

آموزش کاربردی

Pro

ASP.NET Core MVC

ویراست ۶

آدام فریمن

ترجمه: مهندس نادر نبوی

انتشارات پندار پارس

| | |
|---------------------|--|
| سرشناسه | : فریمن، آدام، ۱۹۷۲ - م. Freeman, Adam |
| عنوان و نام پدیدآور | : آموزش کاربردی Pro ASP.NET Core MVC / آدام فریمن؛ ترجمه نادر نبوی. |
| وضعیت ویراست | : [ویراست ۲]. |
| مشخصات نشر | : تهران: پندار پارس، ۱۳۹۶. |
| مشخصات ظاهری | : xviii، ۷۴۴ ص.؛ مصور، جدول. |
| شابک | : 978-600-8201-53-3 : ۵۳۰۰۰۰ ریال |
| وضعیت فهرست نویسی | : فیبا |
| إدداشت | : عنوان اصلی: Pro ASP.NET Core MVC, 6th ed, 2016 |
| إدداشت | : ویراست قبلی کتاب حاضر تحت عنوان "مرجع کامل ASP.NET Core MVC" به صورت جلدی در سال ۱۳۹۶ منتشر شده است. |
| عنوان دیگر | : مرجع کامل ASP.NET Core MVC. |
| موضوع | : مایکروسافت دات نت فریمورک |
| موضوع | : Microsoft .NET Framework |
| موضوع | : وبگاه‌ها -- برنامه‌های تالیفی |
| موضوع | : Web sites -- Authoring programs |
| موضوع | : علوم کامپیوتر |
| موضوع | : Computer science |
| موضوع | : نرم‌افزار -- مهندسی |
| موضوع | : Software engineering |
| شناسه افزوده | : نبوی، نادر، ۱۳۴۰ - مترجم |
| ده بندی کنگره | : ۷۶۵۸/۷۶۵۶ الف ۱۳۹۶ ف ۴ م ۲۳ /م |
| ده بندی دیویی | : ۰۵/۳۷۶ |
| شماره کتابشناسی ملی | : ۴۹۹۱۴۷۳ |

Telegram Chanel: @PendarePars

انتشارات پندار پارس



دفتر فروش: انقلاب، ابتدای کارگر جنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ www.pendarepars.com
تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸ همراه: ۰۹۲۱۴۳۷۱۹۶۴ info@pendarepars.com

نام کتاب : آموزش کاربردی Pro ASP.NET Core MVC

ناشر : انتشارات پندار پارس

تالیف : Adam Freeman

ترجمه : نادر نبوی

چاپ نخست : دی ماه ۹۶

شمارگان : ۵۰۰ نسخه

طرح جلد : رامین شکراللهی

چاپ، صحافی : روز

قیمت : ۵۳۰۰۰ تومان شابک : ۹۷۸-۶۰۰-۸۲۰۱-۵۳-۳

* هرگونه کپی برداری، تکثیر و چاپ کاغذی یا الکترونیکی از این کتاب بدون اجازه ناشر تخلف بوده و پیگرد قانونی دارد *

سخن مترجم

کتابی که در دست دارید، ویرایش ششم کتاب Pro ASP.NET Core MVC است. در فصل‌های ۱ تا ۳ کتاب در مورد تکامل برنامه‌نویسی سمت سرور میکروسافت، از پروژه‌های وب ASP.NET تا چرخش مثبتی که به سمت پروژه‌های MVC ایجاد شد و در پایان منجر به پروژه‌های Core MVC شد، توضیح جامعی داده شده است. همچنین، با تشریح مفاهیم پایه‌ی MVC، به پیاده‌سازی یک پروژه‌ی کامل می‌پردازیم. خوانندگانی که از پیش با برنامه‌نویسی فرم‌های وب آشنایی داشته‌اند، با مشکلات این پروژه‌ها و دردهای آنها در پیاده‌سازی پروژه‌های بزرگ و سازمانی، دست و پنجه نرم کرده‌اند.

خوشبختانه برای یادگیری معماری جدید Core MVC با خواندن این کتاب، نیاز به آشنایی با واسط‌های برنامه‌نویسی قدیمی وب، که به آنها اشاره شد ندارید. به عنوان تنها پیش‌نیاز لازم، آشنایی با مفاهیم وب به همراه توانایی کار با HTML و CSS، زبان C# به همراه Entity Framework و نوشتار کوئری‌های LINQ، کافی است. معنی این گفته این است که می‌توانید برنامه‌نویسی سمت سرور وب را از ابتدا با همین کتاب شروع کنید.

نویسنده، متن اصلی کتاب را به دو بخش اصلی، که می‌توان آنها را مقدماتی و پیشرفته نامید، تقسیم کرده است. در بخش نخست، همراه با تشریح مفاهیم پایه‌ی MVC به پیاده‌سازی یک پروژه‌ی کامل پرداخته است. همین می‌تواند نقطه‌ی شروع خوبی برای یادگیری Core MVC باشد.

کد کامل پروژه‌ی اصلی کتاب و سایر کدهایی که به شکل مثال در فصل‌های مختلف آورده شده‌اند را می‌توانید با رفتن به آدرس زیر به راحتی به دست آورید:

<http://www.github.com/apress/pro-asp.net-core-mvc>

در ادامه، به تشریح جزئیات مباحث گفته شده در ۱۲ فصل آغازین و تحکیم مبانی تئوریک آن پرداخته شده است. هر فصل، با پیاده‌سازی یک پروژه، به تشریح مباحث آن فصل می‌پردازد که این، موجب کاربردی شدن کتاب برای خواننده‌ای که قصد فراگیری کامل MVC را دارد، خواهد شد.

توجه: کد کامل پروژه‌ی اصلی کتاب و سایر کدهایی که به شکل مثال در فصل‌های مختلف آورده شده‌اند را می‌توانید از آدرس زیر به آسانی به دست آورید:

<http://www.github.com/apress/pro-asp.net-core-mvc>

در پایان، از خوانندگان عزیز تقاضا دارم که پرسش‌ها و مشکلات خود را در سایت انتشارات به آدرس www.pendarepars.com و یا مستقیماً به آدرس پست الکترونیکی خودم، nanavijobmail@yahoo.com در میان بگذارند.

نادر نبوی

پاییز سال ۱۳۹۶

فهرست

| | |
|----|--|
| ۱ | فصل یکم؛ آشنایی با ASP.NET Core MVC |
| ۱ | ۱- تاریخچه‌ی ASP.NET Core MVC |
| ۱ | ۱-۱- پروژه‌های فرم‌های وب |
| ۲ | ۱-۱-۱- مشکلات پروژه‌های فرم‌های وب |
| ۳ | ۱-۲- پروژه‌های MVC قدیمی |
| ۳ | ۱-۲-۱- مشکل پروژه‌های قدیمی MVC |
| ۴ | ۱-۳- فهم ASP.NET Core |
| ۴ | ۱-۳-۱- مزایای اصلی ASP.NET Core |
| ۵ | ۱-۳-۱-۱- معماری MVC |
| ۵ | ۱-۳-۱-۲- گسترش‌پذیری |
| ۶ | ۱-۳-۱-۳- کنترل کامل بر HTML و HTTP |
| ۶ | ۱-۳-۱-۴- آزمایش‌پذیری |
| ۶ | ۱-۳-۱-۵- روش مسیریابی قوی |
| ۷ | ۱-۳-۱-۶- رابط برنامه‌نویسی قوی |
| ۷ | ۱-۳-۱-۷- چند پلتفرمی |
| ۸ | ۱-۳-۱-۷- متن باز بودن |
| ۸ | ۱-۴- نیازمندی‌ها |
| ۸ | ۱-۵- ساختار کتاب |
| ۹ | فصل دوم؛ ایجاد نخستین پروژه MVC |
| ۹ | ۲-۱- نصب ویژوال استدیو |
| ۱۱ | ۲-۲- ایجاد پروژه جدید MVC |
| ۱۴ | ۲-۲-۱- افزودن کنترلر به پروژه |
| ۱۶ | ۲-۲-۲- بررسی و فهم مسیرها |
| ۱۶ | ۲-۳- پردازش و نمایش صفحات وب |
| ۱۷ | ۲-۳-۱- ایجاد نما |
| ۱۹ | ۲-۳-۲- خروجی پویا |
| ۲۱ | ۲-۴- پروژه‌ی ساده‌ای برای ورود اطلاعات |
| ۲۱ | ۲-۴-۱- تنظیم سناریوی پروژه |
| ۲۲ | ۲-۴-۲- طراحی مدل داده |
| ۲۳ | ۲-۴-۳- ایجاد نمای مقید به داده |
| ۲۵ | ۲-۴-۴- متصل کردن اکشن‌ها به وسیله‌ی لینک |
| ۲۶ | ۲-۴-۵- ایجاد فرم ورود داده‌ها |
| ۲۷ | ۲-۵- دریافت اطلاعات فرم |
| ۲۹ | ۲-۵-۱- استفاده از مقیدسازی مدل |
| ۳۰ | ۲-۵-۲- ذخیره‌سازی اطلاعات فرم |
| ۳۱ | ۲-۶- نمایش پاسخ‌ها |

