بسم اللّه الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

افزودن قابلیت جستجو

مدرس : مهندس افشین رفوآ

دوره آموزش MVC

افزودن قابلیت جستجو

در این فصل از مقاله ی آموزشی MVC، نحوه ی افزودن قابلیت Search به متد Index را برای جستجوی movie ها بر اساس ژانر و اسم آن فیلم، شرح خواهیم داد.

بروز رسانی فرم Index

متد Index موجود در کلاس MoviesController را به صورت زیر ویرایش کنید:

```
public ActionResult Index(string searchString)
{
  var movies = from m in db.Movies
         select m;

  if (!String.IsNullOrEmpty(searchString))
  {
    movies = movies.Where(s => s.Title.Contains(searchString));
  }

  return View(movies);
}
```

اولین خط متد Index، برای انتخاب movie ها یک LINQ query ایجاد می کند:

```
var movies = from m in db.Movies
         select m;
}
```

Query در این برهه از زمان تعریف شده، اما هنوز بر روی پایگاه داده اجرا نشده است.

چنانچه پارامتر searchString دربردارنده ی یک رشته باشد، کوئری movie به گونه ای اصلاح می شود که مقدار search string با استفاده از کد زیر فیلتر شود:

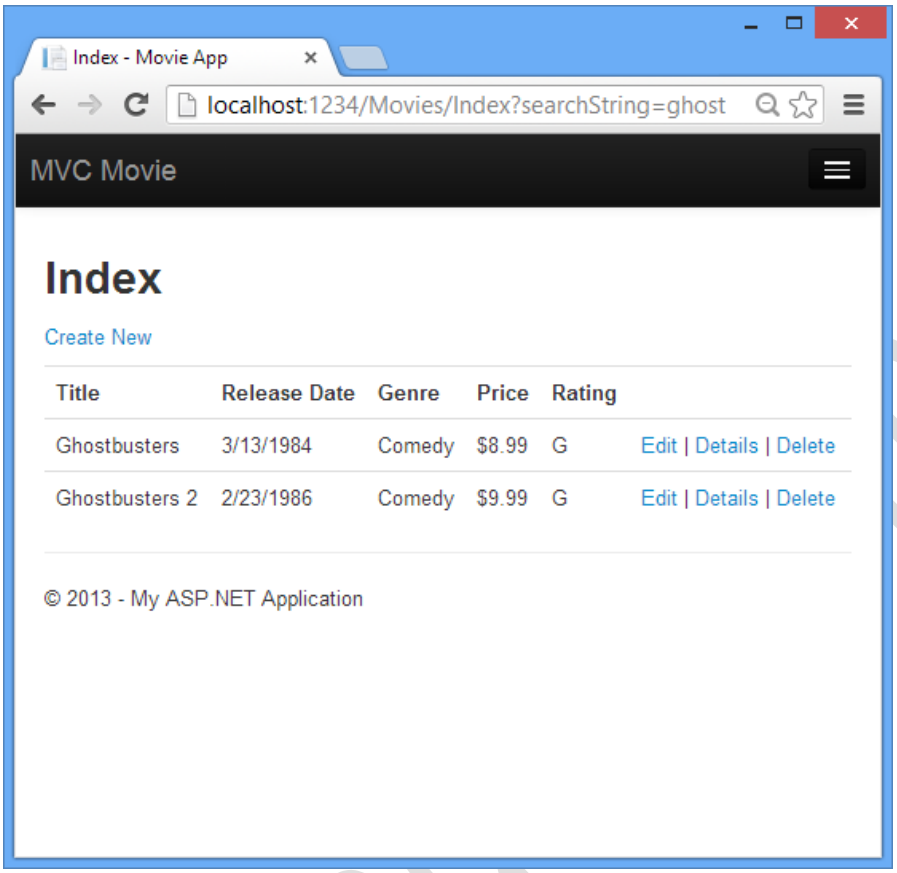
```
if (!String.IsNullOrEmpty(searchString))
  {
    movies = movies.Where(s => s.Title.Contains(searchString));
  }
```

قسمت s => s.Title کد بالا، یک عبارت Lambda است. عبارت های Lambda در LINQ query های مبتنی بر متد، به عنوان آرگومان برای متدهای query operator (مانند تابع where که در مثال فوق بکار گرفته شده) استفاده می شوند. LINQ query ها به محض تعریف شدن یا در زمانی که به واسطه ی فراخوانی یک متد همانند Where یا OrderBy مورد دستکاری قرار می گیرند ، اجرا نمی شود. بلکه اجرای query به تعویق می افتد، بدین معنا که ارزیابی عبارت تا زمانی که متد ToList فراخوانی نشده یا مقدار داخل حلقه نیافتاده و تکرار نشده به تاخیر افتاده و صورت نمی گیرد. در نمونه ی Search، query در Index.cshtml view اجرا می شود.

