شروع کار با

ASP.NET MVC 5

چگونه در یک روز برنامه نویس شویم؟!

نویسنده: مرتضی پورمحمد

زمان انتشار: آبان ۱۳۹۳

☑ سخن نویسنده

بعد از دو پروژه ی نا موفق و ادامه پیدا نکردن دو کتاب پیشین تصمیم گرفتم تا این کتاب حاضر را به نگارش درآورم و پس از اتمام کامل منتشر نمایم. وقتی شروع به نگارش MVC3 کردم اصلاً خبر نداشتم که یک دوست و همکار عزیز در حال ترجمه ی کتابی است که بنده به عنوان ریفرنس استفاده می کردم. وقتی کتاب ایشان منتشر شد با کم لطفی دوستان مواجه شدم در صورتیکه تاریخ های شروع پست های بنده خیلی قبل تر از شروع ترجمه ی آقای مهندس بهروز راد بود. البته دوستان زیادی هم به بنده لطف داشتند و پیگیر ادامه ی آن نوشته ها بودند. اما بسیار متاثر از اتفاقات رخ داده بودم و همچنین درگیری های زندگی بسیار راه را برای بنده سخت و دشوار نمود. تا اینکه توفیق اجباری دست داد تا دوباره شروع به آموزش کنم. پروژه ای پیچیده تحت MVC پیشنهاد شد. وقتی پروژه را قبول کردم نگاهی به MVC انداختم و شاهد تغییرات زیادی در نسخه ی ۵ این تکنولوژی محبوب شدم. تغییرات مانند ساختمانی است که پایه هایش تغییر چندانی نکرده (و مطمئناً قویتر شده) اما در طبقاتش بسیار تغییرات اتفاق افتاده. به خصوص که فریمورک Entity یک آپدیت جدید به همراه تغییرات بزرگ داشت و از طرفی مایکروسافت در نسخه ی جدید MVC تکنولوژی Scaffold را که در نسخه ی ۳ معرفی کوتاه داشت در این نسخه بسیار بزرگ و قویتر نمود به نوعی که ادعا می کند شما اگر ساختار را درست بنویسید بقیه ی کدها را ما برایتان می نویسیم. و کلی مطلب دیگر که همه و همه دست به دست هم داد تا دوباره به این عرصه بازگردم و البته کمی محتاط تر. در ابتدا نیز توضیح دادم تا اتمام کتاب آنرا منتشر نخواهم کرد! و پس از انتشار این کتاب به فاصله ی کمتر از یک ماه کتابی دیگر در ادامه ی این کتاب منتشر خواهم کرد که به مباحث پیچیده تر MVC پرداخته می شود. هنگامی که این مطالب را می نویسم کتاب دیگری که توضیح دادم تقریباً به انتهای خود رسیده است. اگر عمری باقی ماند فصل آخر آن کتاب را نیز می نویسم و سپس این کتاب را منتشر می کنم. مثل همیشه از اینکه در این مدت کنارم بودید و با تماس هایتان دلگرمی به من میدادید بسیار سپاسگذارم. شیرین ترین لحظات زندگی حرفه ای کاری را شما عزیزان با ابراز محبتتان برایم ساختید. همین که پیگیر کتاب بودید و یا پیام تبریک و تشکر میفرستادید برایم از هزاران درامد مالی با ارزش تر بود. نگارش این کتاب ها تنها با نیت پیشرفت جامعه ی علمی کشور است و نه حتی شهرت! به پیشنهاد دوستان این کتاب را نیز مانند دیگر کتاب هایم با زبان محاوره به نگارش در می آورم و در اینترنت به صورت کاملاً رایگان به همراه سورس کدها، منتشر می کنم.

همیشه گفتم دانش بنده بسیار محدود است. لطفاً اگر جایی از کار مشکل دارد حتماً متذکر شوید تا اصلاح کنم. انتقادات شما باعث پیشرفت بنده ی حقیر می شود. راه های دسترسی به من بسیار راحت است. یا می توانید از طریق پست الکترونیک farjadp@live.com و یا تماس با شماره تلفن ۰۹۱۲۲۸۳۰۷۹۵ بنده پاسخگوی شما بزرگواران خواهم بود.

صفحات شبکه های اجتماعی بنده نیز

http://ir.linkedin.com/pub/farjad-pourmohammad/45/257/8a2/

https://www.facebook.com/farjadp

https://twitter.com/farjadp

☑ مقدمه

شما در این کتاب پیش رو با مباحث MVC5 در قالب یک پروژه ی فروشگاهی آشنا می شوید. در کل این کتاب سعی می شود تا مباحث به زبان بسیار ساده عنوان شود تا عزیزان و علاقه مندانی که حتی آشنایی کوچکی با دات نت ندارد بتوانند در روند انجام پروژه با قدرت باقی بمانند! البته حالت پیشرفته ی فروشگاه را در کتاب بعدی که در خصوص EF می باشد به سرانجام خواهم رساند.

پروژه ی فروشگاهی حاضر به صورت برنامه وبسایتی (Web App) توسط زبان برنامه نویسی C# و MVC نسخه ی ۵ که جدیدترین و آخرین نسخه ی این زبان برنامه نویسی تحت وب می باشد، پایه ریزی و انجام می شود. همچنین برای انجام این پروژه به Visual Studio 2013 و SQL Server 2014 نیاز است و ما نیز توسط این دو نرم افزار پروژه را انجام می دهیم. همچنین در پایان این آموزش کل کدهای نوشته شده به صورت کاملاً رایگان و کدباز در اختیار دوستان قرار داده می شود. در طول پروژه به علت رابطه ی تنگاتنگ MVC5 و EF6 به صورت خیلی کوتاه به این فریمورک توسعه یافته و بسیار قدرتمند مایکروسافت نیز خواهیم پرداخت.

مطمئناً اولین سوالی که برایتان پیش میاید این است که چرا باید MVC را برای اجرایی کردن پروژه ی خود انتخاب کنید؟!! چون یک محصول تجاری بسیار خوب است که کمپانی بزرگ مایکروسافت به شدت از این پروژه حمایت می کند! کافی نیست؟! چون معماری به شدت کاربر پسند دارد و برنامه در پاسخ به هر سوال یا درخواست کاربر بهترین و مناسب ترین پاسخ را می دهد! هنوز کافی نیست؟ چون برنامه نویسان از سردرگمی کدنویسی در لایه های مختلف N-Layer رها می شوند! باز هم کافی نیست؟ چون در نسخه ی جدید بسیاری از کدها توسط Scaffold نوشته میشه! فکر کنم دارید قانع میشید. مطمئن باشید اگر با نحوه ی کار MVC آشنا بشید به شدت عاشق این کدنویسی و تکنولوژی میشید.

یکی از مهمترین نکته هایی که در خصوص این تکنولوژی میتونم اشاره کنم اینه که ASP.Net MVC مبتنی بر پلت فرم Net. هست پس قابلیت نوشتن کد در هر زبان مبتنی بر پلت فرم Net. وجود داره. یعنی شما میتونید یه قسمت از برنامه رو بر فرض با F# بنویسید بدون اینکه مشکلی وجود داشته باشه! و یا اینکه از موتور Razor استفاده می کنه. و یا حتی پشتیبانی قویتر و موثرتر از قابلیت تزریق وابستگی یا همون Dependency Injection . پشتیبانی به شدت قوی از JavaScript و فرمت تبادل داده ای JSON بر مبنای JQuery از مزیت های دیگه ی این تکنولوژی بسیار محبوب هست . و هزاران نکته ی دیگه که کافیه توی اینترنت سرچ کنید تا با خصوصیات این تکنولوژی آشنا بشید.

در نهایت به دوستان و علاقه مندان پیشنهاد می شود با پیشنیاز های زیر به سراغ این کتاب آموزشی بیایند

- HTML
- CSS
- JQuery
- آشنایی با اینترنت! :دی

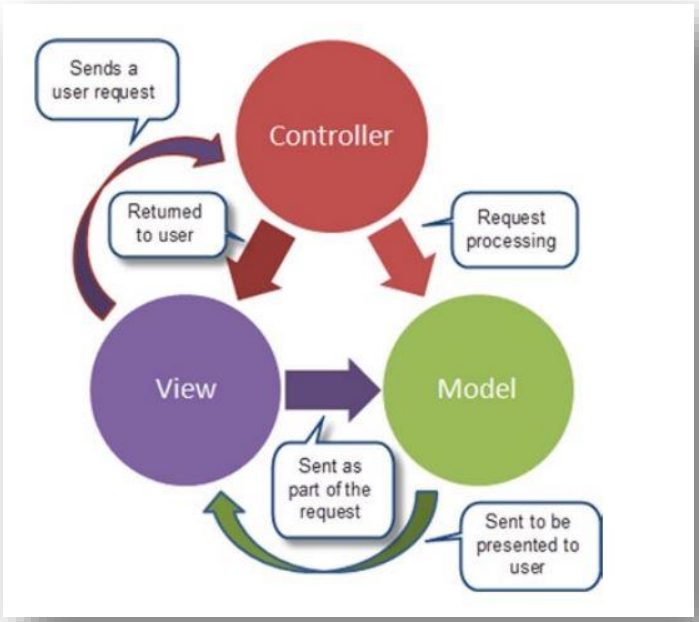
☑ معماری MVC

در کتاب اولم مفصل در خصوص MVC صحبت کردم. به نوعی که حتی رفرنس خیلی از کتاب های دیگه شد. پیشنهاد می کنم به صفحات اولیه ی اون کتاب مراجعه کنید. در اینجا خلاصه ای از MVC میگم بعد به سراغ معماری MVC میرم و بدن معطلی وارد مبحث اصلی میشم. دوست ندارم مثل خیلی از کتاب ها حرف های تکراری بزنم و چندین صفحه ی اول کتاب رو صرف چگونگی نصب نرم افزار و تاریخچه ی اینترنت برم!

MVC یک تکنولوژی سه لایه هست. یعنی این قضیه توی اسمش معلومه اما بخوام فنی تر صحبت کنم میتونم بگم الگوی معماری ام‌وی‌سی به جداسازی داده‌های کاربرد (از جمله محتویات بخش مدل) از مؤلفه‌های ارائه شده به‌صورت گرافیکی (بخش نما یا همون UI) و منطق مربوط به پردازش ورودی‌ها (بخش کنترل‌گر) اقدام می‌کنه. هدف الگوی معماری MVC صرفاً یکپارچگی در معماری نرم‌افزار است و به کمک اون بدست گیری نرم‌افزار در راستای مدیریت و توسعه به راحتی آب خوردن انجام می‌گیره.

- **Model:** شامل مدل داده هایی است که کاربر با آنها سر و کار داره. این مدل می تونه شامل کلاس هایی که از آنها با عنوان View Model یاد می شه باشه یا Domain Model باشه. از View Model ها برای انتقال ساده ی اطلاعات بین View و کنترلر استفاده می شود.
- **Views:** برای نمایش فرم های واسط کاربری با بهره گیری از اطلاعات مدل استفاده می شود.
- **Controllers:** در واقع Controller بقیه کارها رو انجام میده. اینکه چه درخواستی از کاربر رسیده، چه درخواستی از Model باید بشه، اطلاعات گرفته شده از Model چطور باید پردازش بشه، چه Viewای باید انتخاب بشه و چه اطلاعاتی به View باید ارسال بشه.

وعکس زیر از کتاب ASP.Net MVC4 انتشارات Apress تکمیل کننده ی این بحث است



یک توضیح بسیار مهم لازمه که اینجا بدم : اگر این چند صفحه ی اول رو متوجه نمیشید زیاد نگران نباشید و اصلاً نترسید . در طول کتاب با ذکر مثال هر کدوم رو توضیح میدم

❖ Domain Model چیست؟

اما یک مبحث بسیار مهم اینجا مطرح شد و اون Domain Model ها هستند. Domain Modelها کلاس هایی هستند که منطق انجام کار، قوانین تعیین اعتبار و تبدیل داده ها به فرمت مناسب و کارهایی از این قبیل در آنها انجام میگیره. به طور کلی تر Domain Modeling کار شناسایی نیازهای اساسی پروژه رو بر دوش داره. عموماً Domain Model ها مجموعه ای از موارد زیر هستند :

- Domain Type ها که به طور کلی میتونیم بگیم کلاس های خاصی از C#
- متدهایی که کار عملیات ها رو در قسمت Domain Type ها به عهده دارن
- قوانینی که در Domainباید وجود داشته باشه. این قوانین به وسیله ی متدهای مورد قبل انجام میگیرند

به طور خلاصه Domain Model نمایی کلی از تمام منطق برنامه و قوانین سیستم هست. و چون تمام منطق سیستم در یک نقطه متمرکز میشه (توسط همین Domain Model) پس اگر جایی از پروژه به مشکل برخوردیم فقط سراغ همین DM میریم وقسمتی که ایراد داره رو دیباگ می کنیم. پیشنهاد هم میکنم این DM رو در یک اسمبلی جدا قرار بدید که راحت ارجاع داده بشه.

در Asp.Net MVC سه قابلیت برای کار با Domain Model وجود دارد:

- Model Binding: همون ویژگی که در View از آن استفاده می کنید. (ارسال و دریافت پراپرتی ها به صورت یک مدل) و یا به گفته ای دیگه مدل بایندینگ یک قابلیت قرارداد محور هست که پروپرتی های کلاس را با استفاده از داده هایی که از سمت کلاینت می آیند را پُر می کند.
- Model Metadata: روشهایی مانند [Display] که در مدل از آن استفاده می کنید. و یا اجازه ی اضافه کردن توضیحات بیشتر در مورد اجزای کلاس را میدهد.
- Validation: اعتبارسنجی که در مدل انجام می دهید. که این اعتبار سنجی در قسمت Model Binding انجام میشه.

که به این موارد اشاره شده در پاراگراف بالا متدولوژی Domain Driven Design یا DDD میگن.

❖ مفهوم Aggregate

Aggregateبه معنای تجمیع هست. اصولاً من ا ترجمه ی بعضی از اصطلاحات مشکل دارم. اگر در طول کتاب از واژه ی انگلیسی استفاده کردم منظورم همین مفهوم هست. Aggregate وظیفه ی دسته بندی موجودیت های یک مدل رو بر عهده داره.

❖ مفهوم Simplification

Simplification به معنای ساده سازی است. یعنی چطوری دیتابیس و مراحل انجام کار رو ساده سازی کنیم که در صورت یک رخداد بقیه ی اجزا آسیب نبینن. مثلاً در یک دیتابیس چند Table که به همدیگه وصل هستن، چجوری آپدیت یا حذف اطلاعات داشته باشیم که به بقیه ی اطلاعات آسیب وارد نشه . البته این دو مفهوم بسیار با یکدیگر معنا و مفهوم برنامه نویسی پیدا می کنن و به نظر شخصیم نمیشه جدا معنیشون کرد.