| | |
|----|---|
| ۳۴ | ۲-۷ اعتبارسنجی داده‌های فرم |
| ۳۶ | ۲-۷-۱ مشخص کردن فیلدهای نادرست فرم |
| ۳۹ | ۲-۸ کار بر روی ظاهر سایت |
| ۳۹ | ۲-۸-۱ ظاهر نمای خوش‌آمد |
| ۴۰ | ۲-۸-۲ RsvpForm ظاهر نمای |
| ۴۱ | ۲-۸-۳ Thanks ظاهر نمای |
| ۴۲ | ۲-۸-۴ ListResponses.cshtml ظاهر نمای |
| ۴۵ | فصل سوم؛ معماری MVC |
| ۴۵ | ۳-۱ تاریخچه‌ی MVC |
| ۴۵ | ۳-۲ آشنایی با الگوی MVC |
| ۴۶ | ۳-۲-۱ فهم مدل |
| ۴۷ | ۳-۲-۲ فهم کنترلر |
| ۴۷ | ۳-۲-۳ فهم نما |
| ۴۸ | ۳-۲-۴ پیاده‌سازی MVC در ASP.NET |
| ۴۸ | ۳-۳ مقایسه‌ی MVC با دیگر معماری‌ها |
| ۴۹ | ۳-۳-۱ آشنایی با معماری Smart UI |
| ۵۰ | ۳-۳-۲ آشنایی با معماری Model-View |
| ۵۱ | ۳-۴ گونه‌های مختلف MVC |
| ۵۱ | ۳-۴-۱ آشنایی با معماری مدل-نما-نمایشگر |
| ۵۲ | ۳-۴-۲ معماری Model-View-View-Model |
| ۵۲ | ۳-۵ آشنایی با ساختار پروژه‌های ASP.NET Core MVC |
| ۵۲ | ۳-۵-۱ ایجاد پروژه |
| ۵۶ | ۳-۶ قراردادهای MVC |
| ۵۶ | ۳-۶-۱ قراردادهای مربوط به کنترلرها |
| ۵۷ | ۳-۶-۲ قراردادهای مربوط به نماها |
| ۵۷ | ۳-۶-۳ قراردادهای مربوط به الگوی صفحه‌ها |
| ۵۹ | فصل چهارم؛ ویژگی‌های مهم C# |
| ۵۹ | ۴-۱ ایجاد پروژه |
| ۶۱ | ۴-۱-۱ فعال کردن ASP.NET Core MVC |
| ۶۲ | ۴-۲ افزودن عناصر پروژه‌ی MVC |
| ۶۲ | ۴-۲-۱ ایجاد مدل |
| ۶۲ | ۴-۲-۲ ایجاد نما و کنترلر |
| ۶۳ | ۴-۳ کاربرد عملگر شرطی Null |
| ۶۶ | ۴-۵ استفاده از خاصیت‌های خودکار در کلاس‌ها |
| ۶۷ | ۴-۵-۱ خاصیت‌های خودکار فقط خواندنی |
| ۶۸ | ۴-۶ ترکیب رشته‌ها |
| ۶۹ | ۴-۷ مقداردهی آغازین کلکسیون‌ها و اشیاء |
| ۷۲ | ۴-۸ استفاده از متدهای گسترش‌دهنده |
| ۷۳ | ۴-۸-۱ کاربرد متدهای گسترش‌دهنده در رابطه با اینترفیس‌ها |

| | |
|-----------------|---|
| ۷۵..... | ۴-۸-۲ متدهای گسترش دهنده‌ی فیلترکننده |
| ۷۶..... | ۴-۹ عبارت‌های لاندایند |
| ۷۷..... | ۴-۹-۱ تعریف تابع با عبارت لاندایند |
| ۷۹..... | ۴-۹-۲ عبارت‌های لاندایند برای متدها و خصوصیت‌ها |
| ۸۱..... | ۴-۱۰ بیان ضمنی نوع متغیر و انواع بی‌نام |
| ۸۲..... | ۴-۱۰-۱ کاربرد انواع بی‌نام |
| ۸۳..... | ۴-۱۱ متدهای آسنکرون |
| ۸۶..... | ۴-۱۲ دسترسی به نام‌ها |
| ۸۹..... | فصل پنجم؛ کار با Razor |
| ۹۰..... | ۵-۱ آماده کردن پروژه |
| ۹۱..... | ۵-۱-۱ تعریف Model |
| ۹۱..... | ۵-۱-۲ ایجاد کنترلر |
| ۹۲..... | ۵-۱-۳ ایجاد نما |
| ۹۳..... | ۵-۲ کار با شیء مدل |
| ۹۴..... | ۵-۲-۱ استفاده از @import |
| ۹۶..... | ۵-۳ کار با الگوی صفحه |
| ۹۶..... | ۵-۳-۱ ایجاد الگو |
| ۹۸..... | ۵-۳-۲ کاربرد الگو در نما |
| ۹۹..... | ۵-۳-۳ کاربرد فایل ViewStart |
| ۱۰۰..... | ۵-۴ عبارت‌های Razor |
| ۱۰۱..... | ۵-۴-۱ درج داده‌ها |
| ۱۰۲..... | ۵-۴-۲ تنظیم مقدار صفت تگ‌ها |
| ۱۰۳..... | ۵-۴-۳ عبارت‌های شرطی Razor |
| ۱۰۵..... | ۵-۴-۴ آرایه‌ها و کلکسیون‌ها در Razor |
| ۱۰۷..... | فصل ششم؛ کار با ویژوال استدیو |
| ۱۰۷..... | ۶-۱ آماده‌سازی پروژه |
| ۱۰۸..... | ۶-۱-۱ ایجاد مدل |
| ۱۰۹..... | ۶-۱-۲ ایجاد نما و کنترلر |
| ۱۱۰..... | ۶-۲ مدیریت بسته‌های نرم‌افزاری پروژه |
| ۱۱۰..... | ۶-۲-۱ آشنایی با NuGet |
| ۱۱۳..... | ۶-۲-۲ آشنایی با Bower |
| ۱۱۵..... | ۶-۳ آشنایی با روش توسعه‌ی تکرارشونده |
| ۱۱۶..... | ۶-۳-۱ تغییر کد نماها |
| ۱۱۷..... | ۶-۳-۲ تغییر کد کلاس‌ها |
| ۱۱۷..... | ۶-۳-۲-۱ کامپایل خودکار کلاس‌ها |
| ۱۱۹..... | ۶-۳-۲-۲ فعال کردن صفحه‌های استثناها |
| ۱۲۰..... | ۶-۳-۲-۳ استفاده از Debugger |
| ۱۲۱..... | ۶-۳-۲-۴ کاربرد نقاط توقف |
| ۱۲۲..... | ۶-۳-۲-۵ مشاهده‌ی مقادیر داده‌ها |