توجه: متد Contains روی پایگاه داده اجرا شده، نه بر روی کد c# مثال بالا. در پایگاه داده، متد نام برده به SQL LIKE که به کوچک و بزرگی حروف حساس است، نگاشت می شود.

اکنون شما می توانید Index view را که فرم را به کاربر نمایش می دهد، بروز رسانی کنید.

برنامه را اجرا کرده و به /Movies/Index پیمایش کنید. یک query string مانند این نمونه ?searchString=ghost به URL ضمیمه کنید. Movie های فیلتر شده نمایش داده می شوند.



اگر پارامتر ورودی متد Index را متغیری به نام id تغییر دهیم، پارامتر id با {id} placeholder برای مسیرهای تنظیم شده در فایل App_Start\RouteConfig.cs منطبق (match) می شود.

```
{controller}/{action}/{id}
```

متد Index اصلاح نشده:

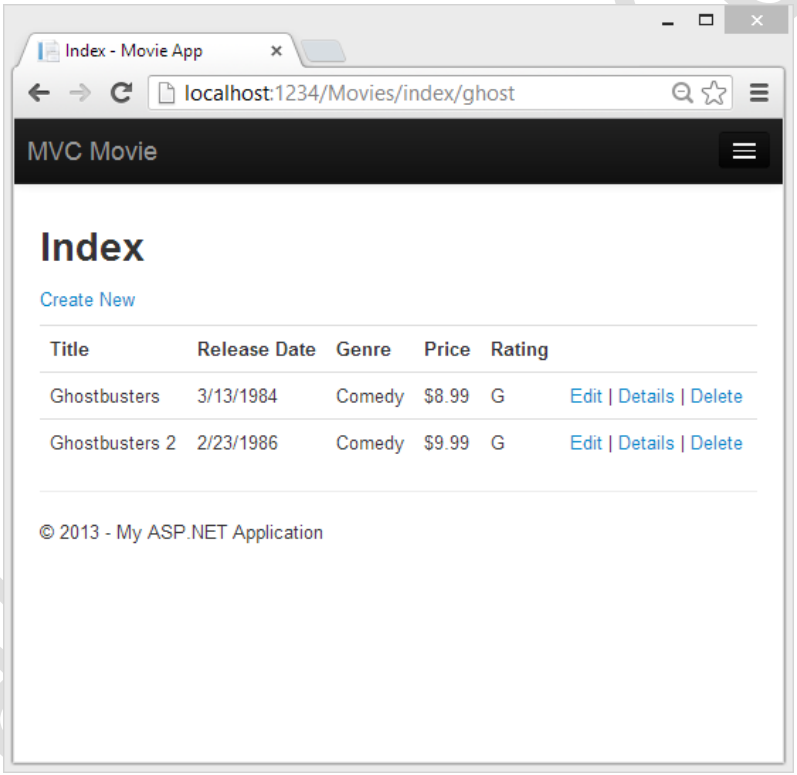
```
public ActionResult Index(string searchString)
{
  var movies = from m in db.Movies
        select m;
  if (!String.IsNullOrEmpty(searchString))
  {
    movies = movies.Where(s => s.Title.Contains(searchString));
  }
  return View(movies);
}
```

متد Index اصلاح شده:

```
public ActionResult Index(string id)
{
```

```
    string searchString = id;
    var movies = from m in db.Movies
                 select m;
    if (!String.IsNullOrEmpty(searchString))
    {
      movies = movies.Where(s => s.Title.Contains(searchString));
    }
    return View(movies);
  }
```

حال می توانید search title را بجای اینکه به عنوان مقدار query string ارسال کنید، آن را به صورت route data (یک بخش از URL) پاس دهید.