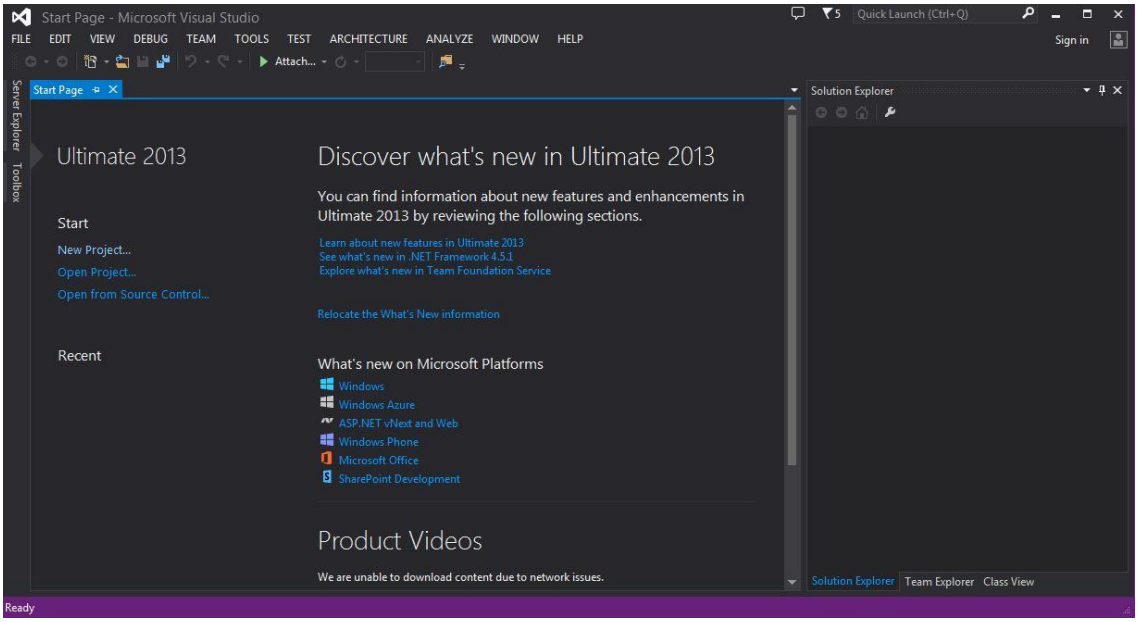
❖ مفهوم Repository

مهمترین بخش هر نرم افزاری مسلماً ذخیره ی اطلاعاته. که ما در اینجا از SQL Server به عنوان این مهم استفاده می کنیم. یعنی زمانی میرسه که ما نیاز داریم تا با داده هایی که در موجودیت های Domain Model قرار دارند کار کنیم. انجام عملیات بر روی داده ها ربطی به Domain Modelنداره و در متدلوژی DDD به عنوان یک جزیره یا قسمت مجزا بهش نگاه میشه. به عبارت ساده تر یعنی نباید کدهای لایه ی دسترسی به داده ها را با کدهای Domain Model یکی کنیم. پس باید چیکار کنیم و چجوری به داده ها دسترسی پیدا کنیم؟! متداولترین راه استفده از Repository هاست. درRepository ها متدهایی که باعث انجام اعمال متداول بر روی داده ها می شوند تعیرف میشن. یعنی در Domain Model متد هایی که در Repositoryوجود دارن و نوشته شده هستند رو فراخوانی می کنیم. یک قرارداد استاندارد هم وجود داره ، به ازای هر Aggregate یک کلاس Repository مینویسیم. به همین راحتی و مطمئنی. اینجوری احتمال نفوذ به دیتابیس هم بسیار محدود میشه و به حداقل ترین شکل ممکن نزول میکنه.

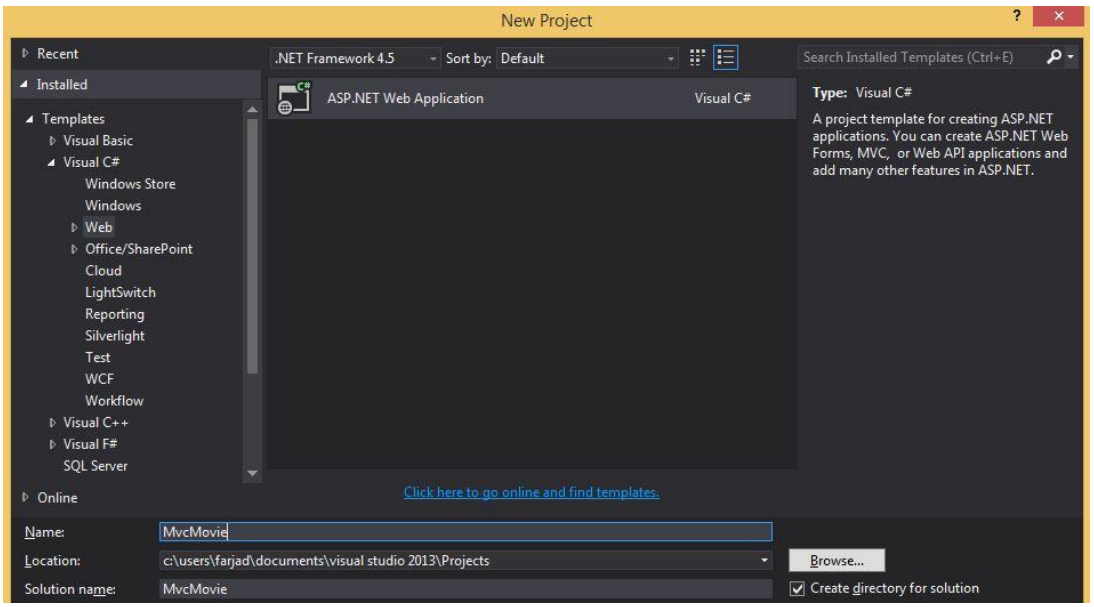
☑ شروع پروژه و استفاده از Controller

❖ قدم صفر!

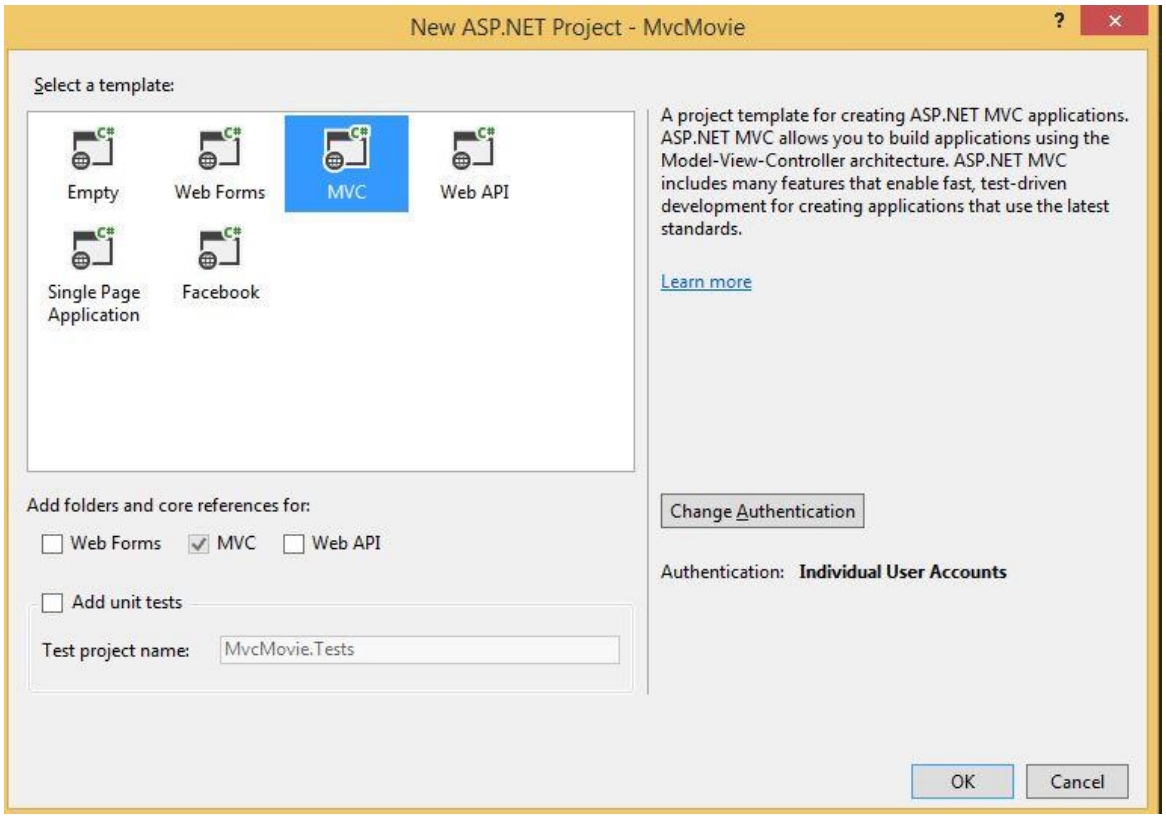
قاعتاً برای شروع باید ویژوال استودیو را اجرا کنیم. وقتی برای بار اول ویژوال استودیو رو باز می کنید با شکل زیر روبرو میشید.



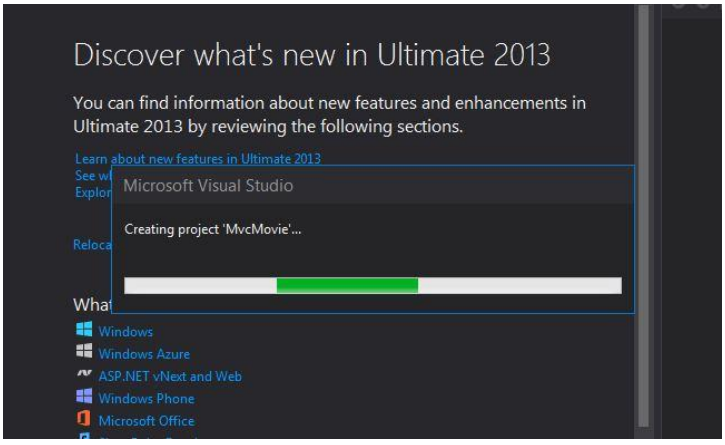
برای ایجاد پروژه ی جدید باید از طریق منوی File>New>Project... را انتخاب کنید و یا توسط کلید های ترکیبی CTRL + Shift + N می توانید به این پنجره دسترسی پیدا کنید.



برای شروع می بایست طبق تصویر C# را انتخاب و نام MvcMovie را در قسمت نام پروژه وارد می نماییم. در مرحله ی بعد شما می توانید نوعی که میخواهید انجام دهید را انتخاب نمایید که ما در اینجا MVC را انتخاب می کنیم. در خصوص دسترسی ها و یونیت تست بعداً صحبت خواهیم کرد.

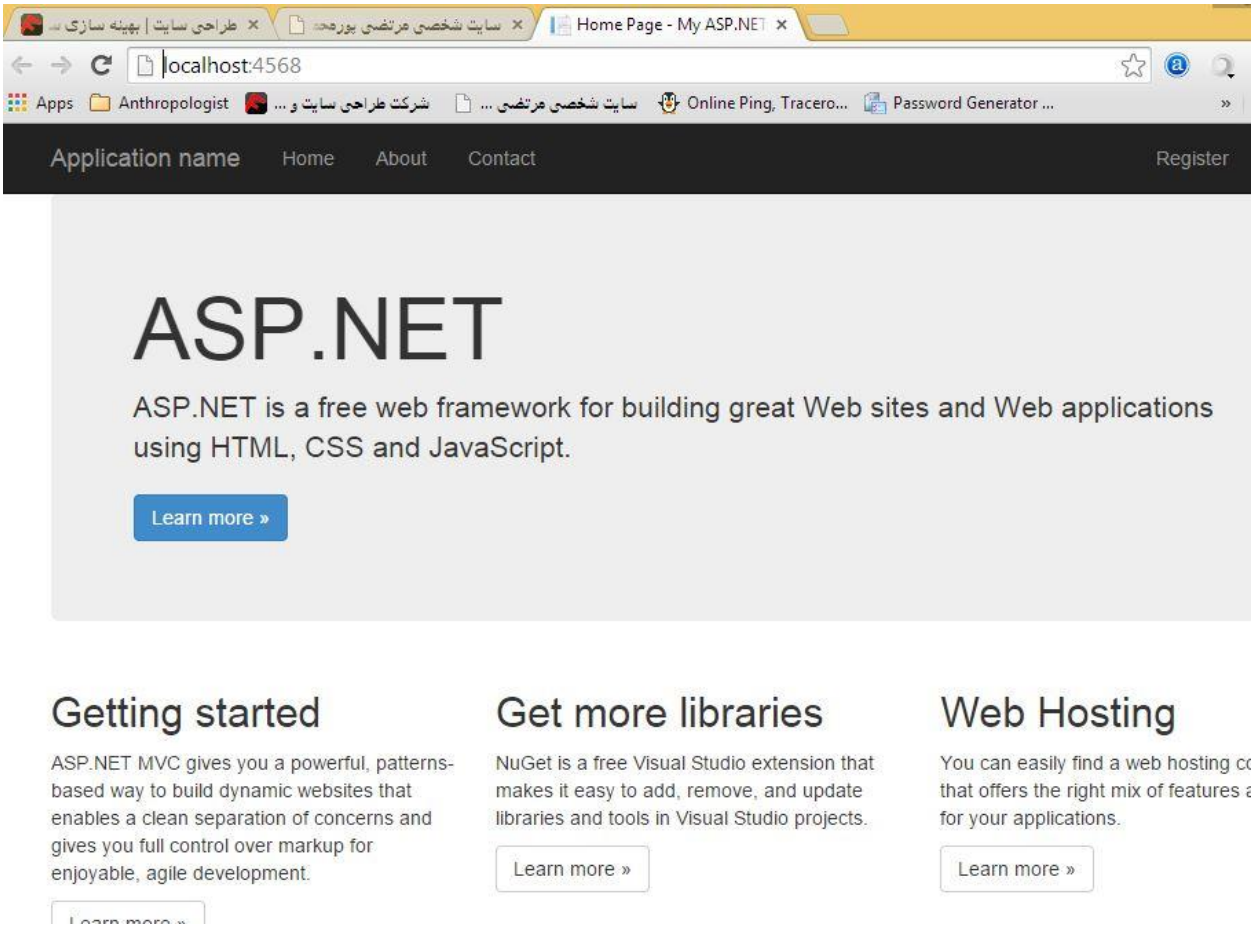


اگر OK کنید ویژوال استودیو شروع به آماده سازی پروژه ی MVC می نماید.



خب تبریک می گویم. شما اولین پروژه ی خود را اجرا کردید. ویژوال استودیو برای شما همه چیز را مهیا کرده است. این همان پروژه ای است که قبلاً برنامه نویسان تحت عنوان "سلام دنیا ! Hello World" میشناختند. مایکروسافت این مرحله را برای برنامه نویسان آسان کرده و خود این برنامه را اجرا می کند :دی

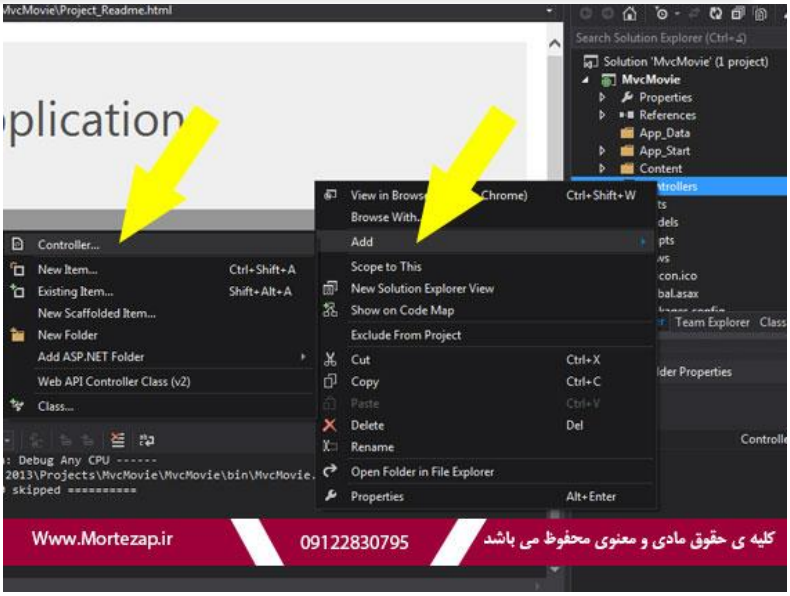
برای اجرای پروژه می توانید کلیدهای ترکیبی CTRL + F5 را فشار دهید تا ببینید مایکروسافت برایتان چه چیزی آماده کرده است! آیا همچنان پیغام معروف است؟!