| | |
|-----|--|
| ۱۲۴ | پنجره‌ی متغیرهای محلی |
| ۱۲۵ | متصل کردن مرورگر به ویژوال استدیو |
| ۱۲۸ | استفاده از چندین مرورگر |
| ۱۲۹ | انتشار جاوا اسکریپت و CSS |
| ۱۳۰ | ارسال محتویات ایستا |
| ۱۳۱ | افزودن محتوای ایستا |
| ۱۳۳ | فشرده‌سازی و بسته‌بندی محتوای ایستا |
| ۱۳۷ | فصل هفتم؛ آزمایش‌های واحد پروژه‌های MVC |
| ۱۳۸ | ۷-۱ پروژه‌ی فصل هفتم |
| ۱۳۸ | ۷-۱-۱ افزودن متدهای اکشن پروژه |
| ۱۳۹ | ۷-۱-۲ ایجاد فرم ورود داده |
| ۱۳۹ | ۷-۱-۳ ویرایش نمای Index |
| ۱۴۱ | ۷-۲ آزمایش واحد پروژه‌های MVC |
| ۱۴۱ | ۷-۲-۱ ایجاد پروژه‌ی آزمایش |
| ۱۴۳ | ۷-۲-۱-۱ پی‌کرنندی پروژه‌ی آزمایش |
| ۱۴۳ | ۷-۲-۱-۲ تنظیم رفرنس پروژه‌ی اصلی |
| ۱۴۴ | ۷-۲-۲ نوشتن و اجرای کد آزمایش‌های واحد |
| ۱۴۷ | ۷-۲-۳ جداسازی کد برای آزمایش واحد |
| ۱۵۴ | ۷-۳ بهبود کارایی آزمایش‌های واحد |
| ۱۵۴ | ۷-۳-۱ پارامتری کردن آزمایش‌های واحد |
| ۱۵۶ | ۷-۳-۲ دستیابی به داده‌های آزمایشی متد یا خاصیت |
| ۱۵۸ | ۷-۳-۳ بهبود پیاده‌سازی‌های ساختگی |
| ۱۶۰ | ۷-۳-۴ استفاده از نرم‌افزار مقلد |
| ۱۶۲ | ۷-۳-۵ ایجاد پروژه‌ی Moq |
| ۱۶۵ | فصل هشتم؛ پروژه‌ی فروشگاه ورزشی |
| ۱۶۶ | ۸-۱ آغاز کار |
| ۱۶۶ | ۸-۱-۱ ایجاد پروژه |
| ۱۶۷ | ۸-۱-۲ افزودن بسته‌های NuGet |
| ۱۶۸ | ۸-۱-۳ ایجاد ساختار پوشه‌ها |
| ۱۶۹ | ۸-۱-۴ پی‌کرنندی پروژه |
| ۱۷۱ | ۸-۱-۵ ایجاد پروژه‌ی آزمایش واحد |
| ۱۷۲ | ۸-۱-۶ اجرای پروژه |
| ۱۷۳ | ۸-۲ کار با مدل دامنه |
| ۱۷۴ | ۸-۲-۱ ایجاد مخزن داده‌ها |
| ۱۷۴ | ۸-۲-۲ ایجاد مخزن داده‌های ساختگی |
| ۱۷۵ | ۸-۲-۳ ثبت سرویس مخزن داده‌ها |
| ۱۷۶ | ۸-۳ نمایش لیستی از محصولات |
| ۱۷۶ | ۸-۳-۱ کنترلر |
| ۱۷۷ | ۸-۳-۲ نما و تنظیمات آن |

| | | |
|-----|---------------------------------------|--------|
| ۱۷۸ | مسیرهای پیش فرض | ۸-۳-۳ |
| ۱۷۹ | اجرای برنامه | ۸-۳-۴ |
| ۱۸۰ | آماده کردن پایگاه داده | ۸-۴ |
| ۱۸۱ | Entity Framework نصب | ۸-۴-۱ |
| ۱۸۱ | کلاس‌های پایگاه داده | ۸-۴-۲ |
| ۱۸۴ | کلاس مخزن داده‌ها | ۸-۴-۳ |
| ۱۸۴ | تعریف رشته‌ی اتصال | ۸-۴-۴ |
| ۱۸۵ | پی‌کربندی پروژه | ۸-۴-۵ |
| ۱۸۷ | برپاسازی پایگاه داده | ۸-۴-۶ |
| ۱۸۸ | صفحه‌بندی داده‌های نما | ۸-۵ |
| ۱۸۸ | نمایش لینک‌های صفحه‌ها | ۸-۵-۱ |
| ۱۸۹ | بخش نما-مدل | ۸-۵-۲ |
| ۱۸۹ | Tag Helper کلاس | ۸-۵-۳ |
| ۱۹۰ | داده‌های نما-مدل | ۸-۵-۴ |
| ۱۹۲ | نمایش لینک‌های صفحه‌ها | ۸-۵-۵ |
| ۱۹۳ | بهبود URLها | ۸-۶ |
| ۱۹۴ | شکل‌دهی نماها | ۸-۷ |
| ۱۹۴ | Bootstrap نصب بسته‌ی | ۸-۷-۱ |
| ۱۹۷ | ایجاد نمای جزئی | ۸-۷-۲ |
| ۱۹۹ | فصل نهم؛ پیمایش سایت | |
| ۱۹۹ | کنترل‌های پیمایش | ۹-۱ |
| ۱۹۹ | فیلتر کردن محصولات | ۹-۱-۱ |
| ۲۰۱ | بازیابی طرح مسیریابی | ۹-۱-۲ |
| ۲۰۴ | ایجاد فهرست گروه محصول | ۹-۱-۳ |
| ۲۰۶ | لیست گروه محصول | ۹-۱-۴ |
| ۲۰۷ | ایجاد نما | ۹-۱-۵ |
| ۲۱۱ | سید خرید | ۹-۲ |
| ۲۱۲ | تعریف مدل سید خرید | ۹-۲-۱ |
| ۲۱۲ | افزودن به سید خرید | ۹-۲-۲ |
| ۲۱۴ | استفاده از نشست | ۹-۲-۳ |
| ۲۱۵ | کنترلر سید خرید | ۹-۲-۴ |
| ۲۱۷ | متدهای توسعه‌یافته برای نشست‌ها | ۹-۲-۵ |
| ۲۱۸ | نمایش محتوای سید | ۹-۲-۶ |
| ۲۲۱ | فصل دهم؛ تکمیل سید خرید | |
| ۲۲۱ | بهبود سید خرید با سرویس | ۱۰-۱ |
| ۲۲۱ | کلاس کمکی سید خرید | ۱۰-۱-۱ |
| ۲۲۲ | ثبت سرویس کمکی سید | ۱۰-۱-۲ |
| ۲۲۳ | ساده کردن کنترلر سید خرید | ۱۰-۱-۳ |
| ۲۲۴ | تکمیل کارآیی سید خرید | ۱۰-۲ |