اما شما نمی توانید انتظار داشته باشید که کاربران هر بار برای جستجو به دنبال یک فیلم، URL را تغییر دهند. بنابراین شما با افزودن UI به کاربر کمک می کنید فیلم ها را فیلتر کند. اگر ورودی متد Index را برای تست اینکه پارامتر ID، route-bound چگونه ارسال می شود، تغییر داده اید، بایستی آن را به حالت قبلی برگردانید تا متد Index یک پارامتر به نام searchString بپذیرد:

```
public ActionResult Index(string searchString)
    {
      var movies = from m in db.Movies
            select m;
      if (!String.IsNullOrEmpty(searchString))
      {
        movies = movies.Where(s => s.Title.Contains(searchString));
      }
      return View(movies);
    }
```

فایل Views\Movies\Index.cshtml را باز کرده و درست بعد از form ، Views\Movies\Index.cshtml

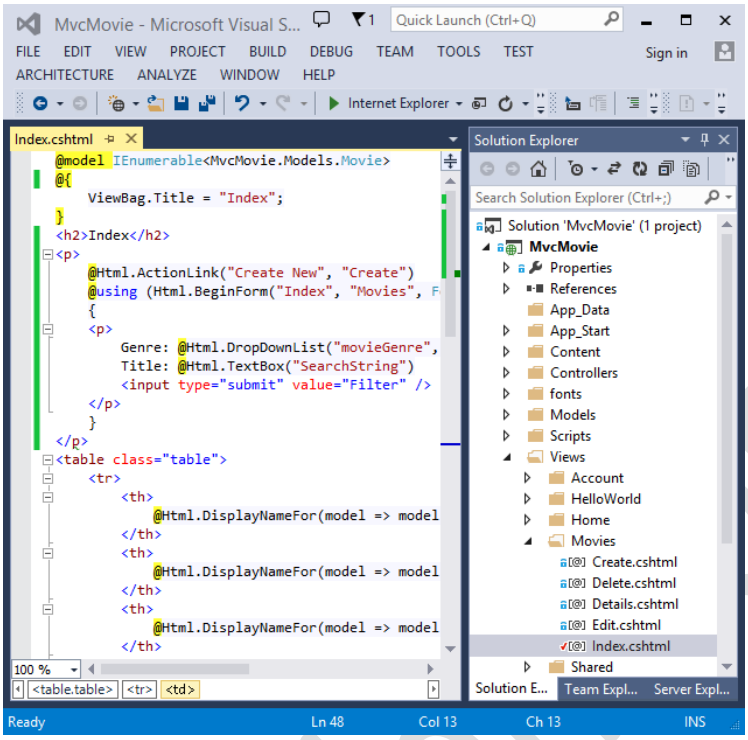
markup رنگی شده ی زیر را اضافه کنید:

```
@model IEnumerable<MvcMovie.Models.Movie>
@{
  ViewBag.Title = "Index";
}
<h2>Index</h2>
<p>
  @Html.ActionLink("Create New", "Create")

  @using (Html.BeginForm())
  {
  <p>
    Title: @Html.TextBox("SearchString") <br />
    <input type="submit" value="Filter" />
  </p>
  }
  </p>
```

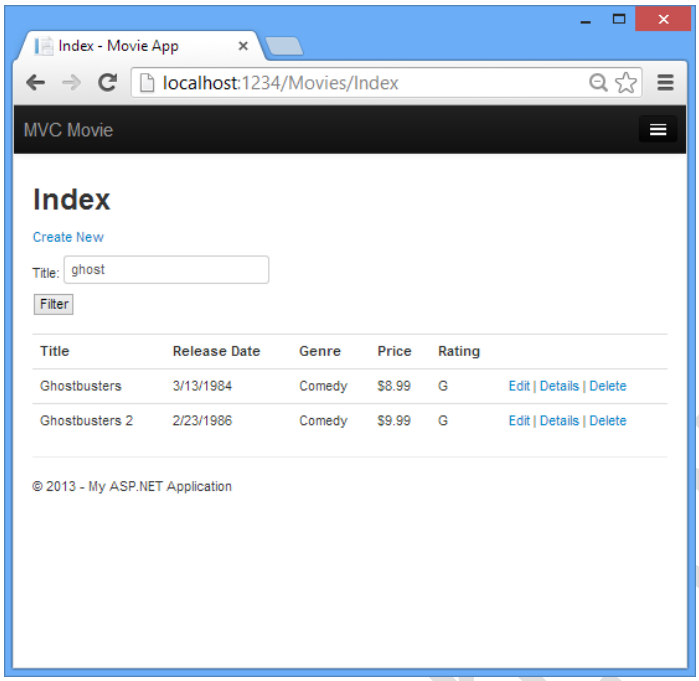
Html.BeginForm یک تگ باز <form> ایجاد می کند. Html.BeginForm باعث می شود هنگامی که کاربر فرم را با کلیک بر روی دکمه ی Filter ارسال (submit) می کند، اطلاعات را به خودش post کند.