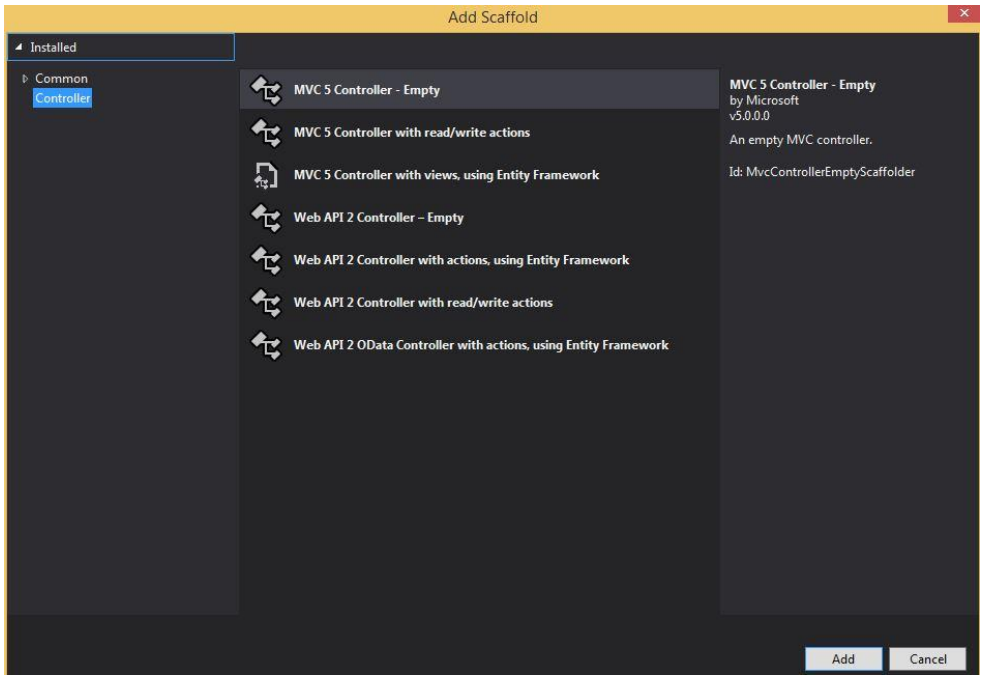
به نظرم که این کار مایکروسافت خیلی بهتر از جمله ی معروف سلام جهان است! و مطمئناً تمامی لینک ها نیز به درستی کار می کند. حتی صفحه ی ثبت نام! امتحان کنید . . . و جالبتر از همه اینکه این صفحه کاملاً ریسپانسیو است. این را نیز امتحان کنید.

❖ قدم اول : اضافه کردن Controller

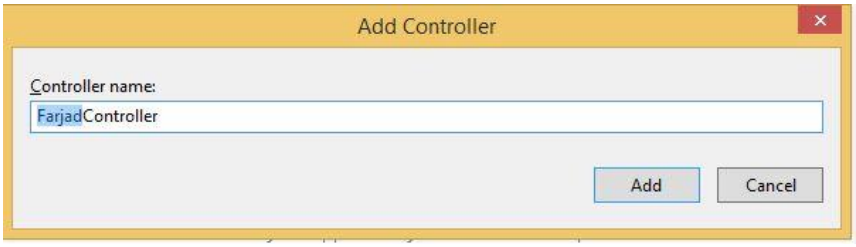
حالا آماده باشید که اولین قدم رو در این راه برداریم. به ویژوال استودیو برگردید و از قسمت Solution Explorer بر روی فولدر Controller سمت راست کلیک کرده، از قسمت ADD گزینه ی ...Controller را انتخاب نمایید. و از پنجره ی باز شده شما می توانید حالت های مختلف کنترلرهای MVC را انتخاب نمایید. ما در اینجا همان گزینه ی اول یعنی EMPTY را انتخاب می نماییم



مانند شکل زیر گزینه ی MVC 5 Controller – Empty را بر میگزینیم و بر روی دکمه ی Add کلیک می نماییم



در پنجره ی حاضر نامی را برای این کنترلر انتخاب می کنیم. بنده چون زیادی نارسیسیم (خوبینی زیاد) دارم اسم خودم رو تایپ می کنم. در نهایت روی Add کلیک کنید.



همانطور که ملاحظه می نمایید اولین کنترلر شما به فولدر Controller اضافه شد . این فایل شامل کدهای پیش فرضیست که ما برخی از انها را عوض می کنیم. و کدهای زیر را به ان اضافه می نماییم.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MvcMovie.Controllers
{
    public class FarjadController : Controller
    {
        //
        // GET: /Farjad/
        public string Index()
        {
            return "که میبینید</br> دی: کنیه می کار داره عالی هم خیلی .هست من کنترلر و صفحه اولین این
میخوام من مثلاً .کنن می کار راحت خیلی اینجا هم ال ام تی اچ مدهای <b> بشه بولد کلمه این</b>";
        }

        //
        // GET : /Farjad/Welcome
        public string Welcome()
        {
            return "آدرس در و شده ایجاد کنترلر توسط استاتیک صورت به که دیگه ی صفحه یک اینم"
localhost/Farjad/Welcome قابل دسترس .هست </br> یعنی کنید انتخاب تابع برای رو اسمی هر کنترلر قسمت در شما
آدرس یه میشه!";
        }
    }
}
```

قبل از اینکه بخوام این کدهارو توضیح بدم اول پروژه رو ذخیره کنید و بعد ران کنید. توسط کنترل های CTRL + F5 میتونید پروژه را دیباگ و ران کنید.

خب اگر آخر آدرس لوکال هاست Farjad رو بزنم وارد همین صفحه ای میشم که توی عکس بالایی دارید میبینید و اگر در آخر آدرس لوکال هاست Farjad/Welcome رو تایپ کنم وارد صفحه ای دیگه میشم مانند عکس زیر

حالا قضیه چیه؟

ما توی کنترلی که ایجاد کردیم یک اکشن (MVC به متدهای Public درون یک کنترلر Action میگه. نمیدونم چه معادل فارسی براش انتخاب کنم) درست کردیم به است Farjad که از نوع String یا همون رشته هست. این اکشن میفهمه که قراره یه نوشته ای رو به کاربر نشون بده. ولی این رشته یا همون نوشته توی کجا قراره به نمایش در بیاد! MVC بهتون میگه این همون آدرسیه که شما دارید تایپ می کنید. یعنی یه صفحه براتون ایجاد می کنه به است Farjad یا Welcome که اگر آخر آدرس تایپ بشه ، بهتون اون رشته رو نمایش میده! شاید کمی سخت توضیح دادم! آدرس URL شما همون چیزیه که دارید اکشن ایجاد می کنید. مثلاً الان میخوام یه صفحه ی دیگه داشته باشم و توش یه سری چیزهای دیگه نوشته شده! مثلاً همین پارارگراف. براش یه اسم هم انتخاب می کنم. Par1 چطوره؟ کافیه یه اکشن دیگه ایجاد کنم تحت همین نام و یه رشته توش بنویسم. مثل کد زیر و عکس زیر

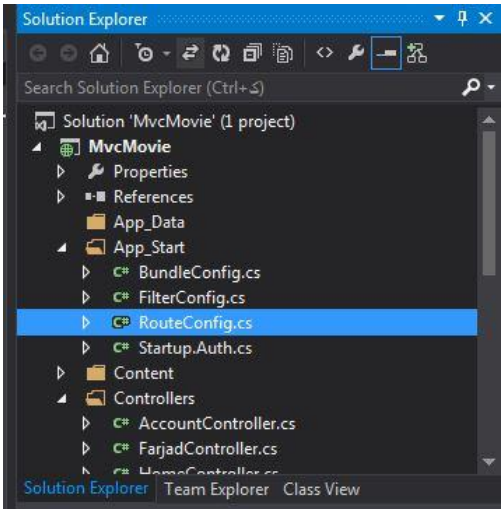
این کدش:

```
public string Par1()
        {
            return "یا String نوع از که Farjad است به کردیم درست اکشن یک کردیم ایجاد که کنترلی توی ما
قراره کجا توی نوشته همون یا رشته این ولی .بده نشون کاربر به رو ای نوشته یه قراره که میفهمه اکشن این .هست رشته همون
است به کنه می ایجاد براتون صفحه یه یعنی .کنید می تایپ دارید شما که آدرسیه همون این میگه بهتون MVC !بیاد در نمایش به
آدرس !دادم توضیح سخت کمی شاید !میده نمایش رو رشته اون بهتون ، بشه تایپ آدرس آخر اگر که Welcome یا Farjad
URL شما همون چیزیه که دارید ایجاد می .کنید ایجاد تابع دارید که میخوام الان مثلاً .کنید می صفحه یه میخوام توش و باشم داشته دیگه ی صفحه یه
پارارگراف همین مثلاً !شده نوشته دیگه. ";
        }
```

اینم عکسش :

اگر شما آدرس رو بدون Farjad وارد کنید به شما ارور میده. و حتما باید آخر لوکال هاست و اسم کنترل این اکشن رو فراخوانی کنید. پس نحوه ی آدرس دهی به این صورته : Local/ControlName/ActrionName

نحوه ی آدرس دهی در MVC5 بسیار پیچیده تر و بهبود یافته تر شده. به نظر شخصیم MVC5 در این نسخه همه چیز رو بهبود داده به خصوص نحوه ی آدرس دهی که معروف به MapRoute هست. حالا ما چطور میتونیم این مپ روت رو تغییر بدیم و به اصطلاح مطابق با سلیقمون پیاده سازی کنیم. شاید از خودتون بپرسید چرا باید این کار رو انجام بدید!؟ توی پروژه های کوچیکی مثل همین پروژه ای که داریم کار می کنیم تغییر آدرس دهی زیاد مهم نیست. اما وقتی بحث یه پروژه ی بسیار بزرگ در میونه شما اطلاً نمیتونید از آدرس دهی استاندارد و پیش فرض استفاده کنید. سعی می کنم در طول کتاب این مبحث رو کامل توضیح بدم. شما برای تغییر این نحوه ی آدرس دهی میتونید از طریق فولدر APP_Start فایل و کلاس RouteConfig.cs رو انتخاب کنید.



همونطور که میبینید مایکروسافت برای آدرس دهی در این نسخه از MVC یک فایل مجزا در نظر گرفته! و شاید براتون جالب باشه بدونید که یه کتاب هم براش منتشر کرده که چطور میتونید آدرس دهی کنید! شما در فایل مربوطه با کدهای زیر مواجه میشید

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace MvcMovie
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
```

```
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
        }
    }
}
```

به صورت خیلی خلاصه در صفحه ی قبل توضیح دادم. اینجا کمی مفصل تر توضیح میدم. در خط ۱۸ ما شاهد نوعی آدرس دهی هستیم. "{controller}/{action}/{id}" خب همونطور که معلومه Controller اول به ما نام کنترلی را که میخواهیم فراخوانی کنیم می گوید. ما در ابتدای قدم اول کنترلی ایجاد کردیم تحت عنوان Farjad. حال ممکن است در طول پروژه کنترل های زیادی ایجاد کنیم که آدرس دهی از طریق این نام ها اتفاق می افتد. بعد از نام کنترلر نام Action مورد نظر را فراخوانی می کنیم. یعنی می گوییم برو در داخل کنترلر X و اکشنی که نام Y را دارد برای ما فراخوانی کن. ما در این کنترلر Farjad سه اکشن را ایجاد کردیم. و در نهایت ID که این آیدی در صورتی که اکشن شما دارای آی دی های مختلفی باشد یکی که مد نظر شما است را فراخوانی می کند. به عنوان مثال شما یک فروشگاه دارید و هر محصول یه آی دی یونیک دارد. برای اینکه کارکرد این روتینگ را متوجه شوید کمی تغییر در این کدها می کنیم. بعد از کنترل یک عنوانی را وارد می نمایید. مثلاً من test نوشتم.

```
    public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/test/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
```

حال اگر دوباره پروژه را دیباگ و اجرا کنم با ارور مواجه می شوم. مانند شکل زیر

اما اگر مانند آنچه در روتینگ نوشتم کلمه ی test را اضافه کنم مشکل حل میشود یعنی آدرس را باید مانند زیر تغییر دهم : http://localhost:4568/Farjad/test/Par1 تا شاهد اجرای پروژه به درستی ومانند شکل زیر شوم

فعلاً اون کلمه ی test رو از فایل کانفیگ روت پاک می کنیم.

ما در مثال بعدی خواهیم دید چطور این موارد را میشود به مرورگر انتقال داد.

برای واضح تر شدن مسایلی که در بالا و مبحث روتینگ مطرح شد و برای اینکه با کارکرد کنترل ها و اکشن ها بیشتر آشنا شوید یک کنترل دیگر با هر نامی که دلخواهتان است ایجاد کنید. من در اینجا نام Morteza را انتخاب و ایجاد می کنم و در کنترلر جدید اکشنی تحت عنوان testparam را تایپ می کنم. مانند کد های زیر

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MvcMovie.Controllers
{
    public class MortezaController : Controller
    {
        //
        // GET: /Morteza/
        public string testparam(string name, int numT=1)
        {
            return "تست مطلب";
        }
    }
}
```

برای اینکه مطمئن بشیم پروژه به درستی کار می کنه یکبار دیگه دیباگ و ران می کنیم و آدرس همین کنترلر رودر مرورگر وارد می کنیم : http://localhost:4568/Morteza/testparam باید شبیه عکس زیر بشه

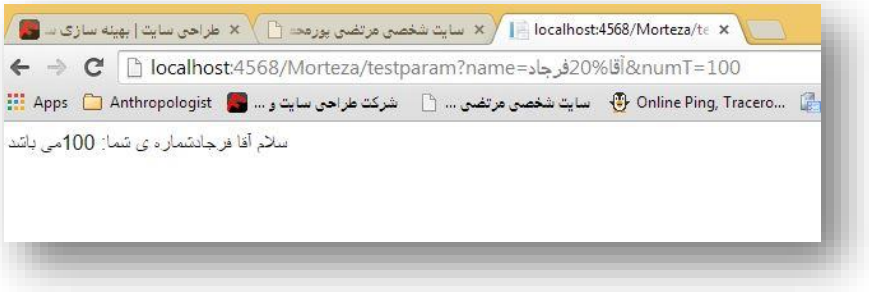
می خواهم توسط این کدها پارامترهایی را به مرورگر انتقال و نمایش دهم. اما این پارامترها کاملاً انعطاف پذیر هستند و دیگر من در کدها چیزی نمینویسم. ابتدا به کدها توجه کنید :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MvcMovie.Controllers
{
    public class MortezaController : Controller
    {
        //
        // GET: /Morteza/
        public string testparam(string name, int numT= 23 )
        {
            return HttpUtility.HtmlEncode("سلام " + name +  "شما  ی شماره:  " + numT + "می
باشد");
        }
    }
}
```

قبل از اینکه توضیح بدم اول پروژه رو دیباگ و ران می کنیم تا شاهد عکس زیر بشیم

همونطور که متوجه شدید ما دو پارامتر و تابع ایجاد کردیم که یکیش عدد بود و دیگری اسم بود! ولی در موقع اجرای پروژه فقط شماره منتقل شد و نمایش داد. حالا بیاید آخر آدرس کمی تغییر بدیم من آخر آدرس اسم خودم رو خیلی محترمانه تایپ و صفحه رو رفرش می کنم. مطابق عکس اسمم رو به صفحه اضافه می کنه

و اگر بعد از این آدرس &numT=100 رو وارد کنم عدد نیز تغییر میکنه. مثل شکل زیر



ما اومدیم و داخل اکشن testparam دو پارامتر ایجاد کردیم. یکی از نوع string بود و دیگری int برای ذخیره سازی اعداد. در داخل این اکشن نیز از توابع HTMLکه در MVC تعریف شده اند کمک گرفتیم. توابع html یکی از معروف ترین و پرکاربردترین توابع MVC می باشند.

