetmen 1",
26                     Players=new string[] {"oyuncu 1", "oyuncu 2"},
27                     ImageUrl="2.jpg"
28                 },
29                 new Movie
30                 {
31                     Title="film 3",
32                     Description="açıklama 1",
33                     Director="Yönetmen 1",
34                     Players=new string[] {"oyuncu 1", "oyuncu 2"},
35                     ImageUrl="3.jpg"
36                 }
37             };

```

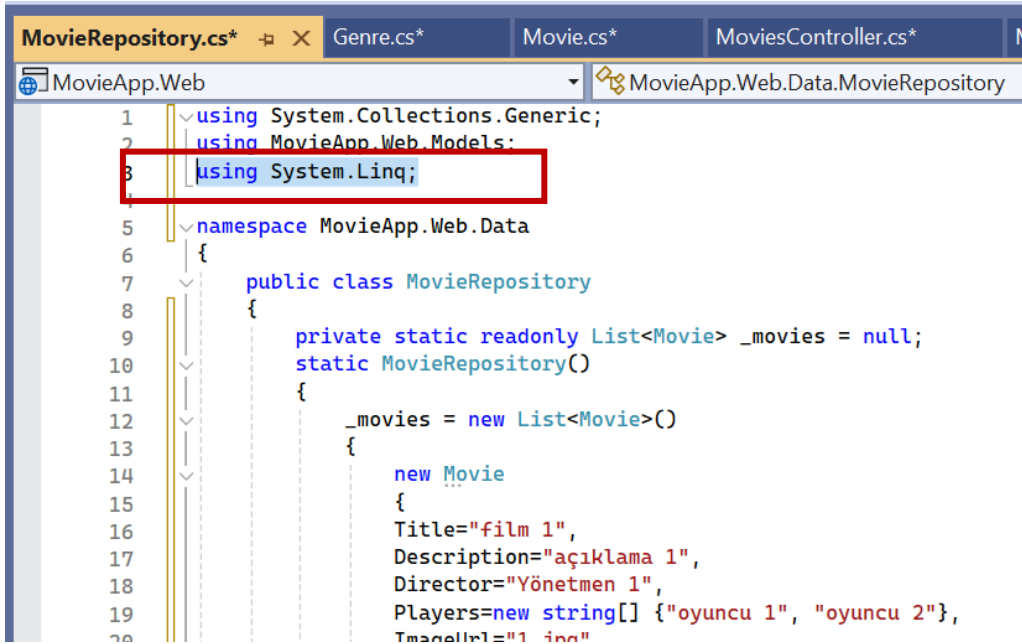
ADIM 6: Yine **MovieRepository.cs** içerisinde daha sonra kullanılacak ve açıklanacak olan metodlar için üç ayrı kod bloğunu ekliyoruz.