| | |
|-----|---|
| ۲۲۴ | حذف کالا از سبد خرید |
| ۲۲۶ | لیست کالاهای سبد خرید |
| ۲۲۶ | استفاده از فونت‌های Awesome |
| ۲۲۷ | ایجاد نما و کلاس عنصر نما |
| ۲۲۹ | ثبت سفارش |
| ۲۲۹ | ایجاد کلاس مدل |
| ۲۳۰ | افزودن فرآیند ثبت سفارش |
| ۲۳۲ | پردازش سفارش |
| ۲۳۲ | گسترش پایگاه داده |
| ۲۳۳ | مخزن داده‌های سفارش |
| ۲۳۵ | تکمیل کنترلر Order |
| ۲۳۶ | نمایش خطاهای اعتبارسنجی |
| ۲۳۷ | نمایش صفحه‌ی پایانی |
| ۲۳۹ | فصل یازدهم؛ مدیریت برنامه. |
| ۲۳۹ | ۱۱-۱ مدیریت سفارش |
| ۲۳۹ | ۱۱-۱-۱ تغییرات مدل |
| ۲۴۰ | ۱۱-۱-۲ اکشن‌ها و نماها |
| ۲۴۳ | ۱۱-۲ مدیریت کالاها |
| ۲۴۳ | ۱۱-۲-۱ ایجاد کنترلر CRUD |
| ۲۴۴ | ۱۱-۲-۲ ایجاد نما برای کنترلر Admin |
| ۲۴۵ | ۱۱-۲-۳ ویرایش کالاها |
| ۲۴۶ | ۱۱-۲-۳-۱ متد اکشن Edit |
| ۲۴۶ | ۱۱-۲-۳-۲ ایجاد نمای Edit |
| ۲۴۸ | ۱۱-۲-۳-۳ مخزن داده‌های کالا |
| ۲۴۹ | ۱۱-۲-۳-۴ ویرایش درخواست‌های POST |
| ۲۵۰ | ۱۱-۲-۳-۵ نمایش پیام تأیید |
| ۲۵۱ | ۱۱-۲-۳-۶ اعتبارسنجی مدل |
| ۲۵۳ | ۱۱-۲-۳-۷ اعتبارسنجی سمت مشتری |
| ۲۵۵ | ۱۱-۲-۴ درج محصول جدید |
| ۲۵۷ | ۱۱-۲-۵ حذف محصول |
| ۲۶۱ | فصل دوازدهم؛ امنیت و انتشار پروژه. |
| ۲۶۱ | ۱۲-۱ مدیریت و امنیت |
| ۲۶۱ | ۱۲-۱-۱ بسته‌ی تشخیص هویت |
| ۲۶۲ | ۱۲-۱-۲ پایگاه داده‌ی هویت‌ها |
| ۲۶۳ | ۱۲-۱-۲-۱ تعریف رشته‌ی اتصال |
| ۲۶۳ | ۱۲-۱-۲-۲ پیکربندی پروژه |
| ۲۶۴ | ۱۲-۱-۲-۳ تعریف داده‌های پایه |
| ۲۶۵ | ۱۲-۱-۲-۴ همگام‌سازی پایگاه داده با مدل |
| ۲۶۶ | ۱۲-۱-۳ تعیین سیاست تشخیص هویت |

| | |
|-----|--|
| ۲۶۸ | ۱۲-۱-۴ کنترلر حساب کاربری و نماهای آن |
| ۲۷۱ | ۱۲-۲ انتشار پروژه |
| ۲۷۱ | ۱۲-۲-۱ ایجاد پایگاه‌های داده |
| ۲۷۲ | ۱۲-۲-۱-۱ باز کردن دسترسی فایروال برای پیکربندی |
| ۲۷۲ | ۱۲-۲-۱-۲ دسترسی به رشته‌های اتصال |
| ۲۷۳ | ۱۲-۲-۲ آماده کردن پروژه برای انتشار |
| ۲۷۳ | ۱۲-۲-۲-۱ کنترلر خطا و نمای آن |
| ۲۷۳ | ۱۲-۲-۲-۲ تنظیمات پایگاه داده |
| ۲۷۴ | ۱۲-۲-۲-۳ پیکربندی پروژه |
| ۲۷۷ | ۱۲-۲-۳ انتشار پروژه |
| ۲۸۱ | فصل سیزدهم؛ پیکربندی پروژه‌ها |
| ۲۸۴ | ۱۳-۱ آماده‌سازی پروژه |
| ۲۸۵ | ۱۳-۲ فایل‌های پیکربندی JSON |
| ۲۸۷ | ۱۳-۲-۱ پیکربندی سالوشن |
| ۲۸۹ | ۱۳-۲-۲ پیکربندی پروژه |
| ۲۹۰ | ۱۳-۲-۲-۱ dependencies تنظیمات بخش |
| ۲۹۱ | ۱۳-۲-۲-۲ tools تنظیمات بخش |
| ۲۹۲ | ۱۳-۳ آشنایی با کلاس Program |
| ۲۹۴ | ۱۳-۴ آشنایی با کلاس Startup |
| ۲۹۵ | ۱۳-۴-۱ کارکرد کلاس Startup |
| ۲۹۷ | ۱۳-۴-۲ آشنایی با سرویس‌های ASP.NET |
| ۳۰۰ | ۱۳-۴-۲-۱ آشنایی با سرویس‌های MVC |
| ۳۰۰ | ۱۳-۴-۳ آشنایی با میان‌افزارهای MVC |
| ۳۰۱ | ۱۳-۴-۳-۱ میان‌افزار تولید محتوا |
| ۳۰۳ | ۱۳-۴-۳-۲ کاربرد سرویس در میان‌افزار |
| ۳۰۴ | ۱۳-۴-۳-۳ میان‌افزار میان‌بر |
| ۳۰۶ | ۱۳-۴-۳-۴ میان‌افزار ویرایش درخواست |
| ۳۰۹ | ۱۳-۴-۳-۵ میان‌افزار ویرایش پاسخ |
| ۳۱۱ | ۱۳-۴-۴ چگونگی فراخوانی متد (Configure) |
| ۳۱۱ | ۱۳-۴-۴-۱ استفاده از Application Builder |
| ۳۱۳ | ۱۳-۴-۴-۲ استفاده از اطلاعات میزبانی |
| ۳۱۶ | ۱۳-۴-۴-۳ استفاده از Logging factory |
| ۳۱۹ | ۱۳-۴-۴-۳-۱ ایجاد سیستم لاگ شخصی |
| ۳۲۰ | ۱۳-۴-۵ سایر میان‌افزارهای مهم |
| ۳۲۰ | ۱۳-۴-۵-۱ فعال کردن مدیریت خطاها |
| ۳۲۳ | ۱۳-۴-۵-۲ فعال کردن لینک مرورگر |
| ۳۲۵ | ۱۳-۴-۵-۳ فعال کردن محتوای استاتیک |
| ۳۲۶ | ۱۳-۴-۶ کاربرد داده‌های پیکربندی |
| ۳۲۷ | ۱۳-۴-۶-۱ خواندن داده‌های پیکربندی |

| | | |
|-----|--|----------|
| ۳۳۰ | استفاده داده‌های پیکربندی | ۱۳-۴-۶-۲ |
| ۳۳۱ | داده‌های پیکربندی در میان‌افزارهای پیش‌ساخته | ۱۳-۴-۶-۳ |
| ۳۳۲ | MVC | ۱۳-۵ |
| ۳۳۴ | پیچیده | ۱۳-۶ |
| ۳۳۴ | ایجاد فایل‌های خارجی پیکربندی | ۱۳-۶-۱ |
| ۳۳۵ | ایجاد متدهای پیکربندی | ۱۳-۶-۲ |
| ۳۳۷ | ایجاد کلاس‌های پیکربندی | ۱۳-۶-۳ |
| ۳۴۱ | فصل چهاردهم؛ مسیریابی در MVC | |
| ۳۴۲ | آماده‌سازی پروژه | ۱۴-۱ |
| ۳۴۳ | مدل | ۱۴-۱-۱ |
| ۳۴۴ | ایجاد کنترلر | ۱۴-۱-۲ |
| ۳۴۵ | ایجاد نما | ۱۴-۱-۳ |
| ۳۴۷ | آشنایی با الگوهای آدرس | ۱۴-۲ |
| ۳۴۸ | ایجاد و ثبت یک مسیر | ۱۴-۲-۱ |
| ۳۵۰ | تعریف مقادیر پیش‌فرض | ۱۴-۳ |
| ۳۵۳ | بخش‌های استاتیک آدرس | ۱۴-۴ |
| ۳۵۸ | تعریف متغیرهای شخصی | ۱۴-۵ |
| ۳۶۰ | متغیرهای شخصی در متد اکشن | ۱۴-۵-۱ |
| ۳۶۱ | تعریف بخش دلخواه در مسیر | ۱۴-۵-۲ |
| ۳۶۳ | تعریف مسیریابی با تعداد بخش‌های متغیر | ۱۴-۵-۳ |
| ۳۶۵ | محدود کردن مسیرها | ۱۴-۶ |
| ۳۶۹ | محدودسازی مسیر با عبارت دلخواه | ۱۴-۶-۱ |
| ۳۷۱ | کاربرد قیود نوع و مقدار | ۱۴-۶-۲ |
| ۳۷۲ | ترکیب قیدها | ۱۴-۶-۳ |
| ۳۷۴ | تعریف قیدهای شخصی | ۱۴-۶-۴ |
| ۳۷۶ | مسیریابی به وسیله‌ی صفات | ۱۴-۷ |
| ۳۷۷ | کاربرد مسپردهی صفات | ۱۴-۷-۱ |
| ۳۷۹ | تغییر نام متد اکشن | ۱۴-۷-۲ |
| ۳۸۰ | مسیرهای پیچیده‌تر | ۱۴-۷-۳ |
| ۳۸۱ | قیدهای مسیر | ۱۴-۷-۴ |
| ۳۸۳ | فصل پانزدهم؛ مسیریابی پیشرفته | |
| ۳۸۳ | آماده‌سازی پروژه فصل پانزدهم | ۱۵-۱ |
| ۳۸۴ | آدرس‌های خروجی در نماها | ۱۵-۲ |
| ۳۸۷ | دسترسی به کنترلرهای دیگر | ۱۵-۲-۱ |
| ۳۸۹ | ارسال مقادیر به متغیرهای مسیر | ۱۵-۲-۲ |
| ۳۹۲ | ایجاد آدرس‌های کامل | ۱۵-۲-۳ |
| ۳۹۳ | ایجاد آدرس از مسیر مشخص | ۱۵-۲-۴ |
| ۳۹۴ | ایجاد مستقیم آدرس، نه لینک | ۱۵-۳ |
| ۳۹۵ | ایجاد آدرس در متدهای اکشن | ۱۵-۳-۱ |