خب تا اینجا که راحت بود. حالا وقتشه کمی به مبحث شیرین و پرکاربرد MapRoute برگردیم. همین مثال آخری که زدیم رو مد نظر قرار بدید. اینگونه آدرس دادن کمی ناشایسته! اصولاً ما ایرانیا جدیداً خیلی درگیر سئو شدیم! و برامون خیلی مهمه که آدرس ها بسیار خوانا باشن! اینجوری آدرس دادن مطمئناً همتون میدونید که اصلاً خوب نیست. پس باید چیکار کنیم؟! روتینگ به ما کمک میکنه! ما میخواهیم جوری مپ روت (MapRoute) کنیم که به بقیه ی آدرس ها و کنترل ها ضربه وارد نشه. مشکلی نداره. مپ روت در نسخه ی جدید ام وی سی به شما این امکان رو میده هرجایی از سایت رو که خواستید به سلیقه ی خودتون تغییر آدرس دهی کنید به طوریکه بقیه ی آدرس ها و ... تغییر نکنن. دوباره فایل RouteConfig.cs رو باز کنید و کدهای زیر رو که اضافه شده کپی/پیست کنید.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace MvcMovie
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
```

```
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
             routes.MapRoute(
                name: "Morteza",
                url: "{controller}/{action}/{name}/{numT}"
                );
        }

    }
}
```

خب ما اینجا خیلی راحت یک روت ایجاد کردیم و اسمشو دقیقاً اسمی که برای کنترل گذاشتیم انتخاب کردیم و در قسمت url اونجوری که مد نظرمون بود آدرس دهی کردیم. یعنی این دو خط :

```
name: "Morteza",
url: "{controller}/{action}/{name}/{numT}"
```

یعنی قسمت name باید نام کنترلی که قراره MapRoute –ه خاصی بگیره ، وارد می کنیم. در قسمت url نیز اول میگیم اسم کنترل رو بخون. بعد اسم اکشن، بعد نام که همون string بود و در آخر هم اون عدد رو بخون که اینم اگر یادتون باشه int بود. حالا ذخیره می کنیم و دیباگ ران می کنیم و آدرس دهی جدید رو وارد می کنیم. به آدرس دهی در عکس زیر توجه کنید. شما هر اسم و عددی که بخواهید به جای اون دو مورد وارد کنید:

ما در بخش های بعدی که قراره دیتابیس ایجاد کنیم و به قول معروف سایت کاملاً داینامیک بشه به این مدل آدرس دهی نیاز پیدا می کنیم. بر فرض شما می تونید بگید اول آدرس دسته ی محصول رو بخون بعد آی دی محصول. یا حتی میتونید دقیقترش هم کنید. مثلاً اول اسم شرکت تولید کننده بعد مجموعه و بعد اسم محصول و در نهایت آی دی محصول و انواع مختلف آدرس دهی. حتی میتونید ساده ترش هم کنید. یعنی اول آدرس کنترل و بدون هیچ چیزی آی دی محصول!

خب این از قدم اول بود. فکر کنم تا اینجای کار مشکلی وجود نداشته باشه. در قدم دوم سراغ View ها میریم و یاد میگیریم چجوری صفحات رو داینامیک کنیم.

☑ استفاده از Views

❖ قدم دوم : ایجاد اولین View

در این قدم ما یاد میگیریم چطور با ویوها کار کنیم و پاسخی به درخواست کاربر درقالب یک صفحه ی HTML ارسال کنیم. یاد میگیریم چگونه توسط موتور Razor و View Engin یک قالب برای نمایش ایجاد کنیم اما قبلش باید کمی با رازور و قوانین آشنا بشیم

❖ شاید از خودتون بپرسید Razor چیه؟

Razorیا رازور(ریزور) یک انجین تبدلی کدهای سرور ساید یا سمت سرویس دهنده Asp.net به اچ تی ام است که از نسخه MVC 3 به بعد از طرف ماکروسافت ارائه شده و نسبت به صفحات قدیم asp.net که عموما با پسوند aspxشناخته می شوند از cshtml استفاده می کنه.

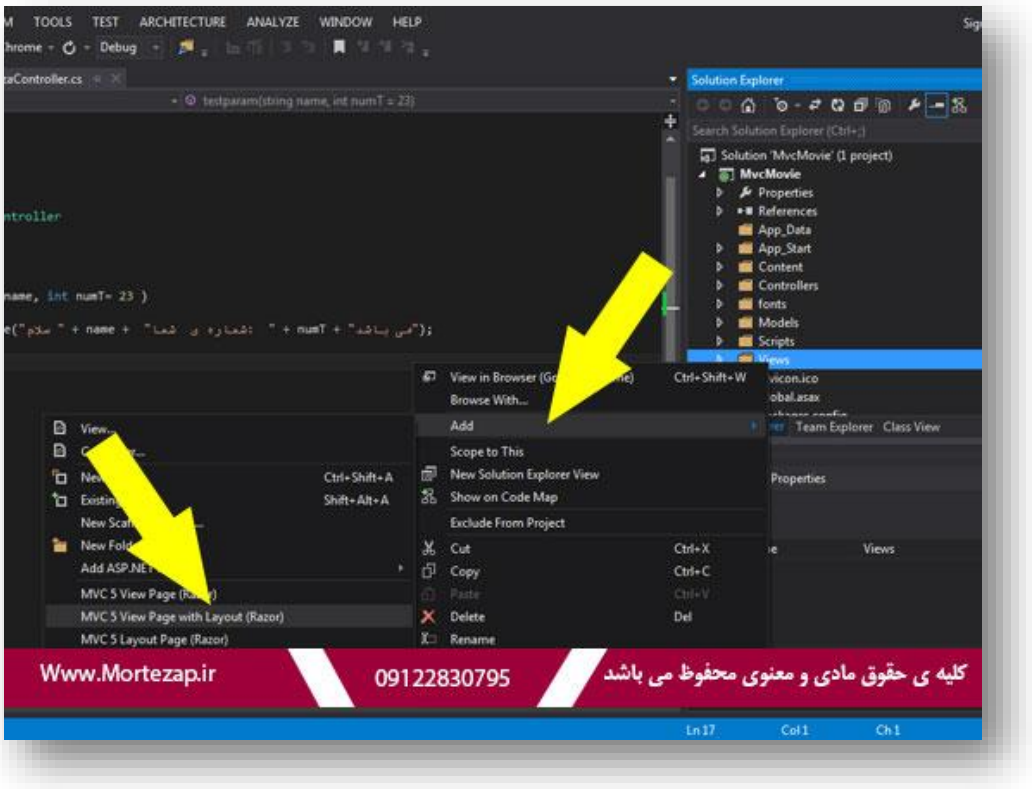
در این مدل صفحات که برای برنامه نویسان Asp.net MVC توصیه شده اما برنامه نویسان Asp.net Web Form هم می توانند از آن استفاده کنند، کار با Razor بسیار راحت تر شده و شما می تونید کد کمتر تایپ و بهره بیشتری از مطالب خود ببرید و در عین سادگی بسیار حرفه ای تر کد نویسی کنید ، سرعت بارگذاری یا رندر شدن این صفحات از صفحات قدیمی aspx بالاتر است و دیگر از کنترل های <asp:> و view state خبری نیست و همین امر در مصرف حافظه و ... تاثیر گذار است و Performance سایت را بالاتر می برد.

- Razorگرامری است برای افزودن کد های سمت سرور به.Webpage
- Razorقدرت نشانه گذاری ASP.NET سنتی را دارد، با این تفاوت که Razor را راحت تر می آموزیم و راحت تر به کار می بریم.
- Razorگرامر نشانه گذاری سمت سرور است که بیشتر مشابه PHP و ASP می باشد.
- Razor زبان های برنامه نویسی Visual Basic و C# را پشتیبانی می کند.

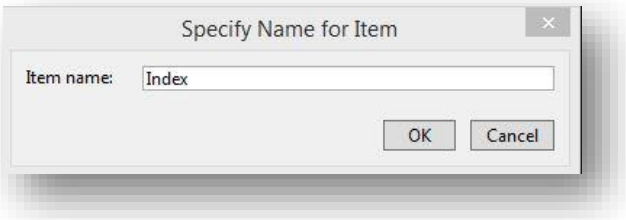
حالا این رازور بای خودش یه سری قوانین (برای زبان C#) هم داره که باید باهاشون آشنا بشیم و به کار بگیریمشون:

- بلوک کد Razor در @{....} محصور گردیده است.
- عبارت های Inline (توابع و متغیرها) با @ شروع می شوند.
- خط کدها با نقطه ویرگول خاتمه می یابند.
- متغیرها با استفاده از کلمه کلیدی var تعریف می شوند.
- رشته ها با علامت " محصور می شوند.
- کدهای C# حساس به حروف بزرگ و کوچک هستند.
- فایل های C# دارای پسوند cshtml. هستند.

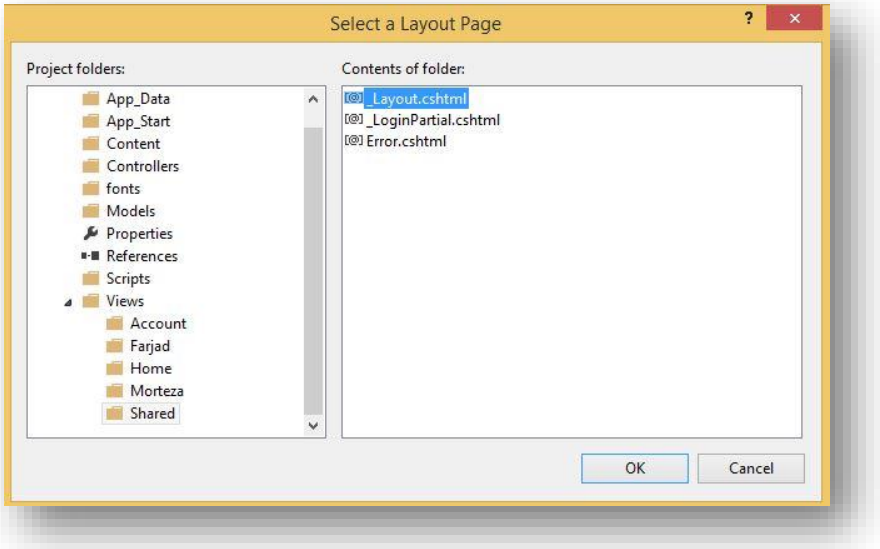
خب حالامیخواهیم اولین View رو ایجاد کنیم. راحت ترین راه اینه که روی پوشه ی View راست کلیک کرده و سپس از منوی Add گزینه ی MVC 5 View Page with (Layout Razor) رو انتخاب کنید. به طور کلی تر میتونیم به موارد زیر در خصوص Razor اشاره کنیم



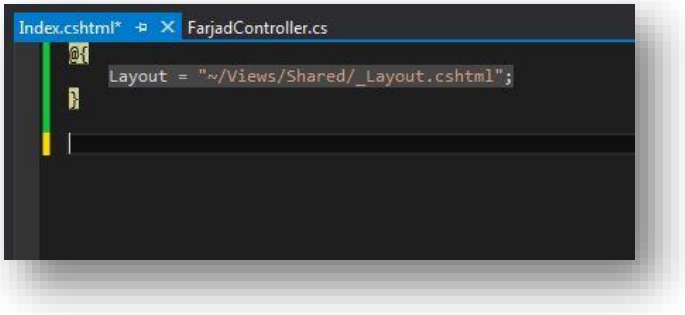
در کادر بعدی یک نام بای View انتخاب می کنیم



و در پنجره ی بعدی لایه ای که میخواهیم از آن ارث بری کند را انتخاب می کنیم. به زبان عامیانه تر باید گفت صفحه که میخواهد شبیه آن شود را انتخاب می کنیم. یه جورایی ضرب المثل معروف "پسر کو ندارد نشان از پدر"



خب میبینیم که اولین View رو ایجاد کردید. یه چیزی شبیه شکل زیر و با نام Index.cshtml



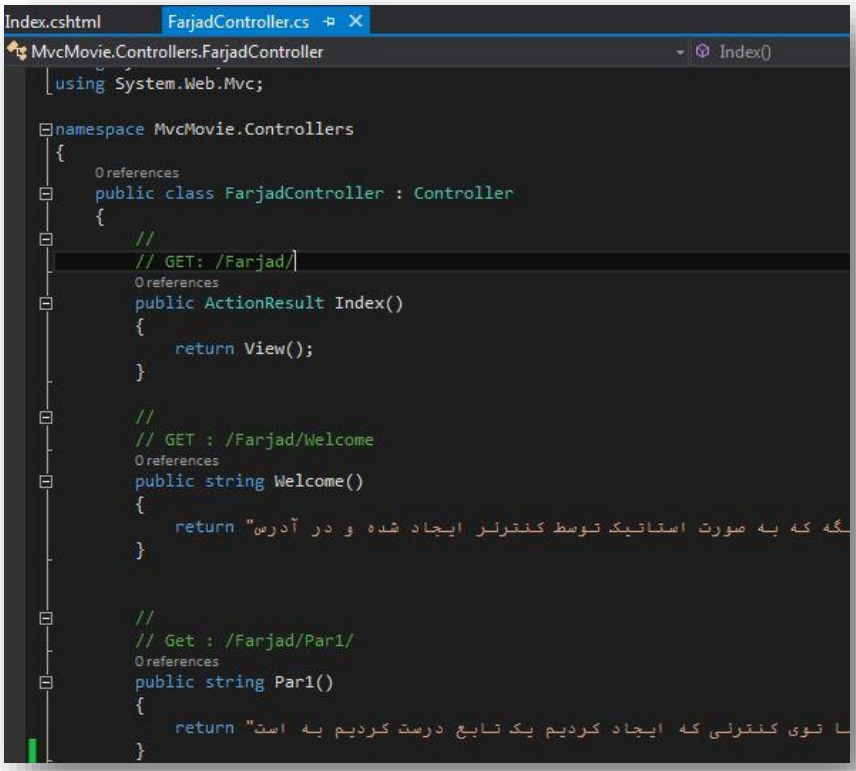
اما ما میخواهیم بر روی یکی از کنترلرها کار کنیم!!! این مراحل بالا را انجام دادیم تا شما با نحوه و روند انجام آشنا شوید و متوجه شوید که مهم نیست اول کنترلر را ایجاد می کنید یا اول View. کمی نوشته به این ویو اضافه می کنیم

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}

@{
    ViewBag.Title = "Index";
}

<h2>دیگه ی نمونه متن یه اینم</h2>

<p>کنید استفادهن اس ام تی اچ کدهای از بخواد دلتون تا تونید می شما .پاراگراف برای متن هم این و</p>
```

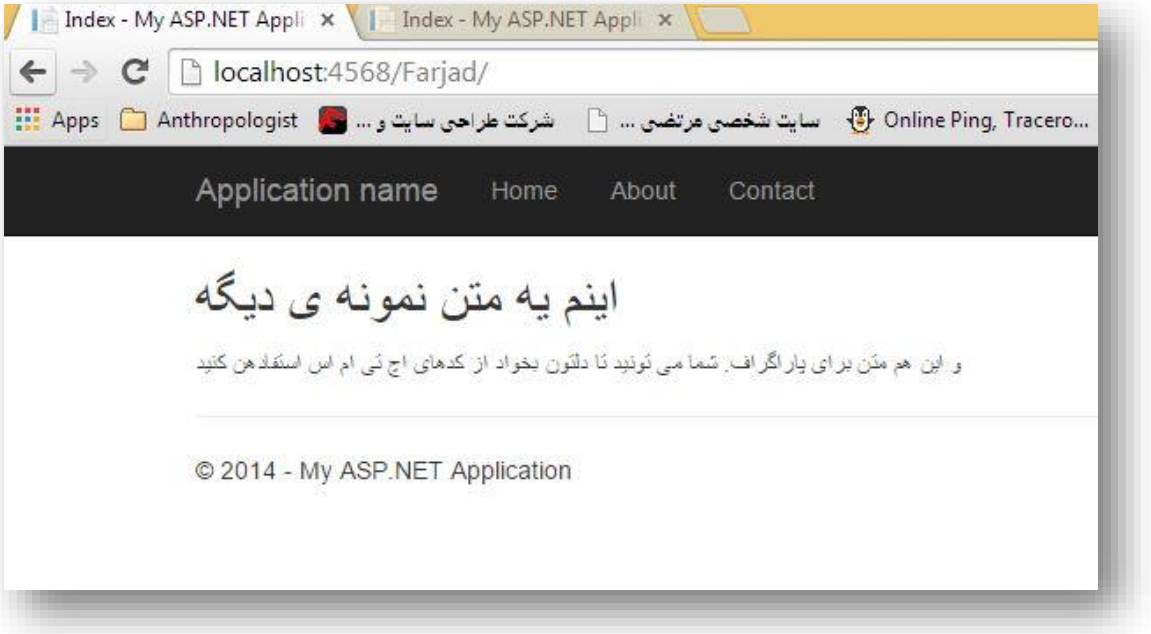
اما باید اینجا یک نکته ی بسیار مهم رو اشاره کنیم بهش و اونم اینه که اگر پروژه رو ران کنید بهتون ارور میده. باید برگردید توی کنترلر و کمی تغییرات بدید. این چند خط رو در فایل کنترلر تغییر میدیم و یا اضافه می کنیم

```
//
// GET: /Farjad/
public ActionResult Index()
{
    return View();
}
```