Visual Studio 2013 یک قابلیت ویژه برای نمایش و ویرایش فایل های view دارد. هنگامی که یک برنامه را اجرا می کنید در حالی که فایل view باز است، Visual Studio 2013 ، controller action method مناسب را برای نمایش view، صدا می زند.



Index view را در محیط Visual Studio باز کرده، کلیدهای F5+Ctr یا F5 را برای اجرای برنامه فشار دهید،

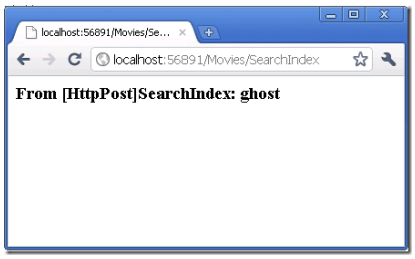
سپس movie مورد نظر را سرچ کنید.



هیچ نسخه ی overload شده ای از خصیصه ی HttpPost متد Index وجود ندارد، به آن نیازی هم ندارید زیرا که متد وضعیت برنامه (application state) را تغییر نمی دهد، بلکه فقط داده ها را فیلتر می کند.

می توانستید متد HttpPost Index را اضافه کنید. در آن صورت، فراخوانده ی تابع (action invoker) با متد HttpPost Index منطبق (match) می شد و متد HttpPost Index به صورتی که در تصویر زیر نمایش داده شده، اجرا می شد:

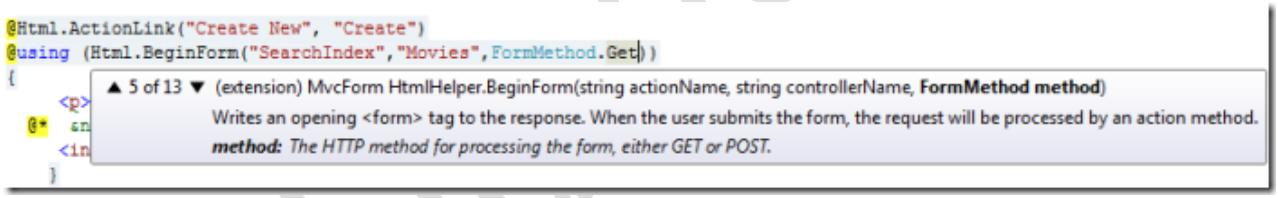
```
    [HttpPost]
public string Index(FormCollection fc, string searchString)
{
  return "<h3> From [HttpPost]Index: " + searchString + "</h3>";
}
```



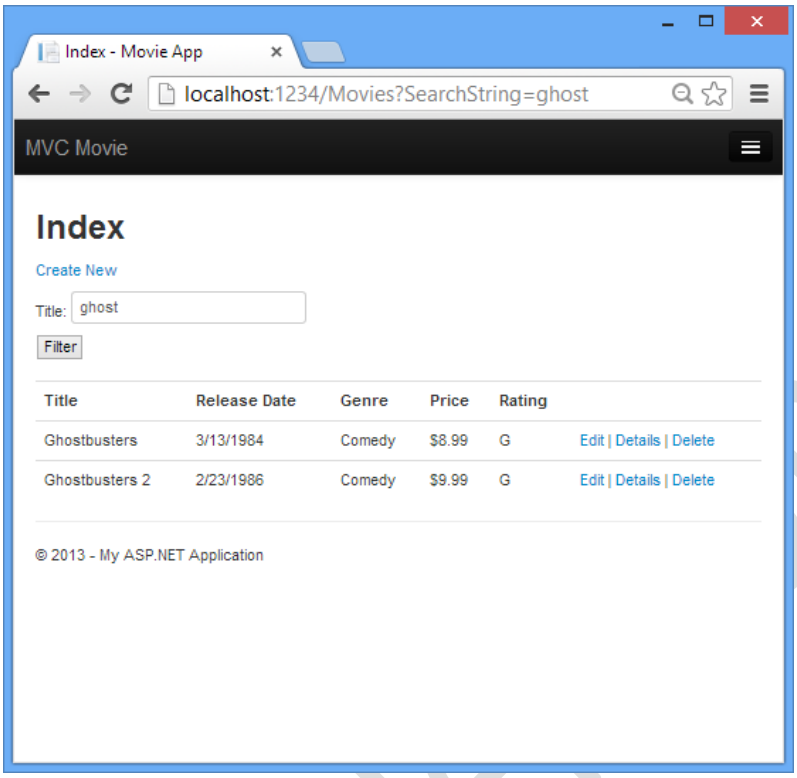
با این وجود، اگرهم این نسخه ی HttpPost متد Index را اضافه می کردید، بازهم در (نحوه ی) پیاده سازی محدودیتی وجود دارد. فرض کنید می خواهید یک search به خصوص را bookmark کنید یا یک لینک را به دوستان خود ارسال کنید تا با کلیک روی آن همان فهرست فیلتر شده از movie ها را مشاهده کنند. دقت داشته باشید که آدرس درخواست HTTP POST با آدرس درخواست GET (localhost:xxxxx/Movies/Index) یکی می باشد – در خود URL هیچ اطلاعات جستجویی وجود ندارد. در حال حاضر، اطلاعات search string به صورت یک مقدار form field به سرور ارسال شده است، بدین معنا که شما نمی توانید آن اطلاعات جستجو را برای bookmark کردن یا ارسال به دوستان در یک URL ضبط (capture) کنید.