```

34         Players=new string[] {"oyuncu 1", "oyuncu 2"},
35         ImageUrl="3.jpg"
36     }
37 }
38
39 public static List<Movie> Movies
40 {
41     get
42     {
43         return _movies;
44     }
45 }
46 public static void Add(Movie movie)
47 {
48     _movies.Add(movie);
49 }
50
51 public static Movie GetById (int id)
52 {
53     // ilgili liste üzerinde bütün elemanlara tek tek döngü şeklinde bakar
54     // m yukarıdaki listenin her bir elemanını ifade eder
55     return _movies.FirstOrDefault(m => m.MovieId == id);
56 }
57
58 }

```

ADIM 7: FirstOrDefault kod satırının çalışması için gerekli olan **using System.Linq;** kütüphanesi **MovieRepository.cs** sayfasına eklenir.

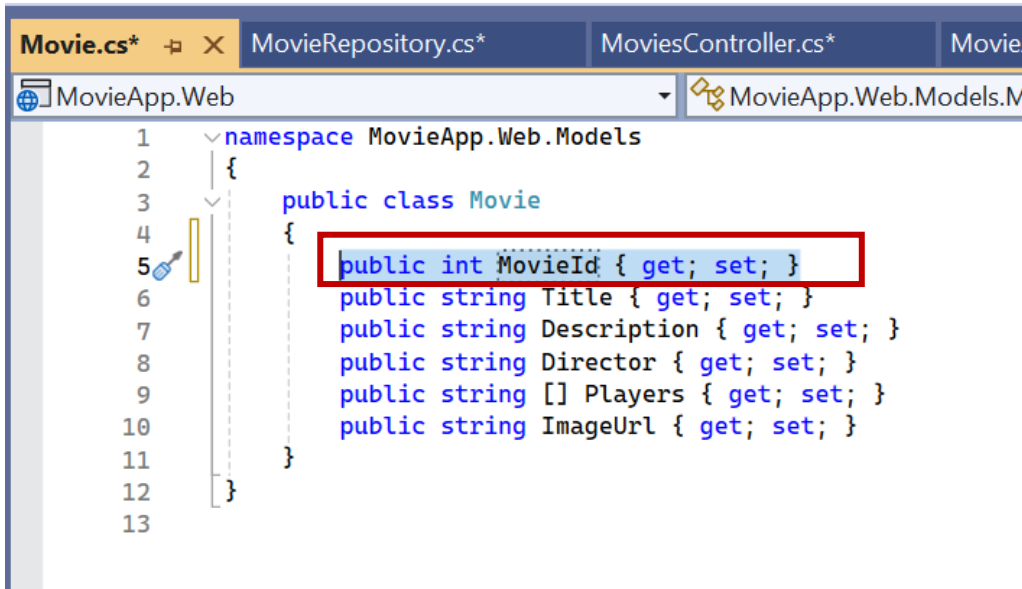


```

1  using System.Collections.Generic;
2  using MovieApp.Web.Models;
3  using System.Linq;
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                     Title="film 1",
17                     Description="açıklama 1",
18                     Director="Yönetmen 1",
19                     Players=new string[] { "oyuncu 1", "oyuncu 2"},
20                     ImageUrl="1.png"
21                 }
22             }
23         }
24     }
25 }

```

ADIM 8: Yukarıda tanımlanan **GetById** metodu **MovieId** ye göre her bir elemanı döngü şeklinde alır ve her bir elemanı m olarak kabul eder. Fakat bizim daha önce Models Klasörü içinde tanımladığımız **Movie.cs** klasındaki kod bloğunda **MovieId** yoktu. **MovieId** ekliyoruz.



```

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     }
12 }
13

```

ADIM 9: Aynı şekilde **Genre.cs** sayfasına da Filmlerin tür bilgilerine ilişkin id satırını GenreId olarak ekliyoruz.

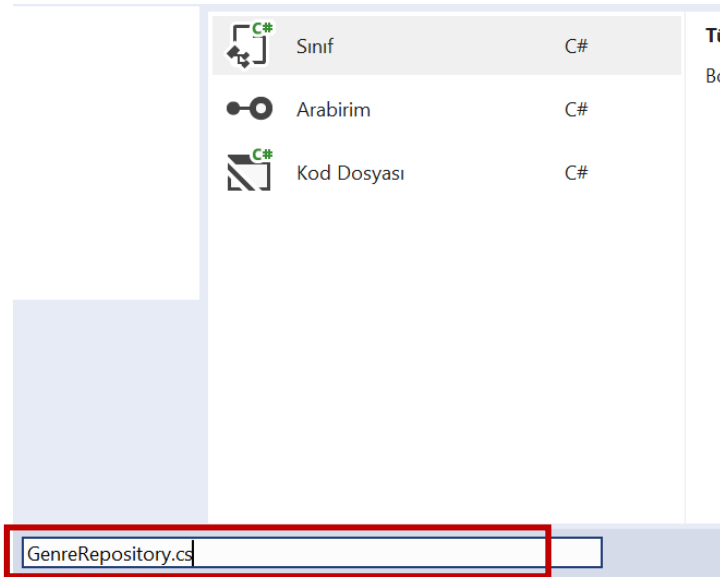
```