یعنی باید شبیه عکس زیر باشه



حالا اگر پروژه رو ران دیباگ کنید و آدرس http://localhost:4568/Farjad/ رو وارد کنید با عکس زیر مواجه میشید:



همین مراحل بالا رو برای بقیه ی اکشن ها در کنترلر میتونید انجام بدید و برای هرکدومشون که خواستید یک ویو درست کنید. یک نمونه دیگه میزنیم تا روشنتر بشه. در کنترلر FarjadController.cs یک اکشن با نام FarjadV می نویسیم مثل کدهای زیر

```
        //
        // Get : /Farjad/FarjadV/
        public ActionResult FarjadV()
        {
            return View();
        }
```

بعد یک ویو طبق مراحل بالا براش ایجاد می کنیم و کدهای زیر رو داخلش می نویسیم

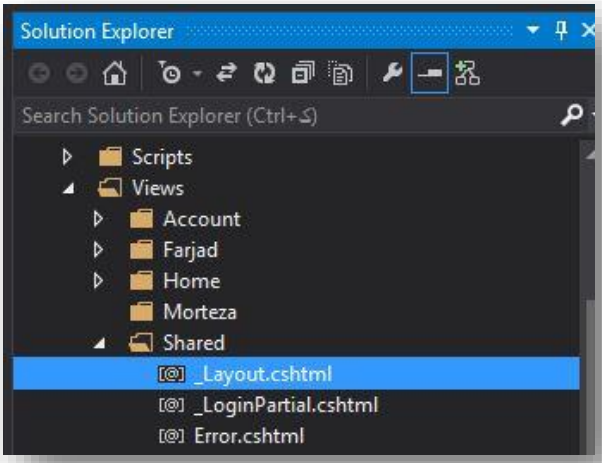
```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}

@{
    ViewBag.Title = "Index";
}

<h2>ویو از دیگر مثالی در جدید اکشن برای نمونه متن</h2>

<p>کنید استفادهن اس ام تی اچ کدهای از بخواد دلتون تا تونید می شما .پاراگراف برای متن هم این و</p>
<p><b>میشه بولد که دیگه پاراگراف یک یا و</b></p>
```

یعنی چیزی شبیه شکل زیر:

در کد بالا مفهوم @ که قرار دادیم ، آسان ترین راهبرای معرفی یک فضای نام (namespace) به این ویو حاضر میباشد. و کد View bag هم دارای تعدادی متد است که پرکاربردترین آنها Title می باشد. این تگ همان کار تگ <title>TEST Title</title>در HTML را انجام می دهد. کافیست پارامتر را عوض کنید و صفحه را رفرش نمایید تا بیشتر آشنا شوید.

❖ تغییرات ظاهری View

همیشه ظاهر برای یک سایت مهمه اما ما در این بخش قرار نیست که یه طراحی خیلی حرفه ای انجام بدیم. فقط قراره با هم دیگه ،در چیزی که مشاهده می کنیم کمی تغییرات اعمال کنیم. برای شروع کافیه فایل _Layout.cshtml رو از مسیر Views > Shared باز کنید



همانطور که ملاحظه می کنید فایلی شبیه به عکس زیر شامل کدهای HTML و کمی هم کدهای MVCدر این فایل مشاهده می کنید

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Application name", "Index", "Home", null, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>
                @Html.Partial("_LoginPartial")
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
```

دوستانی که با HTML آشنایی دارند به راحتی می توانند تغییراتی در ظاهر این فایل انجام دهند.

در MVC5 یک متد با نام HTML وجود دارد که دارای پارامترهای بسیار متعددی است. از جمله پارامتر هایپرلینک که به شکل ActionLink بعد از HTML قرار می گیرد و به صورت @Html.ActionLink نمایش داده می شود. حال کار این تگ چیست؟! خیلی راحت و واضح است. هرچیزی را که بخواهید لینک می دهد. برای واضح تر شدن مطلب کمی تغییرات در این صفحه میدهیم. چند خط زیر را تغییر میدهیم

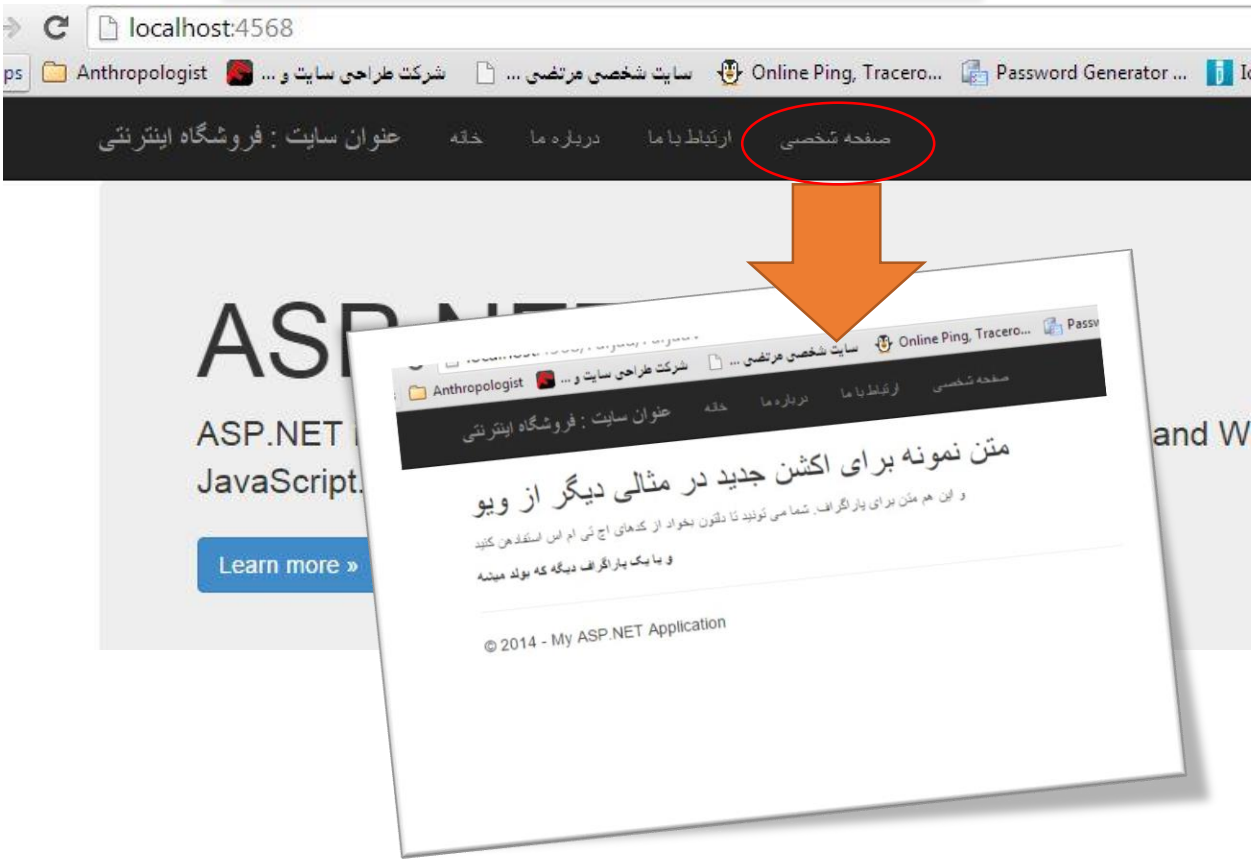
```
    @Html.ActionLink("اینترنتی فروشگاه : سایت عنوان", "Index", "Home", null, new { @class =
"navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("خانه", "Index", "Home")</li>
                    <li>@Html.ActionLink("ما درباره", "About", "Home")</li>
                    <li>@Html.ActionLink("ما با ارتباط", "Contact", "Home")</li>
                </ul>
```

پروژه را ران دیباگ کرده و با صفحه ای مانند شکل زیر مواجه می شویم

همانطور که ملاحظه می کنید تمامی نوشته های لینک ها عوض شد. پس Html.ActionLink@ دارای یکسری پارامتر است. اولین پارامتر نوشته ی معمولی است که ما در اینجا نوشتیم "درباره ما". دومین کوتیشن مربوط به نام اکشن ایست که فراخوانی می کنیم که در اینجا اکشن "About" فراخوانی شده است و سومین کوتیشن مربوط به ویویی است که این لینک باید فراخوانی کند. برای روشن تر شدن موضوع یک لینک دیگر اضافه می کنیم. می خواهیم یکی از اکشن هایی که در صفحات بالا ویو برایش ایجاد کردیم را لینک کنیم. کدهایش شبیه زیر می شود

```
<li>@Html.ActionLink("شخصی صفحه", "FarjadV", "Farjad")</li>
```

یعنی اول متنی که قرار است به کاربر نمایش دهیم را می نویسیم. سپس نام اکشنی که در کنترلر FarjadController.cs وجود دارد را می نویسیم و در نهایت ویویی که برایش ایجاد کردیم در فولدری با عنوان Farjad را تایپ می نماییم. حال پروژه را ران دیباگ می نماییم . بعد از کلیک کردن بر روی لینک جدید متوجه می شویم که وارد آدرس http://localhost:4568/Farjad/FarjadV می شود. زیرا ما در کوتیشن دوم نام این اکشن را وارد نمودیم.



مطمئناً متوجه شدید اگر کوتیشن دوم را به Index تغییر دهم لینک و عکس بالا نیز تغییر می کند یعنی اگر کد زیر را بنویسم

```
<ul class="nav navbar-nav">
    <li>@Html.ActionLink("خانه", "Index", "Home")</li>
    <li>@Html.ActionLink("ما درباره", "About", "Home")</li>
    <li>@Html.ActionLink("ما با ارتباط", "Contact", "Home")</li>
    <li>@Html.ActionLink("شخصی صفحه", "Index", "Farjad")</li>
</ul>
```

لینک وارد صفحه ی دیگر و آدرس http://localhost:4568/Farjad می شود.

❖ ارسال اطلاعات از کنترلر به ویو

تا کنون ما چیزهای زیادی در خصوص کنترلر و ویو یاد گرفتیم. اما اینها فقط موارد ابتدایی بود. قصد راه اندازی ما از سایت ، داینامیک بودن آن است. اینکه چطور اطلاعات را دیگر تایپ نکنیم! در این بخش ما با نحوه ی ارسال اطلاعات از اکشن کنترلر به ویو آشنا می شویم. قبل از اینکه بخواهیم برویم سراغ ایجاد دیتابیس و یا درخصوص Model ها صحبت کنیم بهتره اول در خصوص نحوه ی ارسال اطلاعات از کنترلر به ویو کمی نظراندازی کنیم مثل حافظ :دی

تا قبل از این شمارو با نحوه ی کارکرد کنترلرها آشنا کردیم. حال میخواهم مطالبمان را کاملتر بیان کنیم. کلاس های کنترلر با توجه به URL های ورودی یا درخواستی استناد می کنند و پاسخی را ارائه می دهند. یعنی میبینند در

مرورگر چه چیزی درخواست شده متناسب با درخواست کلاسی را برمیگردانند . البته منظورم کارهایی است که آن کلاس انجام میدهد. اگر یادتان نمیاید برگردید و مثالی که در مرورگر اسم را تایپ می کردیم و این اسم را در صفحه مینوشت ، نگاهی بیندازید. کلاس های کنترلر جایی هستند که شما کدهایی مینویسید تا به درخواست مرورگر پاسخ دهید، پاسخی بر مبنای درخواست، از جمله بازیابی اطلاعات و داده ها از دیتابیس، و اینجاست که کنترلر تصمیم می گرد در سوال یا درخواست مرورگر چه نوع اطلاعاتی را نمایش دهد. در نهایت این پاسخ را در قالب View ها برای کاربر به نمایش می گذارد و شاید لازم باشد دوباره ذکر کنم View ها پاسخ کنترلر را در قالب HTML به کاربر نمایش می دهند. کنترلر ها مسئولیت تامین و مهیا کردن دیتا ها یا همان اطلاعات ها و آبجکت هایی که کاربر درخواست داده است و برای View اولویت دارد را بر دوش می کشد و ویو هم پاسخ را رندر و آماده ی نمایش برای کاربر می نماید. پس این نکته رو هم باید اضافه کنیم که View ها مکانی برای لایه ی منطقی و یا فعالیت های دیتابیس **نیستند**. شرمنده این چند خط شبیه هم بود. خواستم توضیح کافی داده باشم.

خب الان میخواهیم مواردی را که در بالا گفتیم تمرین کنیم. یعنی

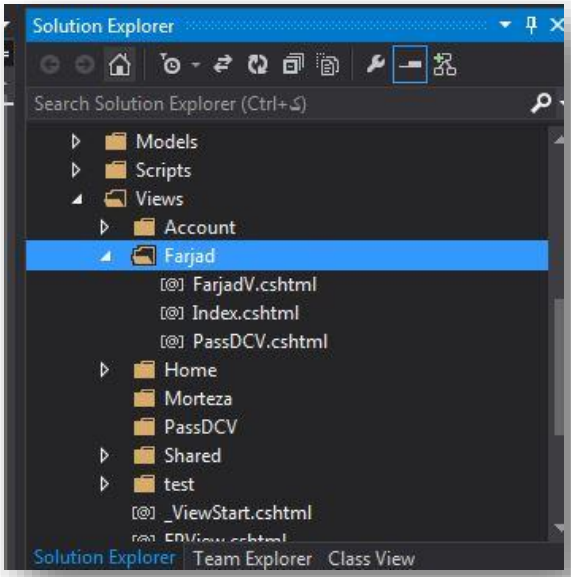
۱. یک کنترلر ایجاد می کنیم و یا از کنترلر های قبلی استفاده کنید
۲. یک ویو برای کنترلر ایجاد می کنیم
۳. در آدرس مرورگر متغیری را وارد می کنیم تا نمایش دهد

پس اول باید کنترلر ایجاد کنیم. در فصل قبل توضیح دادیم. من در کنترلر Farjad یک اکشن ایجاد کردم با نام PassDCV و کدهای زیر را نوشتم:

```
        //
        // Get : /Farjad/PassDCV/
        public ActionResult PassDCV(string name, int numtimes = 1)
        {
            ViewBag.Message = "نمایش برای ای نوشته یا و نام انتقال" + name;
            ViewBag.NumTimes = numtimes;

        }
```

دو متغیر ایجاد کردم یکی برای اسم و دیگری برای تعداد نمایش آن اسم در صفحه. توسط ViewBag این متغیرها را به ویو ارجاع می کنم. بعد از ویوبگ نامی نوشتم. این نام نیز دلخواهی است و هرچیزی می توانید بنویسید. به نوعی ID محسوب می شود. چون در View این نام ها را باید استفاده کنیم.

حالا میریم سراغ ویو. داخل فولدر Farjad در فولدر View یک ویو با نام PassDCV ایجاد می کنم. مانند شکل زیر



حال باید داخل این ویو چیزی را که قرار است نمایش دهیم تایپ می کنیم. در اصل باید مواردی را از کنترلر به نوعی فراخوانی کنیم. کدهای زیر را می نویسیم

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}

@{
    ViewBag.Title="دیتابیس بدون ساده اطلاعات ارسال صفحه نمونه";
}

<h2>کنید استفاده ال ام تی اچ های تگ از توانید می نیز اینجا در</h2>

<ul>
    @for (int i = 0; i < ViewBag.NumTimes; i++)
    {
        <li>@ViewBag.Message</li>
    }
</ul>
```

اومدیم توسط ul درخواست دادیم که بالت بنداز. بعد یک متغیر i تعریف کردیم که به ازای تعداد یکی اضافه کند. و در بخش li نام ها را برگرداند.