راه حل، استفاده از یک نسخه ی overload شده ی متد BeginForm است که مشخص می کند درخواست POST باید اطلاعات جستجو را به URL اضافه کند و اینکه باید به نسخه ی HttpGet متد Index هدایت شود. Markup زیر را جایگزین متد بدون پارامتر BeginForm کنید:

```
@using (Html.BeginForm("Index","Movies",FormMethod.Get))
```



حال زمانی که یک جستجو را انجام و به سرور ارسال می کنیم، یک query string به URL اضافه می شود. جستجو همچنین به متد HttpGet Index فرستاده می شود، حتی اگرهم یک متد HttpPost Index از قبل داشته باشید.



افزودن Search بر اساس Genre

اگر نسخه ی HttpPost متد Index را اضافه کرده بودید، اکنون آن را حذف کنید.

در این بخش یک ابزار یا قابلیت اضافه می کنید که به کاربران اجازه می دهد بر اساس Genre در میان movie ها جستجو انجام دهند. برای این منظور متد Index را با کد زیر جایگزین کنید:

```
public ActionResult Index(string movieGenre, string searchString)
{
  var GenreLst = new List<string>();
  var GenreQry = from d in db.Movies
           orderby d.Genre
           select d.Genre;
  GenreLst.AddRange(GenreQry.Distinct());
  ViewBag.movieGenre = new SelectList(GenreLst);
  var movies = from m in db.Movies
        select m;
  if (!String.IsNullOrEmpty(searchString))
  {
    movies = movies.Where(s => s.Title.Contains(searchString));
```

```
    }
    if (!string.IsNullOrEmpty(movieGenre))
    {
        movies = movies.Where(x => x.Genre == movieGenre);
    }
    return View(movies);
}
```

این نسخه ی متد Index یک پارامتر اضافی می گیرد، که آن پارامتر movieGenre می باشد. اولین خطوط کد، یک شی List برای نگه داشتن movie genre ها از پایگاه داده ایجاد می کنند.

کد زیر یک LINQ query است که تمامی genre ها را از پایگاه داده بازیابی می کند.

```
var GenreQry = from d in db.Movies
               orderby d.Genre
               select d.Genre;
```

کد با استفاده از متد AddRange مجموعه ی ژنریک List، تمامی genre های مجزا و متمایز (distinct) را به لیست اضافه می کند. (بدون تعریف کننده ی Distinct، genre های تکراری اضافه می شد، برای مثال ژانر comedy دوبار به نمونه ی مورد نظر ما افزوده می شد). کد مورد نظر سپس فهرستی از genre ها را در شی ViewBag.movieGenre ذخیره می کند. برنامه های نوشته شده با MVC، معمولا داده های دسته بندی مانند ژانر یک فیلم را به صورت یک شی SelectList در ViewBag ذخیره می کنند، سپس درون یک کادر فهرست کشویی (dropdown list box) به داده های دسته بندی دسترسی پیدا می کنند.

کد زیر نحوه ی بررسی پارامتر movieGenre را نمایش می دهد. در صورت خالی نبودن، کد مزبور query به نام movies را تا آنجایی محدود می کند که فهرست movie های انتخابی به genre تعریف شده محدود شده و فقط ژانر مورد نظر بازیابی شود.

```
if (!string.IsNullOrEmpty(movieGenre))
{
    movies = movies.Where(x => x.Genre == movieGenre);
}
```

همان طور که پیشتر گفته شد، query تا زمانی که لیست movies داخل حلقه نیفتاده و تکرار نشده (این امر در view و پس از اینکه متد Index بازمی گردد، رخ می دهد) بر روی پایگاه داده اجرا نمی شود.