1 namespace MovieApp.Web.Models
2 {
3     public class Genre
4     {
5         public int GenreId { get; set; }
6         public string Name { get; set; }
7     }
8 }
9

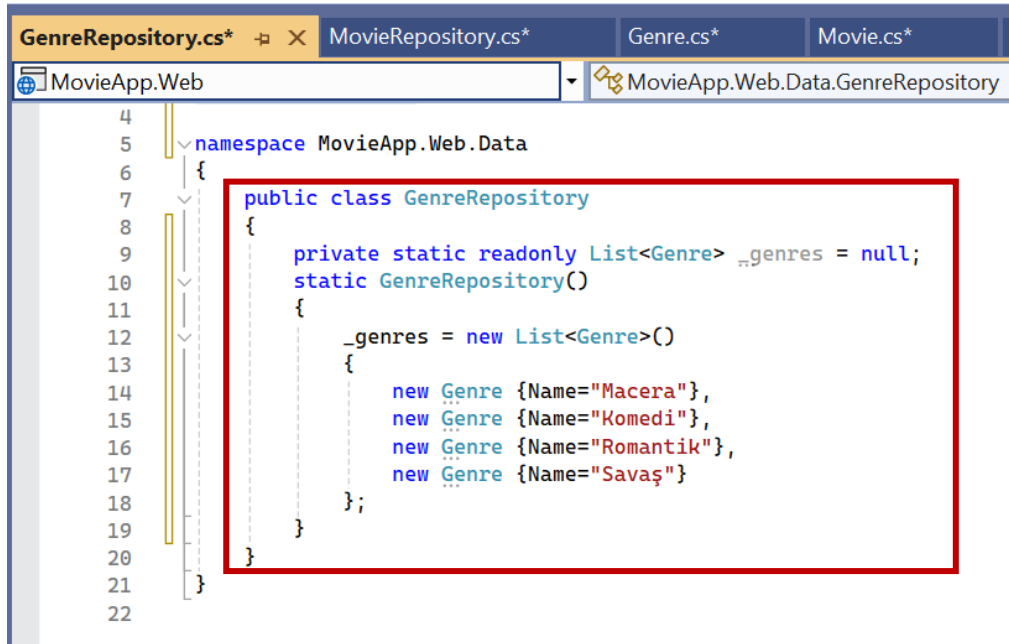
```

Data Klasörü içinde Movie Models için MovieRepository.cs class ı oluşturduk. Şimdi Genre Models'ı için aynı işlemleri tekrarlayalım.

ADIM 10: Data klasörüne sağ tıklanarak yeni bir class oluşturulur. **GenreRepository.cs** olarak isimlendirilir.



ADIM 11: GenreRepository.cs içine de aynı şekilde Genre bilgisini alacak sadece okunabilir bir liste aşağıdaki kod bloğu ile tanımlanır.

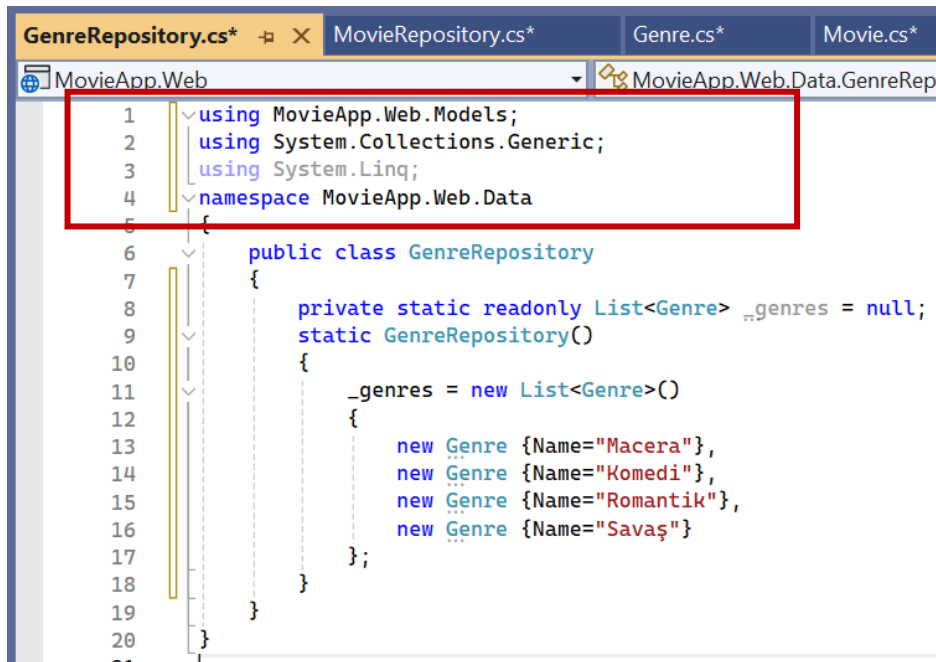