| | | |
|-----|-------|--|
| ۳۹۶ | | ۱۵-۴ شخصی کردن سیستم مسیریابی |
| ۳۹۶ | | ۱۵-۴-۱ تغییر پیکربندی سیستم مسیریابی |
| ۳۹۸ | | ۱۵-۵ ایجاد کلاسی برای مسیریابی |
| ۳۹۹ | | ۱۵-۵-۱ مدیریت آدرس‌های ورودی |
| ۴۰۲ | | ۱۵-۵-۱-۱ کاربرد کلاس شخصی مسیر |
| ۴۰۳ | | ۱۵-۵-۱-۲ آدرس‌دهی کنترلرها |
| ۴۰۷ | | ۱۵-۵-۲ ایجاد آدرس‌های خروجی |
| ۴۱۰ | | ۱۵-۶ کار با ناحیه‌ها |
| ۴۱۰ | | ۱۵-۶-۱ ایجاد یک ناحیه |
| ۴۱۱ | | ۱۵-۶-۲ ایجاد مسیر برای ناحیه |
| ۴۱۲ | | ۱۵-۶-۳ کنترلرها و نماهای ناحیه |
| ۴۱۵ | | ۱۵-۶-۴ ایجاد لینک به اکشن در ناحیه |
| ۴۱۷ | | فصل شانزدهم؛ کنترلرها و متدهای اکشن |
| ۴۱۸ | | ۱۶-۱ ایجاد پروژه‌ی فصل |
| ۴۱۹ | | ۱۶-۱-۱ آماده کردن نما |
| ۴۲۲ | | ۱۶-۲ بررسی کنترلرها |
| ۴۲۳ | | ۱۶-۲-۱ ایجاد کنترلر |
| ۴۲۳ | | ۱۶-۲-۱-۱ ایجاد کنترلرهای POCO |
| ۴۲۵ | | ۱۶-۲-۱-۲ کاربرد کلاس پایه‌ی Controller |
| ۴۲۶ | | ۱۶-۳ دریافت داده‌های context |
| ۴۲۷ | | ۱۶-۳-۱ استخراج داده‌ها از اشیاء context |
| ۴۲۹ | | ۱۶-۳-۲ داده‌های context در کنترلر POCO |
| ۴۳۲ | | ۱۶-۳-۳ پارامترهای متد اکشن |
| ۴۳۴ | | ۱۶-۴ تولید پاسخ |
| ۴۳۴ | | ۱۶-۴-۱ ایجاد پاسخ با شیء context |
| ۴۳۶ | | ۱۶-۴-۲ کار با Action Result |
| ۴۳۸ | | ۱۶-۴-۳ ایجاد پاسخ HTML |
| ۴۳۹ | | ۱۶-۴-۳-۱ فرآیند جست‌وجوی نما |
| ۴۴۰ | | ۱۶-۴-۳-۲ ارسال داده‌ها از اکشن به نما |
| ۴۴۳ | | ۱۶-۴-۳-۳ استفاده از ViewBag |
| ۴۴۴ | | ۱۶-۴-۴ هدایت مشتری به آدرس مشخص |
| ۴۴۵ | | ۱۶-۴-۴-۱ هدایت صریح |
| ۴۴۶ | | ۱۶-۴-۴-۲ هدایت به آدرسی در سیستم مسیریابی |
| ۴۴۷ | | ۱۶-۴-۴-۳ هدایت مشتری به متد اکشن |
| ۴۴۸ | | ۱۶-۴-۴-۴ الگوی Post/Redirect/Get |
| ۴۴۹ | | ۱۶-۴-۴-۵ استفاده از TempData |
| ۴۵۱ | | ۱۶-۵ انواع محتوای خروجی متد اکشن |
| ۴۵۱ | | ۱۶-۵-۱ ایجاد پاسخ JSON |
| ۴۵۲ | | ۱۶-۵-۲ ایجاد پاسخ توسط اشیاء |

| | |
|------------|--|
| ۴۵۳ | خروجی فایل به عنوان پاسخ |
| ۴۵۵ | ۱۶-۷ خطاها و کدهای HTTP به عنوان پاسخ |
| ۴۵۶ | ۱۶-۷-۱ برگشت دادن کد وضعیت مشخص |
| ۴۵۷ | فصل هفدهم؛ تزریق وابستگی |
| ۴۵۸ | ۱۷-۱ آماده‌سازی پروژه‌ی فصل ۱۷ |
| ۴۵۹ | ۱۷-۱-۱ ایجاد مدل و مخزن داده‌ها |
| ۴۶۱ | ۱۷-۱-۲ ایجاد نما و کنترلر |
| ۴۶۳ | ۱۷-۱-۳ ایجاد پروژه‌ی آزمایش واحد |
| ۴۶۴ | ۱۷-۲ مرتبط کردن اجزای پروژه |
| ۴۶۴ | ۱۷-۲-۱ عناصر وابسته |
| ۴۶۶ | ۱۷-۲-۱-۱ جداسازی عناصر وابسته |
| ۴۶۷ | ۱۷-۲-۱-۲ کاربرد کلاس تایپ بروکر |
| ۴۷۱ | ۱۷-۳ معرفی تزریق وابستگی در ASP.NET |
| ۴۷۱ | ۱۷-۳-۱ آماده‌سازی پروژه برای تزریق وابستگی |
| ۴۷۳ | ۱۷-۳-۲ پیکربندی ارئه دهنده‌ی سرویس |
| ۴۷۵ | ۱۷-۳-۳ آزمایش واحد کنترلر |
| ۴۷۶ | ۱۷-۳-۴ وابستگی زنجیری |
| ۴۷۹ | ۱۷-۳-۵ تزریق وابستگی برای کلاس‌های C# |
| ۴۸۱ | ۱۷-۴ چرخه‌ی عمر سرویس |
| ۴۸۲ | ۱۷-۴-۱ چرخه‌ی عمر گذرا |
| ۴۸۷ | ۱۷-۴-۲ AddScoped() متد |
| ۴۸۸ | ۱۷-۴-۳ AddSingleton() متد |
| ۴۸۹ | ۱۷-۵ وابستگی در متد اکشن |
| ۴۹۰ | ۱۷-۶ تزریق خصوصیت |
| ۴۹۱ | ۱۷-۷ درخواست شیء مورد وابستگی |
| ۴۹۳ | فصل هجدهم؛ فیلترها |
| ۴۹۴ | ۱۸-۱ آماده کردن پروژه‌ی فصل |
| ۴۹۵ | ۱۸-۱-۱ فعال کردن SSL |
| ۴۹۶ | ۱۸-۱-۲ ایجاد کنترلر و نما |
| ۴۹۸ | ۱۸-۲ استفاده از فیلترها |
| ۵۰۱ | ۱۸-۳ فهم کارکرد فیلترها |
| ۵۰۲ | ۱۸-۳-۱ داده‌های Context |
| ۵۰۳ | ۱۸-۴ استفاده از فیلترهای اعتبارسنجی |
| ۵۰۳ | ۱۸-۴-۱ ایجاد فیلتر اعتبارسنجی |
| ۵۰۵ | ۱۸-۵ فیلترهای اکشن |
| ۵۰۶ | ۱۸-۵-۱ ایجاد فیلتر اکشن |
| ۵۰۸ | ۱۸-۵-۲ فیلتر اکشن غیرسنکرون |
| ۵۰۹ | ۱۸-۶ کاربرد فیلتر Result |
| ۵۱۰ | ۱۸-۶-۱ ایجاد فیلتری از نوع Result |