افزودن Markup به Index View برای پشتیبانی از قابلیت جستجو بر اساس genre

یک Html.DropDownList helper درست پیش از TextBox helper به فایل

Views\Movies\Index.cshtml اضافه کنید. Markup کامل زیر نمایش داده شده است:

```
@model IEnumerable<MvcMovie.Models.Movie>
@{
  ViewBag.Title = "Index";
}
<h2>Index</h2>
<p>
  @Html.ActionLink("Create New", "Create")
  @using (Html.BeginForm("Index", "Movies", FormMethod.Get))
  {
  <p>
    Genre: @Html.DropDownList("movieGenre", "All")
    Title: @Html.TextBox("SearchString")
    <input type="submit" value="Filter" />
  </p>
  }
```

در کد زیر:

```
@Html.DropDownList("movieGenre", "All")
```

پارامتر "movieGenre" با ارائه دادن کلید به DropDownList helper به آن کمک می کند،

IEnumerable<SelectListItem> را در شی ViewBag بیابد. شی ViewBag در action method پر

شده (populated):

```
    public ActionResult Index(string movieGenre, string searchString)
{
  var GenreLst = new List<string>();
  var GenreQry = from d in db.Movies
           orderby d.Genre
           select d.Genre;
  GenreLst.AddRange(GenreQry.Distinct());
  ViewBag.movieGenre = new SelectList(GenreLst);
  var movies = from m in db.Movies
         select m;
  if (!String.IsNullOrEmpty(searchString))
  {
    movies = movies.Where(s => s.Title.Contains(searchString));
  }
  if (!string.IsNullOrEmpty(movieGenre))
```

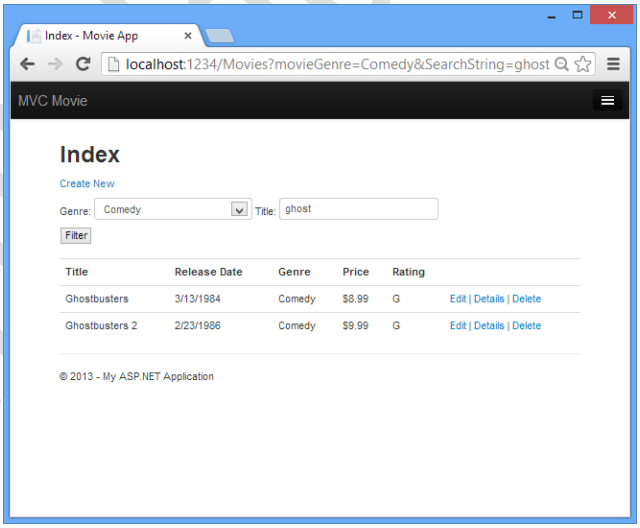
```
    {
        movies = movies.Where(x => x.Genre == movieGenre);
    }
    return View(movies);
}
```

پارامتر "All" آیتمی که باید در لیست از پیش انتخاب شود را فراهم می نماید. اگر از کد زیر استفاده می کردیم:

```
@Html.DropDownList("movieGenre", "Comedy")
```

یکی movie با ژانر "Comedy" در پایگاه داده ی خود داشته و "Comedy" در فهرست کشویی (dropdown list) از پیش انتخاب می شد. به این خاطر که ژانر "All" را نداریم، در SelectList ژانری به نام "All" وجود ندارد. بنابراین زمانی که بدون انتخاب، post back می کنیم، مقدار متغیر movieGenre query string در واقع تهی می باشد.

برنامه را اجرا کرده و به /Movies/Index پیمایش کنید. ابتدا جستجو را بر اساس genre، سپس با movie name و در آخر با هر دو معیار انجام دهید.



در این بخش با کمک هم یک view و یک action method برای جستجو ایجاد کردیم که به کاربران اجازه می دهد بر اساس عنوان فیلم (movie title) و ژانر فیلم ها را سرچ کنند. در مبحث بعد به نحوه ی افزودن یک خاصیت به Movie و همچنین اضافه کردن یک initializer که خود به صورت پیش فرض یک پایگاه داده ی آزمایشی ایجاد می کند خواهیم پرداخت.