```

4
5 namespace MovieApp.Web.Data
6 {
7     public class GenreRepository
8     {
9         private static readonly List<Genre> _genres = null;
10        static GenreRepository()
11        {
12            _genres = new List<Genre>()
13            {
14                new Genre {Name="Macera"},
15                new Genre {Name="Komedi"},
16                new Genre {Name="Romantik"},
17                new Genre {Name="Savaş"}
18            };
19        }
20    }
21 }
22

```

ADIM 12: Gerekli Kütüphaneler eklenir.

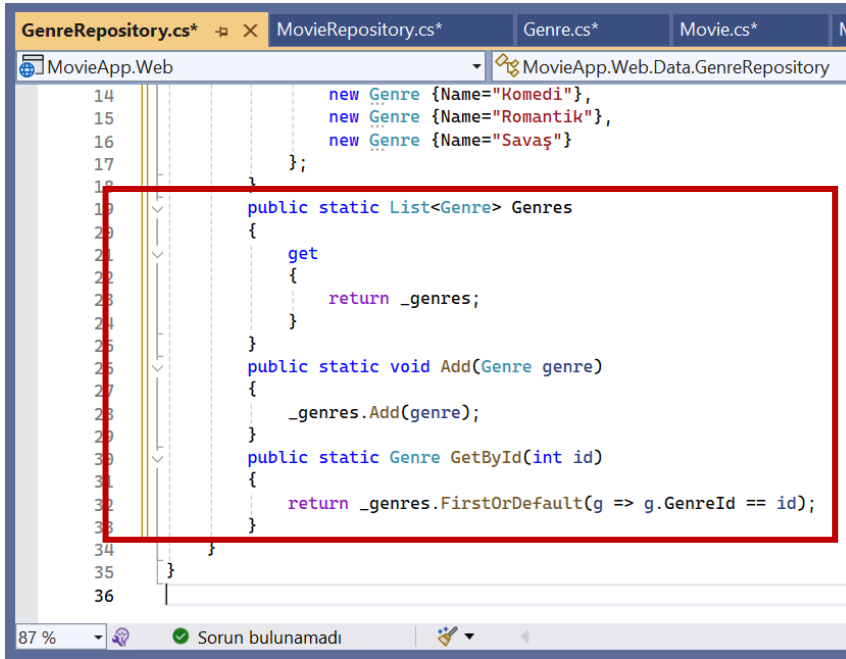