| | | | |
|-----|-------|----------|----------------------------------|
| ۵۱۲ | | ۱۸-۶-۲ | فیلتر Result غیرسنکرون |
| ۵۱۳ | | ۱۸-۶-۳ | فیلترهای ترکیبی |
| ۵۱۶ | | ۱۸-۷ | فیلترهای Exception |
| ۵۱۷ | | ۱۸-۷-۱ | ایجاد فیلتری از نوع Exception |
| ۵۱۹ | | ۱۸-۸ | تزریق وابستگی و فیلترها |
| ۵۱۹ | | ۱۸-۸-۱ | روش مدیریت context |
| ۵۲۴ | | ۱۸-۸-۲ | مدیریت چرخه‌ی عمر فیلتر |
| ۵۲۷ | | ۱۸-۹ | فیلترهای سراسری |
| ۵۲۹ | | ۱۸-۱۰ | ترتیب اجرای فیلترها |
| ۵۳۲ | | ۱۸-۱۰-۱ | تغییر ترتیب اجرای فیلترها |
| ۵۳۳ | | | فصل نوزدهم؛ کنترلرهای API |
| ۵۳۳ | | ۱۹-۱ | ایجاد پروژه‌ی فصل ۱۹ |
| ۵۳۵ | | ۱۹-۱-۱ | ایجاد کنترلر و نما |
| ۵۳۸ | | ۱۹-۱-۲ | پیکربندی پروژه |
| ۵۳۹ | | ۱۹-۱-۲-۱ | تنظیم درگاه HTTP |
| ۵۴۰ | | ۱۹-۲ | نقش کنترلرهای RESTful |
| ۵۴۲ | | ۱۹-۳ | معرفی REST و کنترلرهای API |
| ۵۴۳ | | ۱۹-۳-۱ | ایجاد کنترلر API |
| ۵۴۴ | | ۱۹-۳-۱-۱ | تعریف مسیر |
| ۵۴۵ | | ۱۹-۳-۱-۲ | تعریف وابستگی‌ها |
| ۵۴۵ | | ۱۹-۳-۱-۳ | تعریف متدهای اکشن |
| ۵۴۶ | | ۱۹-۳-۱-۴ | تعریف خروجی متدهای اکشن |
| ۵۴۶ | | ۱۹-۳-۲ | کنترلرهای API در مرورگر |
| ۵۴۹ | | ۱۹-۴ | فرمت محتوا |
| ۵۵۰ | | ۱۹-۴-۱ | سیاست قالب‌گذاری پیش‌فرض |
| ۵۵۱ | | ۱۹-۴-۲ | شناسایی قالب |
| ۵۵۲ | | ۱۹-۴-۲-۱ | فعال کردن قالب XML |
| ۵۵۴ | | ۱۹-۴-۳ | تعیین قالب پاسخ در اکشن |
| ۵۵۵ | | ۱۹-۴-۴ | قالب پاسخ در مسیر و Query String |
| ۵۵۷ | | ۱۹-۴-۵ | گفتگوی محتوا |
| ۵۵۹ | | ۱۹-۴-۶ | دریافت چندین قالب مختلف |
| ۵۶۱ | | | فصل بیستم؛ نماها |
| ۵۶۲ | | ۲۰-۱ | آماده کردن پروژه‌ی فصل |
| ۵۶۴ | | ۲۰-۲ | ایجاد موتور نمای شخصی |
| ۵۶۶ | | ۲۰-۲-۱ | ایجاد نمونه‌ای از IView |
| ۵۶۷ | | ۲۰-۲-۲ | ایجاد نمونه‌ی ViewEngine |
| ۵۶۸ | | ۲۰-۲-۳ | ثبت موتور نمای شخصی |
| ۵۶۹ | | ۲۰-۲-۴ | آزمایش موتور نما |
| ۵۷۱ | | ۲۰-۳ | موتور نمای Razor |

| | | |
|-----|-------|--|
| ۵۷۲ | | ۲۰-۳-۱ ایجاد پروژه |
| ۵۷۴ | | Razor ۲۰-۳-۲ کارکرد نماهای |
| ۵۷۵ | | ۲۰-۳-۲-۱ نام کلاس |
| ۵۷۵ | | ۲۰-۳-۲-۲ آشنایی با کلاس پایه |
| ۵۷۷ | | ۲۰-۳-۲-۳ نمایش نما |
| ۵۷۸ | | Razor ۲۰-۴ محتوای پویای نمای |
| ۵۷۹ | | ۲۰-۴-۱ کاربرد بخش‌ها |
| ۵۸۲ | | ۲۰-۴-۱-۱ آزمایش وجود بخش در نما |
| ۵۸۳ | | ۲۰-۴-۱-۲ نمایش انتخابی بخش‌ها |
| ۵۸۵ | | ۲۰-۴-۲ نماهای جزئی |
| ۵۸۵ | | ۲۰-۴-۲-۱ ایجاد نمای جزئی |
| ۵۸۶ | | ۲۰-۴-۲-۲ استفاده از نمای جزئی |
| ۵۸۷ | | ۲۰-۴-۲-۳ نمای جزئی مقید شده به مدل |
| ۵۸۸ | | ۲۰-۴-۳ محتوای JSON در نماها |
| ۵۹۰ | | Razor ۲۰-۵ پیکربندی |
| ۵۹۳ | | ۲۰-۶ انتخاب نما برای درخواست |
| ۵۹۷ | | فصل بیست و یکم؛ کامپوننت‌های نما |
| ۵۹۷ | | ۲۱-۱ آماده کردن پروژه‌ی فصل |
| ۵۹۹ | | ۲۱-۱-۱ ایجاد مدل و مخزن داده‌ها |
| ۶۰۱ | | ۲۱-۱-۲ ایجاد کنترلر و نماها |
| ۶۰۴ | | ۲۱-۱-۳ پیکربندی پروژه |
| ۶۰۵ | | ۲۱-۲ آشنایی با کامپوننت‌های نما |
| ۶۰۶ | | ۲۱-۳ ایجاد کامپوننت |
| ۶۰۶ | | ۲۱-۳-۱ ایجاد کامپوننت POCO |
| ۶۰۸ | | ۲۱-۳-۲ کلاس پایه‌ی ViewComponent |
| ۶۱۰ | | ۲۱-۳-۳ ViewComponentResult آشنایی با نوع |
| ۶۱۰ | | ۲۱-۳-۳-۱ ایجاد نمای جزئی |
| ۶۱۳ | | ۲۱-۳-۳-۲ خروجی HTML |
| ۶۱۵ | | ۲۱-۳-۴ دریافت داده‌های context |
| ۶۱۸ | | ۲۱-۳-۴-۱ داده‌های context از نمای اصلی |
| ۶۲۱ | | ۲۱-۳-۵ کامپوننت‌های غیرسنکرون |
| ۶۲۳ | | ۲۱-۴ ایجاد فایل‌های ترکیبی کنترلر/کامپوننت |
| ۶۲۴ | | ۲۱-۴-۱ ایجاد نماهای ترکیبی |
| ۶۲۶ | | ۲۱-۴-۲ کاربرد کلاس ترکیبی |
| ۶۲۹ | | فصل بیست و دوم؛ تگ‌های کمکی |
| ۶۲۹ | | ۲۲-۱ آماده‌سازی پروژه‌ی فصل ۲۲ |
| ۶۳۱ | | ۲۲-۱-۱ ایجاد مدل و مخزن داده‌ها |
| ۶۳۲ | | ۲۲-۱-۲ ایجاد نما و کنترلر |
| ۶۳۴ | | ۲۲-۱-۳ پیکربندی پروژه |