حال پروژه را ران دیباگ می کنیم و وارد آدرس : http://localhost:4568/Farjad/PassDCV میشویم

همانطور که ملاحظه می کنید فقط یکبار متن را برایمان به نمایش در آورد . همان متنی که در کنترلر نوشته بودیم.
اما کمی تغییر می دهیم. یعنی URL را تغییر می دهیم

http://localhost:4568/Farjad/PassDCV?name=MVC5Education&numtimes=4
شاهد صفحه ی زیر خواهیم بود

همانطور که ملاحظه می کنید متنی را که در کنترلر بود به همراه متنی که در مرورگر وارد کردیم به تعداد ٤ مرتبه نمایش درآورد

❖ ViewBag چیست؟

ViewBag یک شیءِ dynamic است که در دات نت ٤ به بعد امکان تعریف آن به قابلیت های این فریمورک اضافه شده است. به این معنا که هر نوع خاصیت دلخواهی را می‌توان به این شیءِ انتساب داد و سپس این اطلاعات در View نیز قابل دسترسی و استخراج خواهد بود. در مثال های بالا ما نیز همین کار را کردیم. نوعی انتقال اطلاعات به وسیله ی این شی انجام دادیم. یعنی ابتدا این ویوبگ را در کنترلر توسط یک آی دی می شناسانیم! و بعد در ویو

همان را عیناً تایپ می کنیم و با اطلاعاتی که منتقل می شود هر کاری می توانیم انجام دهیم. به زبان راحت تر نوعی حمل کننده ی مبدا و مقصد است! و این حمل کننده در مبدا با هر نامی که حرکت می کند در مقصد نیز همان نام را به همراه محتویاتش دارد.

در بخش بعدی به توضیح مدل می پردازیم و اینکه چگونه یک دیتابیس ایجاد نماییم و اطلاعات را کاملاً داینامیک کنیم.

☑ کار با Model

❖ Model چیست؟

در قسمت قبلی متوجه شدیم چگونه اطلاعات بین صفحات منتقل می شوند . اما در آنجا ما با دیتابیس سرو کار نداشتیم. حال در این بخش قرار است ما دیتابیس ایجاد کنیم تا پروژه ی مان تقریباً شبیه سایت شود! یعنی سایتی که به صورت کاملاً داینامیک اطلاعات را ثبت و نمایش می دهد.

Model ها کلاسهایی هستند برای کمک به پاراگراف بالا. کلاسهایی که می توان از آنها با عنوان لایه ی تجاری در طراحی سه لایه ی ASP.Net اشاره کرد. این بخش و کلاً Model ها مهمترین کارکرد را در MVC دارند و از دید شخصی به عنوان مغز برنامه هستند. اما این کلاسها چگونه کار می کنند؟ حتماً تا به حال واژه ی Entity Framework به گوشتان خورده است. و مطمئناً با این مبحث آشنا هستید. اما برای اون دسته از عزیزانی که با EF آشنایی ندارند باید کمی توضیح دهم. EF فریمورکی بر پایه ی دات نت است که کار دسترسی به دیتا را انجام میدهد. البته توضیح من به دلیل اینکه قرار است در این کتاب خیلی ساده همه چیز را توضیح دهم بسیار بسیار بسیار خلاصه بود. پیشنهاد می کنم برای آشنایی بیشتر حتماً مقالاتی که هم به زبان فارسی و هم به زبان انگلیسی در اینترنت موجود می باشد را مطالعه کنید.

❖ Entity Framework چیست؟

فرقی نمی کنه شما با C#.NET کار می کنید یا VB.NET. برای کار با انواع مختلف دیتابیس از چه تکنولوژی استفاده می کنید؟ معمولاً با SQL Server دیگه! Entity Frameworkدرواقع یکی از قدرتمندترین ORM های دسترسی به منابع داده است. به وسیله Entity Framework شما می تونید با انواع مختلف دیتابیس از قبیل SQL Server ، SQLite و ... کار کنید بدون اینکه نیاز به عملیات مستقیم در دیتابیس داشته باشید. این تکنولوژی چند ویژگی مهم داره که توجه اکثر برنامه نویسان را به خودش جلب کرده است.

۱. کار کردن با Entity Framework بسیار ساده است. حتی یادگیری اون ده ها برابر ساده تر از مدل های مشابه مانند ADO.NET است.
۲. سرعت در دسترسی به داده ها و اجرای دستورالعمل هاش بسیار بالاست و در پروژه های بزرگ نیز امکان استفاده از این فریمورک وجود داره.
۳. این فریمورک به شما زمان هدیه میده!
٤. دیتابیس شما در قالب کلاس های شی گرا با سایر اجزای پروژه یکپارچه می شود و علاوه بر خوانایی بالا در کدنویسی، عملیات ارتقا و بروزرسانی نرم افزاری را سرعت می بخشد و خیلی از خطاهای رایج رو دیگه توی این فریمورک اصلاً بهشون فکر هم نمیکنید
۵. در برخی از حالت ها شما نیاز به طراحی پایگاه داده ندارید! زیرا با استفاده از امکانات EF خود برنامه دیتابیس را از روی مدل یا کلاس های شما می سازد. جلوتر به این مبحث بیشتر میرسیم

در نهایت اینکه Entity Framework تکنولوژی توسعه یافته ADO.Net است که فاصله بین برنامه نویسی شی گرایی و بانک اطلاعاتی رابطه ای را پر می کند. این فاصله معمولا تحت عنوان عدم تطابق شناخته می شود. و یک تکنیک برنامه نویسی برای تبدیل ارتباطات در Database به مفاهیم Object Oriented در برنامه نویسی است. در واقع می‌توان گفت که کلاس ها را به Table ها map می‌کنه. وقتی که شما می‌خواهید به Database دسترسی پیدا کنید، یا اطلاعاتی را ذخیره کنید، این کارها را مستقیما بر روی اشیاءِ (Object تان) انجام می‌دهید.

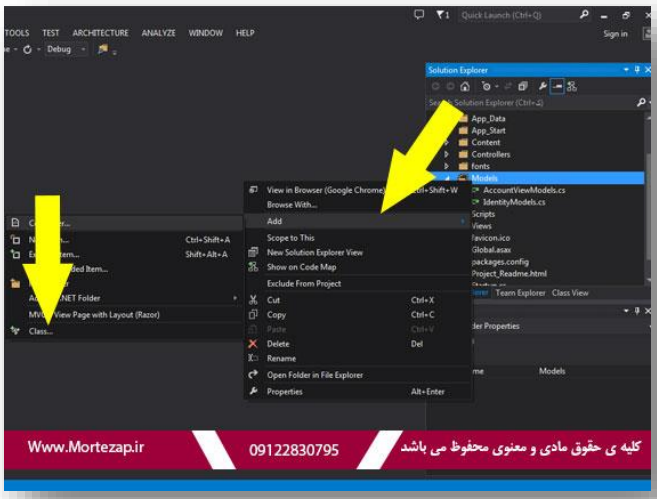
اما در EF سه نوع مدل سازی داریم : (این قسمت عیناً ترجمه از کتاب Pro EF چاپ انتشارات O'Relly هست)

۱. **روشDatabase First :** یعنی وظیفه توسعه دهنده نرم افزار این است که پایگاه داده را ایجاد کند. سپس با دادنای پایگاه داده بهEntity Framework ، یک فایل مدل با پسوند edmx تحویل بگیرد. در این فایل که یک فایلxml است، اطلاعات مربوط به موجودیتها، خصوصیات هر موجودیت و نوع داده هر خصوصیت قرار گرفته است. (به این قسمت از فایل مدل ، مدل مفهومی گفته می شود). در فایل مدل علاوه بر مدل مفهومی، مدل فیزیکی هم قرار دارد. مدل فیزیکی اطلاعات مربوط به جداول پایگاه داده و فیلدهای هر جدول و نیز نوع داده هر فیلد قرار گرفته است. فایل مدل شامل قسمت دیگری است که بیان می کند جداول پایگاه داده به چه موجودیتهایی نگاشت شده اند.
۲. **روش Model First :** در این روش وظیفه توسعه دهنده نرم افزار، ایجاد مدل می باشد. یعنی او خود فایلedmx را می سازد. بعد از این کار می تواند ازEF بخواهد تا پایگاه داده مربوط به این مدل را ایجاد کند. برای این کار او ازیک ابزار ویژوال به اسمEntity Data Model Designer استفاده می کند. این ابزار ویژوال بسیار قدرتمند می باشدو امکاناتی از قبیل ساختن موجودیتها، تعیین ارتباط بین موجودیتها، ایجاد کلید برای هر موجودیت، کاردینالیتی رابطه ها و غیره را در اختیار توسعه دهنده قرار می دهد. در روش های بالا ، درCode Behind مدل کلاسهایی مربوط به هر موجودیت و نیز شیContextوجود دارد.
۳. **روش Code First :** این روش خود شامل دو قسمت می باشد:
   I. اول؛ این است که توسعه دهنده کلاس های مربوط به هر موجودیت را ایجاد می کند. او این کار را با برنامه نویسی انجام می دهد. یعنی باید تک تک کلاسها، به همراه خصوصیات هر کلاس و البته خصوصیاتNavigation را بابرنامه نویسی ایجاد کند. خصوصیات Navigationآن دسته از خصوصیاتی هستند که ارتباط آن کلاس را با کلاس های دیگر مشخص می کنند. در این روش هیچ فایل مدلی وجود نخواهد داشت.
   II. دوم؛ این است که مثل روش Database First ابتدا توسعه دهنده یک پایگاه داده ایجاد می کند، سپس مدل را ایجاد می کند، سپس از یک افزونه می خواهد که موجودیتها را به همراهContext ایجاد کند و اینها را از مدل جدا کند. اگر موجودیتها به این طریق ایجاد شوند به آنها POCO گفته می شود. فرق آن با روش Database First در همین نکته است. در اینجا فایل مدل (فایلedmx) فقط اطلاعات مربوط به موجودیتها را نگهداری می کند و درCode Behind آن هیچ کدی نوشته نمی شود.

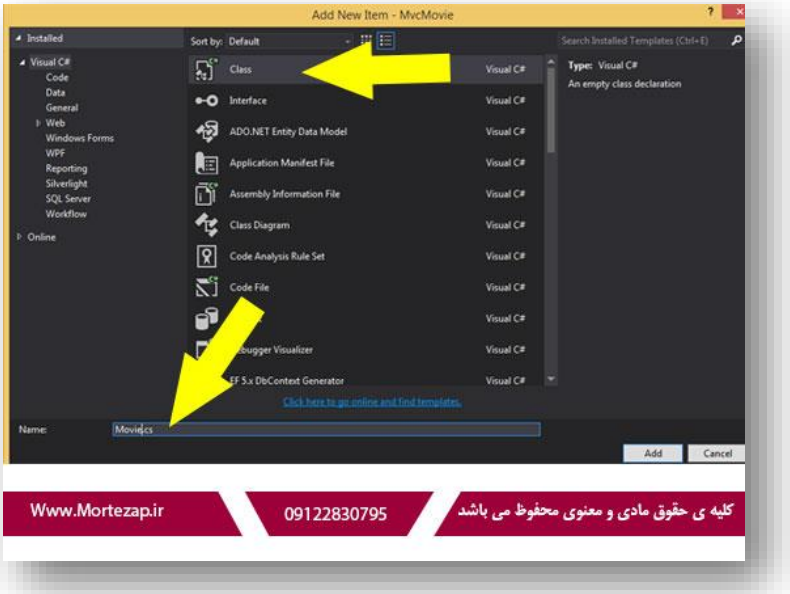
در کتاب بعدی مفصل راجع به این موضوع به همراه کلی مثال توضیح میدهم.

❖ قدم سوم : اضافه کردن کلاس Model

طبق معمول میریم سراغ ویژوال استودیو و در قسمت Solution Explorer بر روی فولدر Models راست کلیک کرده و از زیر منوی Add گزینه ی Class رو انتخاب می کنیم



در مرحله ی بعد یک نام برای کلاستان انتخاب می کنید. ما در اینجا نام Movie را برمی گزینیم



وقتی روی Add کلیک می کنید کلاستان با موفقیت ایجاد شده است. ما قرار است در این مرحله کلاسی بنویسیم که وظیفه اش ایجاد نام فیلم به همراه مشخصات آن است. چند فیلد ساده هم بیشتر نخواهد داشت :

- آی دی یونیک
- نام یا عنوان فیلم
- تاریخ انتشار فیلم
- ژانر فیلم
- قیمت

اما قبل از اینکه بخواهیم این کدها را بنویسیم و توضیح دهیم باید یک کار دیگر انجام دهیم. کدهای اولیه که ویژوال استودیو برایمان تولید کرده است بسیار ساده می باشد

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MvcMovie.Models
{
    public class Movie
    {
    }
}
```

ما چون می خواهیم کار ورود اطلاعات و کار با دیتابیس را انجام دهیم یا به حرفی دیگر می خواهیم از EF کمک بگیریم باید فضای نامی (namespace) به این کدها اضافه کنیم. این فضای نام using System.Data.Entity; می باشد.

خب حالا میریم سراغ کدنویسی آن ۵ فیلدی که در بالا ذکر کردیم.

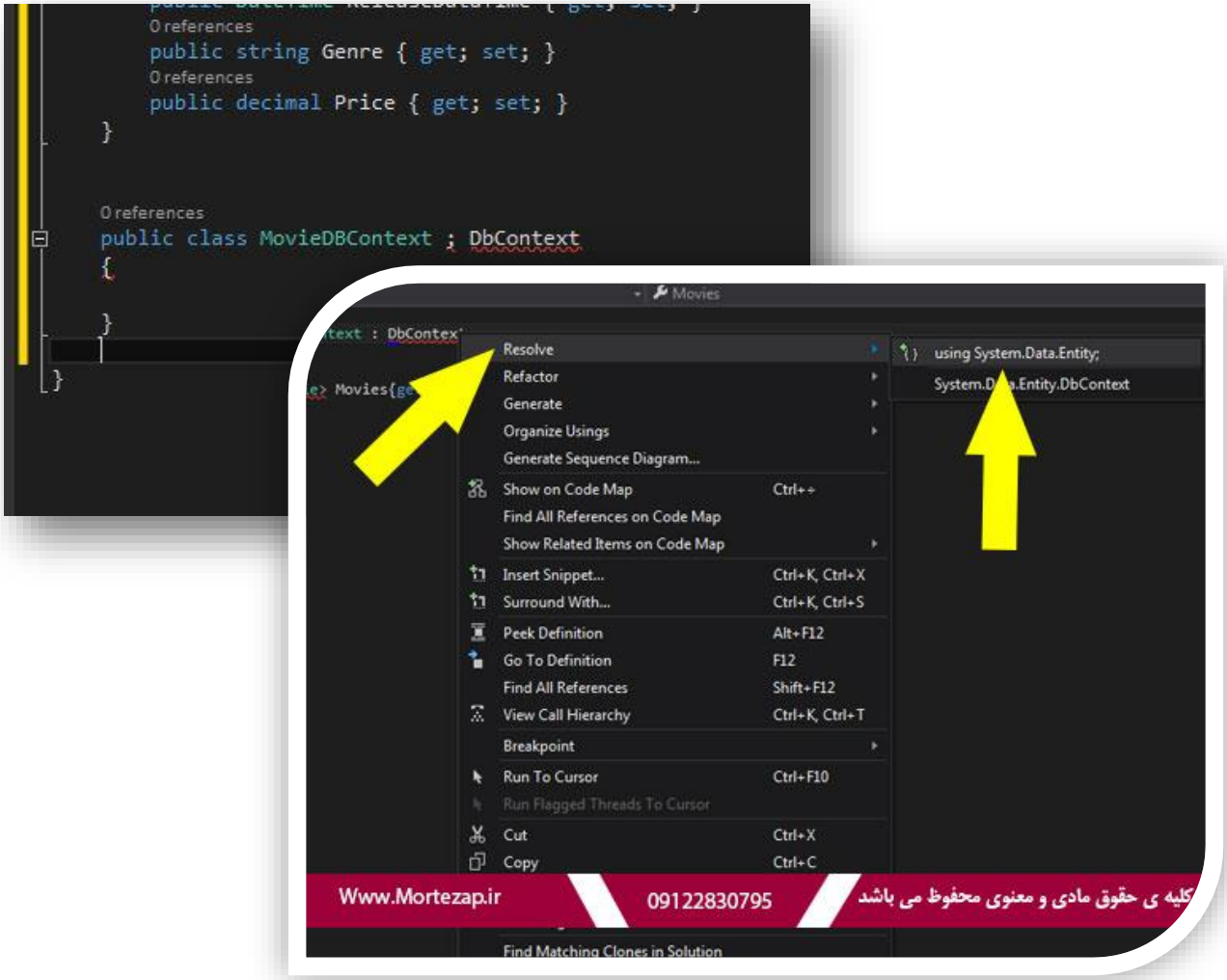
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MvcMovie.Models
{
    public class Movie
    {
        public int ID { get; set; }
        public string MovieTitle { get; set; }
        public DateTime ReleaseDataTime { get; set; }
        public string Genre { get; set; }
        public decimal Price { get; set; }
    }
}
```

یک کلاس عمومی ایجاد کردیم و اطلاعاتی را که میخواهیم به نمایش درآیند و یا دریافت گردند را وارد کردیم. کدها واضح هستند اما کمی هم توضیح می دهم. آی دی فیلم از نوع اینتیجر خواهد بود و با اینکه ما در اینجا پابلیک انتخاب کردیم ولی در ویو به کاربر نمایش نخواهیم داد. عنوان فیلم هم که قاعدتاً باید از نوع رشته باشد و به همین ترتیب تا گزینه ی آخر. یه جورایی انگار ما در اینجا داریم دیتابیس طراحی می کنیم. فرقش این است که در اس کیو ال ما به صورا ویزاردی نیز می توانیم اینکار را انجام دهیم.