```

1 using MovieApp.Web.Models;
2 using System.Collections.Generic;
3 using System.Linq;
4 namespace MovieApp.Web.Data
5 {
6     public class GenreRepository
7     {
8         private static readonly List<Genre> _genres = null;
9         static GenreRepository()
10        {
11            _genres = 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        }
19    }
20 }
21

```


ADIM 13: İlgili kod bloğu eklenir.

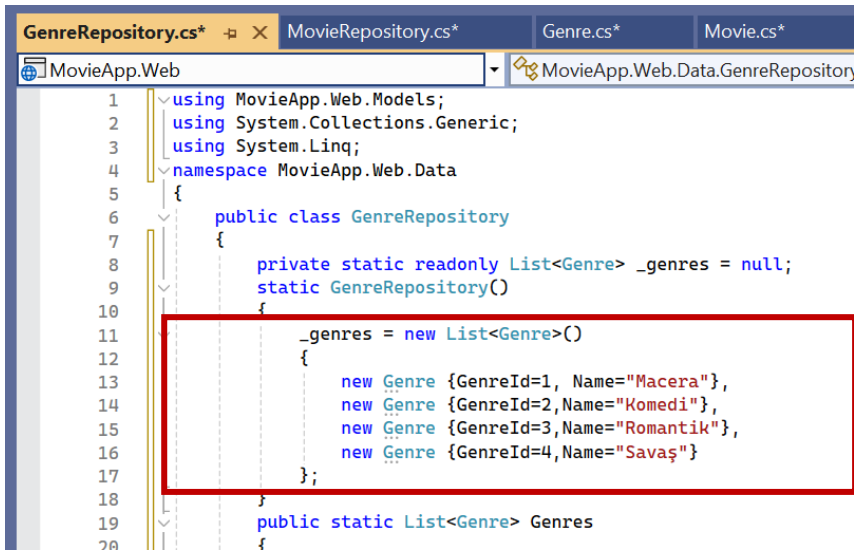


```

14         new Genre {Name="Komed"},
15         new Genre {Name="Romantik"},
16         new Genre {Name="Savaş"}
17     };
18
19     public static List<Genre> Genres
20     {
21         get
22         {
23             return _genres;
24         }
25     }
26     public static void Add(Genre genre)
27     {
28         _genres.Add(genre);
29     }
30     public static Genre GetById(int id)
31     {
32         return _genres.FirstOrDefault(g => g.GenreId == id);
33     }
34 }
35
36

```

ADIM 14: GenreRepository.cs de GenreId bilgileri eklenir.

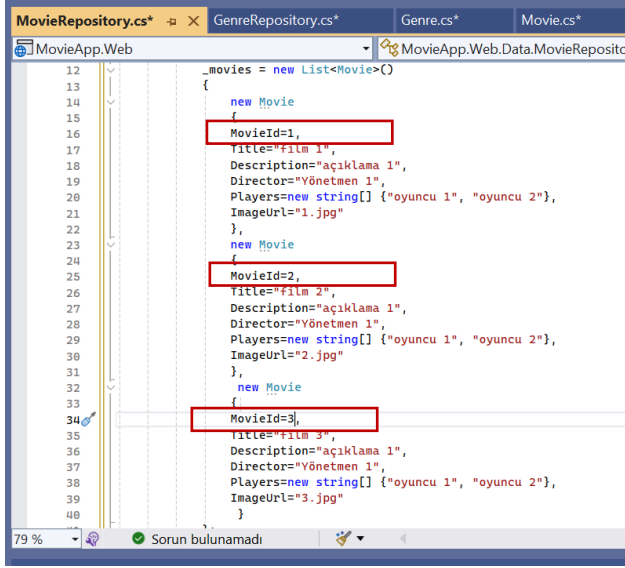


```

1  using MovieApp.Web.Models;
2  using System.Collections.Generic;
3  using System.Linq;
4  namespace MovieApp.Web.Data
5  {
6      public class GenreRepository
7      {
8          private static readonly List<Genre> _genres = null;
9          static GenreRepository()
10         {
11             _genres = new List<Genre>()
12             {
13                 new Genre {GenreId=1, Name="Macera"},
14                 new Genre {GenreId=2, Name="Komed"},
15                 new Genre {GenreId=3, Name="Romantik"},
16                 new Genre {GenreId=4, Name="Savaş"}
17             };
18         }
19         public static List<Genre> Genres
20         {
21

```

ADIM 15: MovieRepository.cs dosyasında ilgili kod bloğuna MovieId ler eklenir.



```
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  }
```

ÇALIŞMA SORUSU

Uygulamada GenresViewComponent metodu ne amaçla kullanılmaktadır açıklayınız.