| | | |
|-----|-------|--|
| ۶۳۶ | | ۲۲-۲ ایجاد یک تگ کمکی |
| ۶۳۶ | | ۲۲-۲-۱ ایجاد کلاس تگ کمکی |
| ۶۳۷ | | ۲۲-۲-۱-۱ دریافت اطلاعات عنصر HTML |
| ۶۳۸ | | ۲۲-۲-۱-۲ تولید خروجی |
| ۶۳۹ | | ۲۲-۲-۲ ثبت تگ کمکی |
| ۶۳۹ | | ۲۲-۲-۳ کاربرد تگ کمکی |
| ۶۴۰ | | ۲۲-۲-۴ مدیریت ناحیه‌ی کارکرد تگ کمکی |
| ۶۴۱ | | ۲۲-۲-۴-۱ محدود کردن ناحیه‌ی دید تگ کمکی |
| ۶۴۳ | | ۲۲-۲-۴-۲ گسترش ناحیه‌ی کارکرد تگ کمکی |
| ۶۴۵ | | ۲۲-۳ ویژگی‌های پیشرفته تگ‌های کمکی |
| ۶۴۵ | | ۲۲-۳-۱ ایجاد عناصر شخصی HTML |
| ۶۴۷ | | ۲۲-۳-۲ جای‌گذاری تگ کمکی در محل مشخص |
| ۶۵۱ | | ۲۲-۳-۳ دسترسی به اطلاعات درخواست و مدل نما |
| ۶۵۴ | | ۲۲-۳-۴ کار با مدل نما |
| ۶۵۷ | | ۲۲-۳-۵ اشتراک داده‌ها بین تگ‌های کمکی |
| ۶۵۹ | | ۲۲-۳-۶ جلوگیری از نمایش عناصر HTML |
| ۶۶۱ | | فصل بیست و سوم؛ تگ‌های کمکی فرم |
| ۶۶۱ | | ۲۳-۱ آماده‌سازی پروژه‌ی فصل |
| ۶۶۱ | | ۲۳-۱-۱ تغییر وضعیت ثبت تگ‌های کمکی |
| ۶۶۲ | | ۲۳-۱-۲ تغییر نماها و Layout |
| ۶۶۴ | | ۲۳-۲ کار با عناصر فرم |
| ۶۶۴ | | ۲۳-۲-۱ تعیین کنترلر و اکشن هدف |
| ۶۶۵ | | ۲۳-۲-۲ ویژگی anti-forgery |
| ۶۶۷ | | ۲۳-۳ کار با عناصر input |
| ۶۶۸ | | ۲۳-۳-۱ پیکربندی عنصر input |
| ۶۷۰ | | ۲۳-۳-۲ فرمت مقادیر داده‌ها |
| ۶۷۴ | | ۲۳-۴ عنصر label |
| ۶۷۶ | | ۲۳-۵ کار با عناصر select |
| ۶۷۸ | | ۲۳-۵-۱ منبع داده‌های select |
| ۶۷۹ | | ۲۳-۵-۱-۱ مدل به عنوان منبع عناصر option |
| ۶۸۴ | | ۲۳-۶ کار با عنصر TextArea |
| ۶۸۷ | | فصل بیست و چهارم؛ مقیدسازی مدل |
| ۶۸۸ | | ۲۴-۱ آماده‌سازی پروژه‌ی فصل |
| ۶۸۹ | | ۲۴-۱-۱ ایجاد مدل و مخزن داده‌ها |
| ۶۹۰ | | ۲۴-۱-۲ ایجاد کنترلر و نما |
| ۶۹۲ | | ۲۴-۱-۳ پیکربندی پروژه |
| ۶۹۳ | | ۲۴-۲ آشنایی به مقیدسازی مدل |
| ۶۹۵ | | ۲۴-۲-۱ مقادیر پیش‌فرض در مقیدسازی مدل |
| ۶۹۷ | | ۲۴-۲-۲ مقیدسازی انواع ساده |

| | | | |
|-----|-------|----------|---------------------------------------|
| ۶۹۸ | | ۲۴-۲-۳ | مقیدسازی انواع پیچیده |
| ۷۰۴ | | ۲۴-۲-۳-۱ | تعریف پیشوندهای شخصی |
| ۷۰۷ | | ۲۴-۲-۳-۲ | مقیدسازی خصوصیات انتخاب شده |
| ۷۰۹ | | ۲۴-۲-۴ | مقیدسازی آرایه و کلکسیون |
| ۷۱۰ | | ۲۴-۲-۴-۱ | مقیدسازی آرایه‌ها |
| ۷۱۲ | | ۲۴-۲-۴-۲ | مقیدسازی کلکسیون‌ها |
| ۷۱۳ | | ۲۴-۲-۴-۳ | کلکسیونی از انواع پیچیده |
| ۷۱۷ | | ۲۴-۳ | منبعی برای مقیدسازی مدل |
| ۷۱۸ | | ۲۴-۳-۱ | انتخاب منبع داده‌ی استاندارد |
| ۷۱۹ | | ۲۴-۳-۲ | هدر درخواست به عنوان منبع مقیدسازی |
| ۷۲۳ | | ۲۴-۳-۳ | بدنه‌ی درخواست به عنوان منبع مقیدسازی |

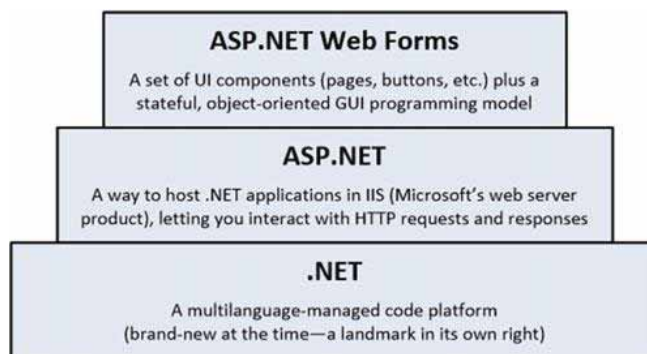
فصل یکم

آشنایی با ASP.NET Core MVC

ASP.NET Core MVC فریم ورک توسعه‌ی برنامه‌های کاربردی^۱ مایکروسافت است که اثربخشی و سازماندهی مناسب معماری مدل-نما-کنترلر^۲ را با بهترین بخش‌های .NET در هم آمیخته است. در این فصل، همراه با فهم چرایی کاربرد ASP.NET Core MVC، مقایسه‌ای خواهیم داشت با دیگر معماری‌های مشابه آن و در پایان، با تازه‌های افزوده شده به این معماری و آنچه که در این کتاب مورد بررسی قرار خواهد گرفت، آشنا خواهید شد.

۱- تاریخچه‌ی ASP.NET Core MVC

در سال ۲۰۰۲ که مایکروسافت سعی بر حفظ موقعیت برتر خود در دنیای پروژه‌های دسکتاپ و ویندوز داشت، متوجه خطر توسعه‌ی نرم‌افزارهای وب شد و در پی آن ASP.NET را به عنوان فریم ورک توسعه‌ی وب، عرضه کرد. شکل ۱-۱، وضعیت فن‌آوری مایکروسافت را در آن زمان نشان می‌دهد.