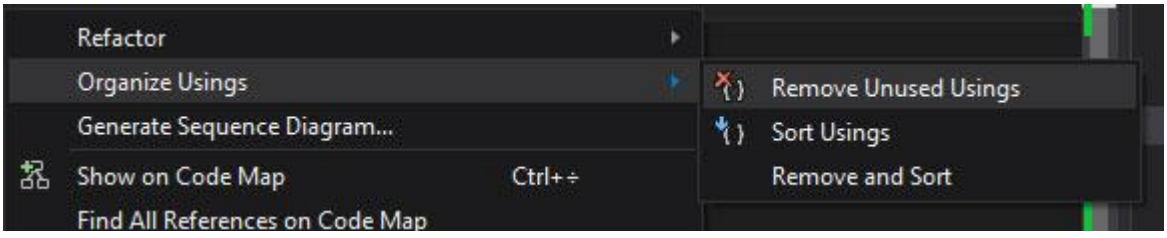
این کلاس بالا هنوز کامل نیست. درست است که تمامی فیلدها را شامل می شود اما یه چیز دیگه کم داره! کلاسی که بتواند کار زخیره، به روز رسانی و ... اطلاعات را به دوش بکشد! کلاسی که پایه را برای EF مهیا کند. ما اسم این کلاس را DbContext می گذاریم و تابعی را می نویسیم که وظیفه اش همین موارد ذکر شده است. یعنی کدهای دیگر را اضافه می کنیم تا شبیه کدهای زیر شود. در خصوص کدها در صفحات بعد توضیح میدهم.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;

namespace MvcMovie.Models
{
    public class Movie
    {
        public int ID { get; set; }
        public string MovieTitle { get; set; }
        public DateTime ReleaseDataTime { get; set; }
        public string Genre { get; set; }
        public decimal Price { get; set; }
    }

    public class MovieDBContext : DbContext
    {
        public DbSet<Movie> Movies{get; set;}
    }
}
```

یک نکته ی دیگه رو هم باید اضافه کنم. شما میتونید فضای نام رو به صورت ویزاردی نیز اضافه کنید. یعنی وقتی زیر Statementتون خط قرمز کشیده میشه روش رایت کلیک کنید و از زیر منوی Resolve گزینه ی فضای نامی مناسبتون رو انتخاب و اتوماتیک ادد کنید



و یا حتی اگر فضاهای نامی که مورد استفاده نیستند رو به صورت اتوماتیک حذف کنید. برای این کار نیز کافیست بر روی صفحه راست کلیک کنید و از منوی Organize Unusing گزینه ی مورد علاقتون رو انتخاب کنید



❖ ایجاد Connection String و کار با SqlServer

کلاسی که ایجاد کردیم تحت عنوان MovieDBContext وظیفه ی برقراری ارتباط با دیتابیس و همچنین ذخیره ی آبجکت های Movie را بر عهده دارد. مطمئناً اولین سوالی که براتون پیش میاد اینه که چطور این کلاس میتونه به دیتابیس SQL متصل بشه. MVC5 این کار رو براتون آسون کرده و همه ی وظیفه رو بر عهده ی EF گذاشته که این

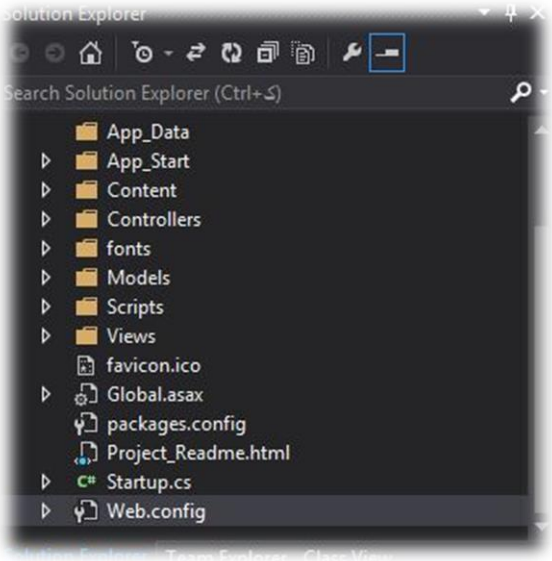
نیز از طریق LocalDB انجام پذیر میشه. ما در این قسمت یاد میگیرم چطور رشته ی اتصال (همیشه با این ترجمه ها و واژگان مشکل داشتم . منظورم همون Connection String هست) رو در فایل Web.config اضافه کنیم.

❖ SQL Server Express LocalDB

LocalDB یک نسخه ی لایت از پروژه ی بزرگ SQL Server هست که معمولاً در فولدر App_Date نگهداری میشه. لوکال دیتابیس به صورت کلی در پروژه های تحت وب بی نیاز از IIS و نرم افزار SQL می باشد البته این بدین معناست که بودن یا نبودن اینها هیچ خللی در روند انجام پروژه های تحت وب ایجاد نمیکنه. و این رو هم اضافه کنم که لوکال دیتابیس به راحتی میتونه به SQL Server و SQL Azure مهاجرت کنه یعنی انتقال پیدا کنه.

در ویژوال استودیو ۲۰۱۳ این لوکال دیتابیس به صورت پیش فرض نصب شده است و نیازی به استفاده و نصب و یا رفع مشکلاتی از پشی تعیین شده نمی باشد. به صورت پیش فرض در EF کلاسی در Connection String وجود دارد با عنوان context و ما نیز برای همین قرارداد در کدهای صفحه ی قبل از نام MovieDBContext استفاده کردیم.

در روت پروژه فایل Web.config را باز کنید. البته یک وب کانفیگ نیز در فولدر ویو وجود دارد که کا با آن کاری نداریم



در کدهای موجود در فایل وب کانفیگ دنبال کد <connectionStrin> بگردید. کدی شبیه کدهای زیر خواهد بود

```
<connectionStrings>
    <add name="DefaultConnection" connectionString="Data
Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\aspnet-MvcMovie-
20141013094548.mdf;Initial Catalog=aspnet-MvcMovie-20141013094548;Integrated
Security=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
```

و به کدهای زیر تغییر میدهیم

```
  <connectionStrings>
    <add name="MovieDBContext" connectionString="Data
Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\Movies.mdf;Integrated
Security=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
```

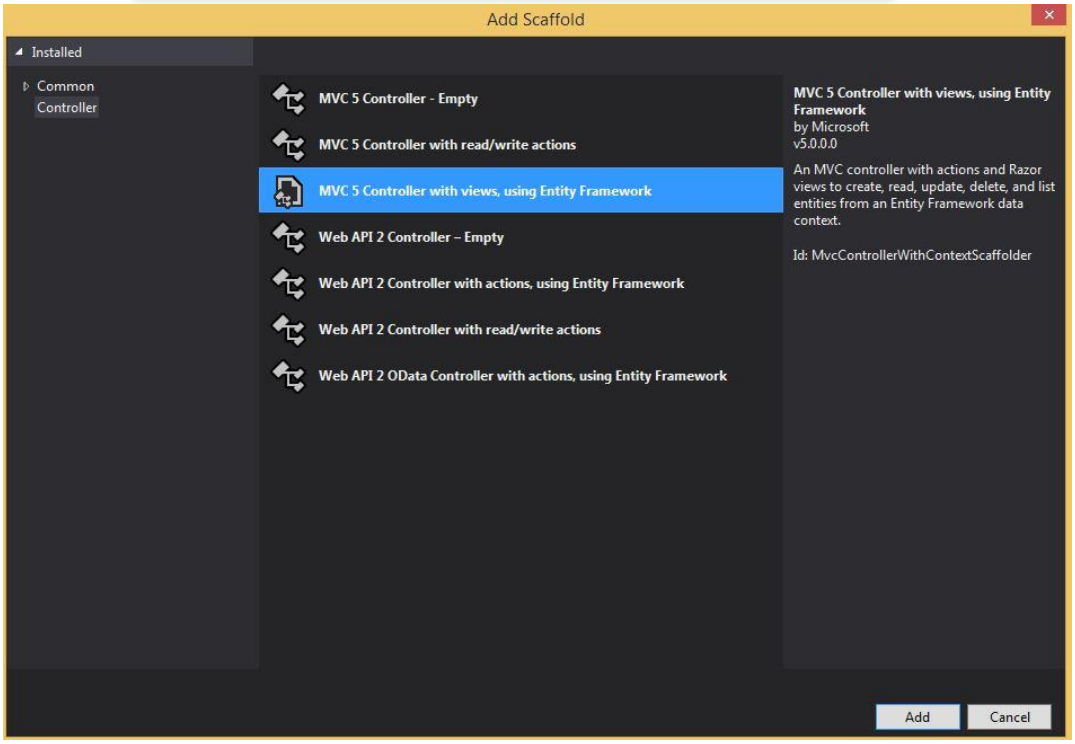
یعنی اسمی که در قسمت Model در کلاس Movie وارد کردیم را اینجا وارد می کنیم و به جای دیتابیس پیشفرضی که گذاشته بود نام آبجکت دیتابیس خودمان که این را نیز باز در همان کلاس Movie برگزیده بودیم را می نویسیم. یعنی اگر ما آن کلاس Movie را تغییر میدادیم و کد public DbSet<Movie> testDB{get; set;} را می نوشتیم ، در این کدها نیز باید اینگونه تایپ میکردیمAttachDbFilename=|DataDirectory|\testDB.mdf

البته می توانید یک دیتابیس جدید اضافه کنید. یعنی بدون اینکه کدهای اولیه را حذف کنید همه را یکجا بنویسید. مانند کد های زیر

```
connectionStrings>
    <add name="DefaultConnection" connectionString="Data
Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\aspnet-MvcMovie-
20141013094548.mdf;Initial Catalog=aspnet-MvcMovie-20141013094548;Integrated
Security=True"
      providerName="System.Data.SqlClient" />
    <add name="MovieDBContext" connectionString="Data
Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\Movies.mdf;Integrated
Security=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
```

❖ دسترسی به Model's Data از طریق Controller

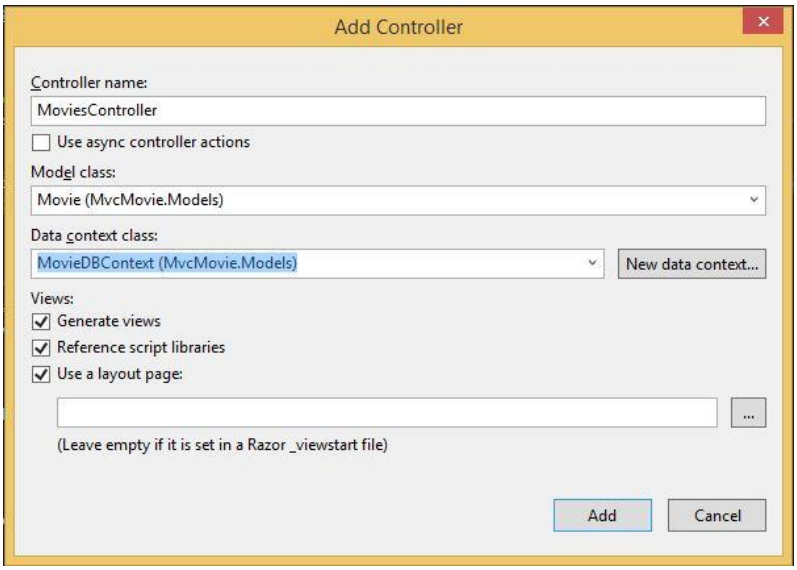
ما همه ی این کارها رو انجام دادیم تا بتونیم در مرورگر یه چیزهایی رو به کاربر نشون بدیم. یعنی این همه کد نوشتیم (خیلی :دی) که به اینجا برسیم. حالا قراره توسط اون کلاسی که درست کردیم نمایش اطلاعات داشته باشیم. اما این کلاسی که نوشتیم فقط مربوط به اطلاعات بود که در بخش Model نوشته شد. حالا برای وارد کردن و نمایش اطلاعات نیز یک کلاس بنویسیم. اما این کلاس اون کلاس نیست :دی این کلاسی-ه که ما باید به کاربر نشون بدیم. همونطور که قبلاً هم توضیح دادیم این وظیفه بر دوش Controller ها هست. پس طبق آموزشی که در صفحات اول داشتیم میریم سراغ فولدر کنترلر و یک کلاس کنترلر ایجاد می کنیم. فقط یک نکته فراموش نشه ، ما توی اون صفحات کلاس کنترلر خالی یا همون MVC5 Controller – Empty ایجاد می کردیم. اما اینجا کمی متفاوته. برای شروع روی فولدر کنترلر راست کلیک کرده و از زیر منوی Add گزینه ی Controller... را برمیگزینیم و در پنجره ی Add Scaffold گزینه ی سوم یا MVC5 Controller with views ,using Entity Framework رو انتخاب می کنیم.



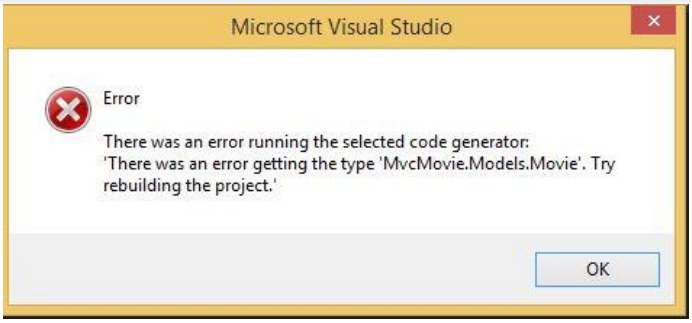
و تنظیمات زیر و براش انجام میدیم:

- اسم این کلاس رل MoviesController را تایپ می کنیم.
- کادر بعدی Movie (MvcMovie.Models) را انتخاب می کنیم
- و از قسمت کلاس Data Context گزینه ی MovieDBContext (MvcMovie.Models)

و در نهایت Add می کنیم



اگر مثل من ارور داشتید باید پروژه را یکبار Build کنید و سپس مراحل بالا را انجام دهید

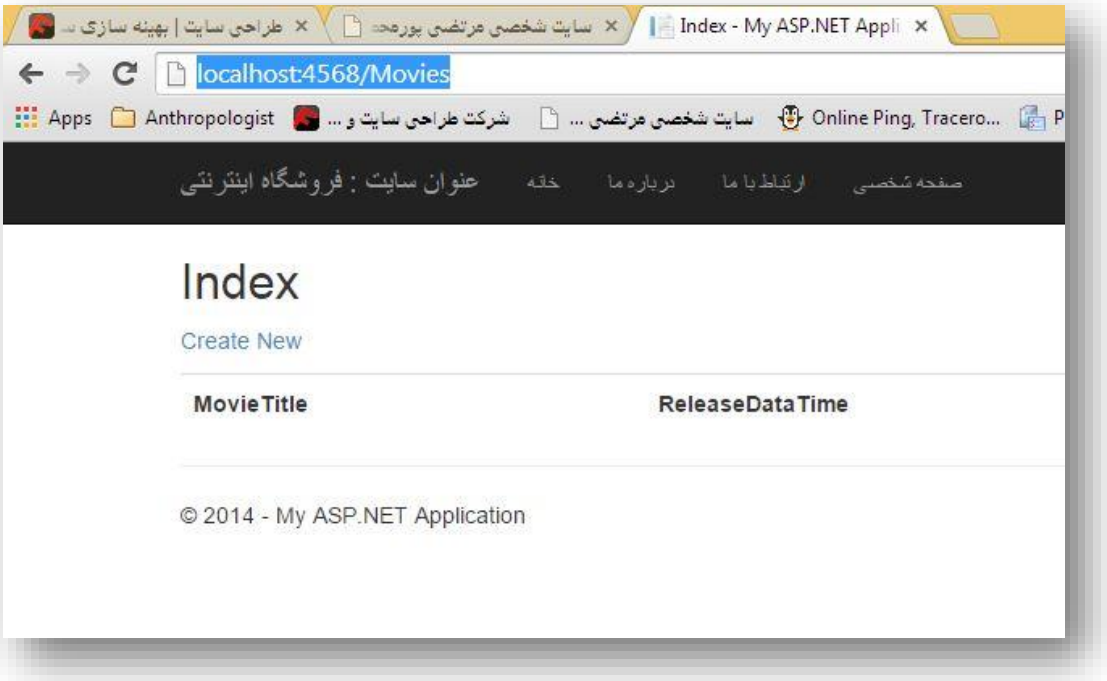


برای بیلد پروژه از کلیدهای ترکیبی CTRL + Shift + B نیز می توانید استفاده کنید

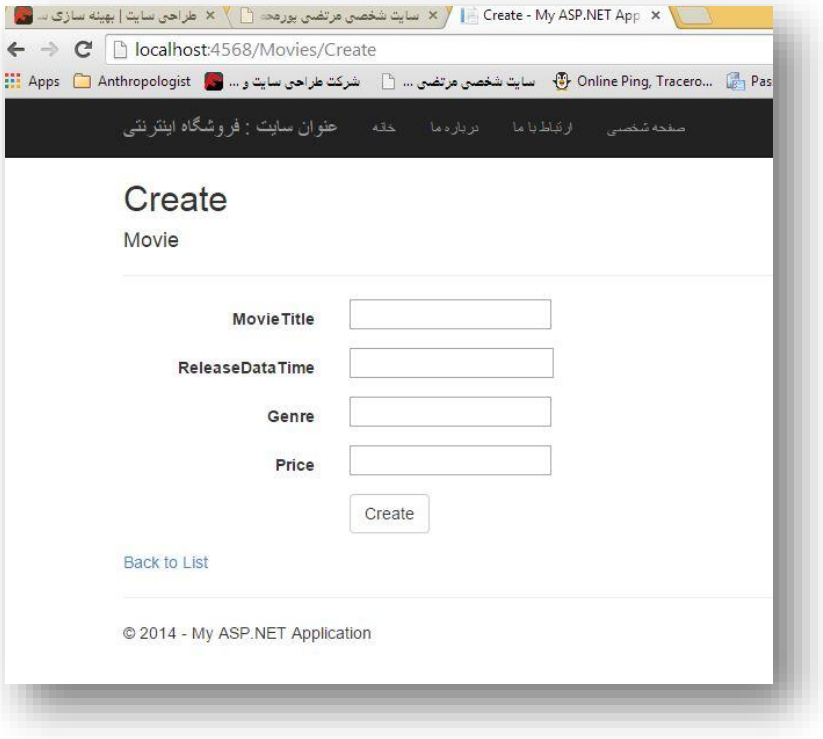
وقتی با موفقیت این کار انجام شد ویژوال استودیو براتون یکسری فایل و فولدر و کد درست کرده :

- فایل MoviesController.cs را در فولدر Controllers
- فولدر Movie در فولدر اصلی View
- ایجاد فایلهای Index.cshtml , Edit.cshtml , Details.cshtml , Delete.cshtml , Create.cshtml در فولدر Movie در فولدر اصلی View

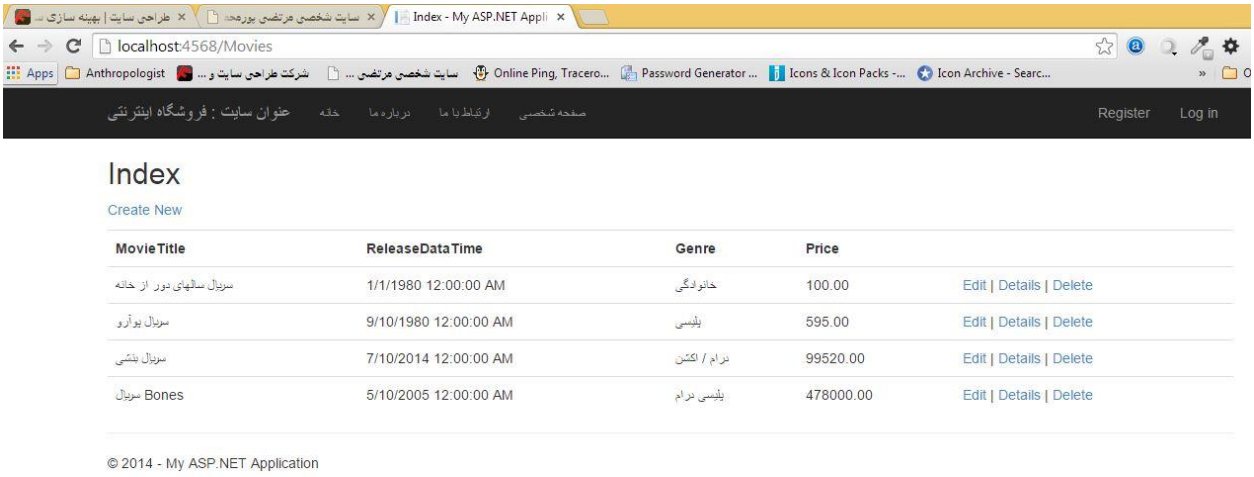
تبریک میگم :دی ویژوال استودیو تمامی اعمال CRUD رو براتون کدنویسی کرده. حتی به جز کلاس نوشتن این کدها براتون صفحاتش رو نیز ایجاد کرده... انصافاً این یکی از شاهکارهای MVC5 هست. شما الان میتونید یک پروژه ی تحت وب کامل رو بدون مشکل اجرا کنید. حتی کار ورود اطلاعات و نمایش بدون مشکل اجرا میشه. فقط کافیه بعد از ران دیباگینگ پروژه آدرس اینترنتی http://localhost:4568/Movies رو وارد کنید تا وارد صفحه بشید :



و یا صفحه ی ورود اطلاعات رو ببینید :

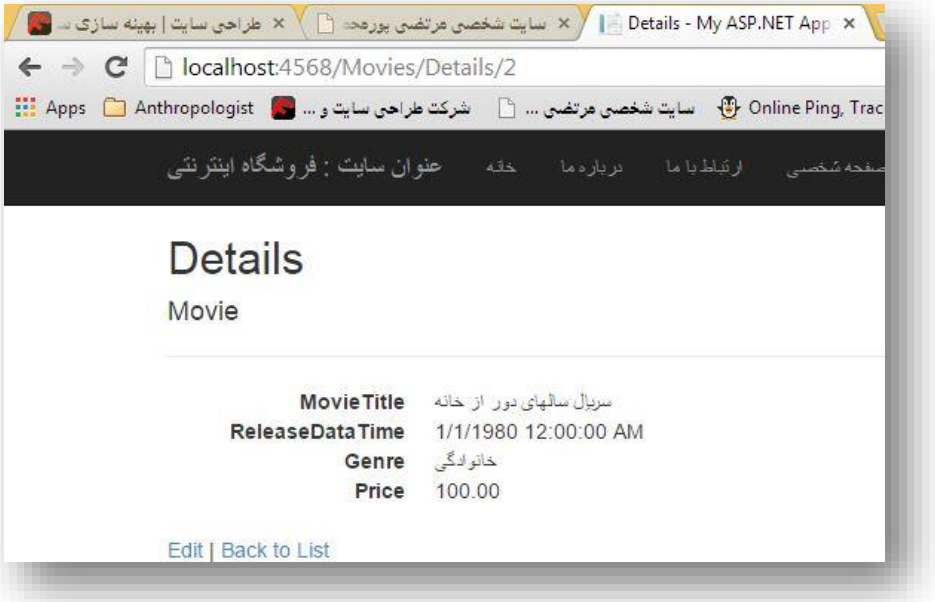


برای اینکه مطمئن بشید کار میکنه یکسری اطلاعات وارد می کنیم:

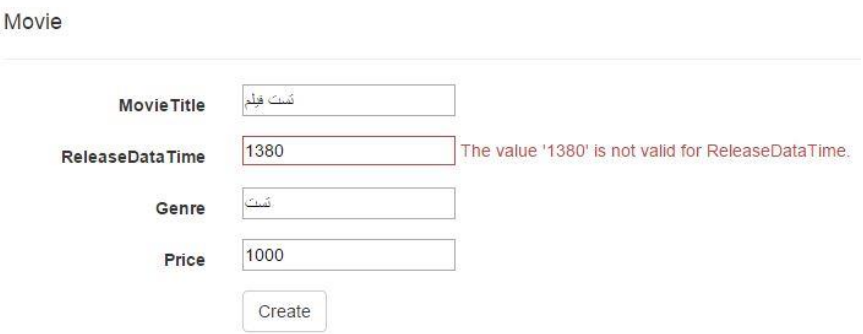


میبینید که بدون مشکل داره کار میکنه . فعلاً مجبوریم از تاریخ میلادی استفاده کنیم تا در مباحث پیشرفته تر بهتون یاد بدم چطور میشه تقویم جلالی اضافه کرد.

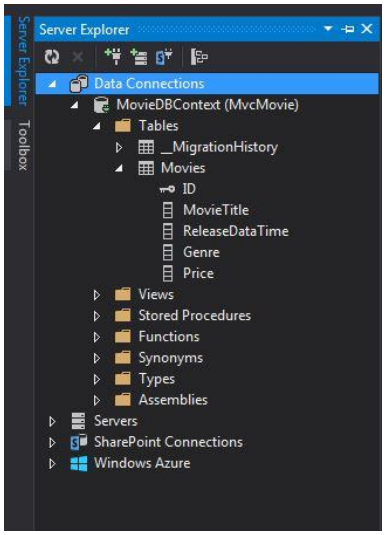
حتی گزینه های دیگه هم به درستی کار میکنه. مثلاً لینک Details یا جزییات هر فیلم نیز کاملاً کار میکنه



حتی اررورها نیز کاملاً به درستی کار می کنند.



برای مشاهده ی دیتابیس هم کافیه روی Server Explorer کلیک کنید تا پنجره ی مربوطه باز بشه و بتونید دیتابیس رو ببینید

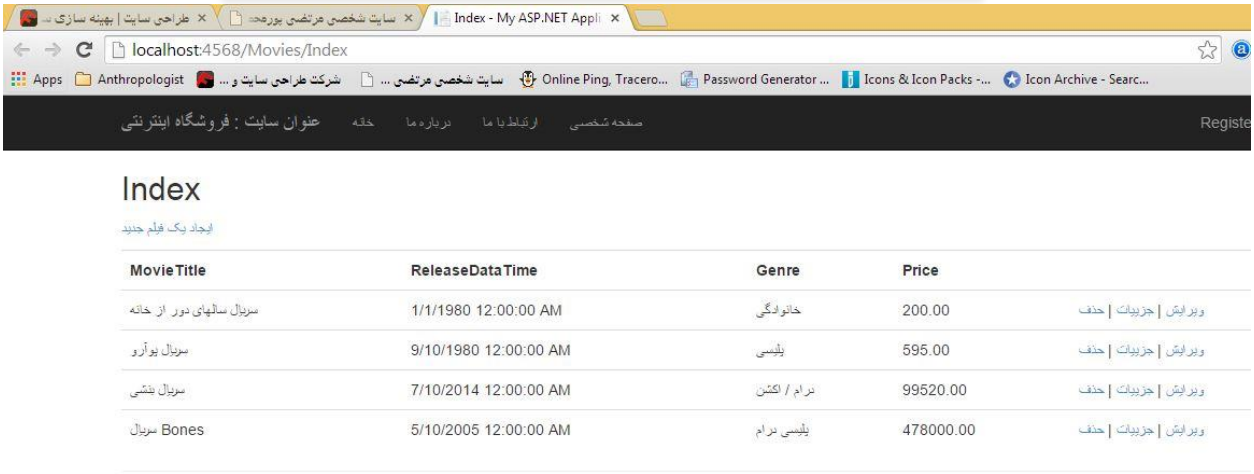


میبینید که بدون خونریزی و فقط با همون چند خط کد خود ویژوال استودیو براتون همه کاری انجام داده.

برای عوض کردن محتوا هم میونید صفحات رو از داخل View عوض کنیدو مثلاً من صفحه ی اول فولدر Movies رو کمی تغییر میدم

```
@model IEnumerable<MvcMovie.Models.Movie>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("جدید فیلم یک ایجاد", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.MovieTitle)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.ReleaseDataTime)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Genre)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Price)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.MovieTitle)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ReleaseDataTime)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Genre)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Price)
        </td>
        <td>
            @Html.ActionLink("ویرایش", "Edit", new { id=item.ID }) |
            @Html.ActionLink("جزییات", "Details", new { id=item.ID }) |
            @Html.ActionLink("حذف", "Delete", new { id=item.ID })
        </td>
    </tr>
}

</table>
```

که شاهد تغییرات در این صفحه خواهم بود. مانند عکس زیر



به همین راحتی شما میتونید صفحات مختلف ایجاد کنید و دیتاهای خودتون رو بدون خونریزی وارد و نمایش بدید.

☑ سخن پایانی

امیدوارم با منطق MVC5 آشنا شده باشید و بدون دردسر تونسته باشید یک پروژه ران کنید. در کتاب بعدی که فصل آخرش رو در حال نگارش دارم، به بررسی خصوصیات بیشتر و پیشرفته تر مانند قابلیت جستجوی پیشرفته و بسیار دقیق، بخش مدیریت، سطوح دسترسی، فروش آنلاین(خرید آنلاین)، طراحی قالب و استفاده از جی کوئری در طراحی و در نهایت به خصوصیات فوقالعاده زیبا و پرکاربرد EF می پردازم.

کدها و پی دی اف این کتاب آموزشی در سایت های زیر موجود هست

www.Faedu.ir | Www.Mortezap.ir | Www.farjadp.com

البته یک نسخه هم در سایت خوب برنامه نویس قرار دادم. یک فایل زیپ که هم پی دی اف موجوده و هم سورسها

منتظر انتقادات و پیشنهاداتتون هستم.

براتون آرزوی موفقیت می کنم

مرتضی پورمحمد

۲۸ مهر ۱۳۹۳