شکل ۱-۱

۱-۱ پروژه‌های فرم‌های وب

معرفی پروژه‌های فرم‌های وب، روشی بود که مایکروسافت برای دور زدن پروتکل انتقال ابرمتن (HTTP)، به دلیل ناماندگاری^۳ آن، و زبان علامت‌گذاری ابرمتن (HTML) که در آن زمان هنوز برای برنامه‌نویسان مفهوم آشنایی نبود، به وسیله‌ی معرفی صفحه‌ای به نام فرم وب، شامل سلسله‌مراتبی از کنترل‌های

^۱ Application Development Framework

^۲ Model-View-Controller (MVC)

^۳ صفحات منتقل شده در این پروتکل Stateless هستند. یعنی برخلاف فرم‌های ویندوز، وقتی در وب از صفحه‌ای به صفحه‌ی دیگر حرکت می‌کنید، عملاً همه‌ی اطلاعات صفحه‌ی پیشین را از دست می‌دهید، م.

سمت وب^۱، در پیش گرفت. در دریافت درخواست مرورگر و پاسخ سرور، هر کنترل مسئول نگهداری وضعیت (State) خود بود و رویدادهای (Events) سمت مشتری، مانند کلیک بر روی یک دکمه، به صورت خودکار به کد رویدادهایی (Event Handlers) که باید در سمت سرور اجرا می‌شدند، متصل بود. در نتیجه، فرم‌های وب، لایه‌ای غول‌آسا برای انتقال رابط کاربری کلاسیک (GUI شبیه همانی که در ویندوز استفاده می‌شود) بر روی بستر وب هستند.

هدف اصلی این بود که برنامه‌نویسی وب تا جای ممکن شبیه برنامه‌نویسی ویندوز شود. برنامه‌نویسان می‌توانستند بر پایه‌ی یک رابط گرافیکی که به خوبی توانایی حفظ وضعیت خود را در حرکت بین فرم‌ها دارا بود، فکر کنند و نیازی به درگیری با درخواست‌های HTTP و پاسخ‌های آنها، نبود. به این ترتیب، هدف مایکروسافت در انتقال جمعیت عظیم برنامه‌نویسان ویندوز به سمت برنامه‌نویسی وب، در حال عملی شدن بود.

۱-۱-۱ مشکلات پروژه‌های فرم‌های وب

پایه‌های تفکر توسعه‌ی فرم‌های وب اشکالی نداشت، ولی در عمل مشکلاتی در این زمینه پیدا شد:

- حجم وضعیت فرم: روشی که در حفظ وضعیت فرم‌ها در پیش گرفته شده بود، موجب انتقال حجم بزرگی از داده‌ها بین سرور و مشتری می‌شد. حتی در فرم‌های وب معمولی، حجم این داده‌ها می‌توانست به چند صد کیلو بایت برسد. این داده‌ها می‌بایست در هر درخواست و پاسخ، بین سرور و مشتری حرکت کنند که نتیجه، پهنای باند وسیع مورد نیاز بر روی سرور، و کندی زمان پاسخ‌گویی بود.
- چرخه‌ی عمر صفحه: روش ارتباط دادن بین رویدادهای سمت مشتری با کد مدیریت این رویدادها (Event Handlers) به عنوان بخشی از چرخه‌ی عمر صفحه‌ی وب، بر روی سرور، پیچیده بود.
- پیاده نشدن جداسازی دغدغه‌ها^۲: روش کدنویسی فرم‌های وب، ظاهراً می‌خواهد کد سمت سرور را از HTML صفحه جدا کند و آن را در کلاسی جداگانه به نام کد پشتی (Code Behind) قرار دهد. این کار برای جداسازی کد منطق برنامه از کد رابط کاربری انجام شده بود. با این حال، برنامه‌نویسان در عمل وادار به ترکیب کد سمت سرور (مثلاً کد C# یا VB) با کد HTML بودند.
- نبود کنترل بر HTML: کنترل‌های سمت سرور در انتقال به سمت مشتری تبدیل به کد HTML می‌شدند ولی این HTML همانی نبود که برنامه‌نویس انتظار داشت. در روزهای نخستین فرم‌های وب، HTML

^۱ Server side Controls روش نخستین مایکروسافت در ایجاد کنترل‌ها بر روی سرویس‌دهنده و ارسال ترجمه‌ی HTML آنها به سمت مشتری. مایکروسافت در حال حاضر خودش این روش را نادرست معرفی می‌کند و در معماری‌های جدید از آن استفاده نمی‌کند م.

^۲ Separation of Concerns

تولید شده با استانداردهای وب همخوانی نداشت و به سختی می‌توانست با CSS شکل‌دهی شود. افزون بر این، IDهای ایجاد شده در کد HTML با جاوا اسکریپت قابل دستیابی نبودند.

- آزمایش ناپذیری: طراحان فرم‌های وب نمی‌توانستند پیش‌بینی کنند که زمانی آزمایش کد، بخش مهمی از توسعه‌ی هر نرم‌افزاری می‌شود. معماری ارائه شده توسط آنها در فرم‌های وب، به هیچ روی مناسب روش‌های امروزی آزمایش کد نیست.

باید یادآور شد که توسعه‌ی فرم‌های وب چندان هم بد نبود. میکروسافت تا جای ممکن برای هماهنگی با استانداردهای وب و آسان کردن برنامه‌نویسی آن، کوشش کرد و حتی بعدها از برخی از ویژگی‌های نخستین ASP.NET MVC در معماری فرم‌های وب هم استفاده کرد. هنوز هم اگر نیاز سریع به برنامه‌ی وبی داشته باشید و به گفته‌ی برنامه‌نویسان بخواهید در کمترین زمان سایت را بالا بیاورید، فرم‌های وب حرف اول را می‌زنند.

۱-۲ پروژه‌های MVC قدیمی

در اکتبر سال ۲۰۰۷، میکروسافت، برای پاسخ‌گویی به انتقادهای فرم‌های وب و محبوب بودن برخی روش‌های توسعه‌ی جدید مانند Ruby، پلتفرم توسعه‌ی^۱ جدیدی بر پایه‌ی ASP.NET را اعلام کرد. پلتفرم جدید، ASP.NET MVC نام داشت و هدف آن، برآورده کردن استانداردهای نوین توسعه‌ی وب، مانند CSS و HTML، سرویس‌های REST، آزمایش‌پذیری کد و تسلیم بر این باور بود که برنامه‌نویسان باید بدون وضعیت (Stateless) بودن ذاتی HTTP را بپذیرند.

معرفی MVC، تغییر دیگری را هم در سیاست‌های پیشین میکروسافت موجب شد. پیش از آن، میکروسافت تأکید زیادی بر کنترل اجزای ایجاد‌کننده‌ی نرم‌افزار داشت. ولی با معرفی MVC، و کاربرد ابزار متن‌بازی (Open Source) مانند Queryزدر آن، میکروسافت سعی کرد از بهترین‌های رقبای خود استفاده کند و از همین روی، متن کد اصلی MVC را در اختیار برنامه‌نویسان به عنوان کاربران اصلی MVC، قرار داد.

۱-۲-۱ مشکل پروژه‌های قدیمی MVC

در زمان ایجاد MVC برای میکروسافت طبیعی بود که نرم‌افزار جدید را بر پایه‌ی ASP.NET بر پا کند. تغییرات زیادی لازم بود تا MVC بتواند بر پایه‌ی نرم‌افزاری که برای فرم‌های وب طراحی شده بود، کار کند. بنابراین طراحان اولیه‌ی MVC مجبور به تغییر تنظیمات و پیکربندی‌هایی شدند که ربطی به اجرای کد MVC نداشتند ولی به هر حال برای این که همه چیز به درستی کار کند، لازم بودند.

^۱ Development Platform

با محبوب شدن MVC، میکروسافت شروع به استفاده از برخی ویژگی‌های پایه‌ی آن در فرم‌های وب کرد. در نتیجه، کدی که به خودی خود دارای ویژگی‌های عجیبی بود (زیرا بر پایه‌ی ASP.NET و برای MVC طراحی شده بود) در کاربرد برای توسعه‌ی فرم‌های وب، با تلاش بر این که، همه چیز به هر حال باید کار کند، منجر به کدی پیچیده‌تر و عجیب‌تر شد. در همان زمان، میکروسافت شروع به گسترش ASP.NET با ویژگی‌های جدیدی مانند سرویس‌ها و وب (Web Services) کرد که هر یک نیاز به پیکربندی و تنظیمات مخصوص به خود داشت. نتیجه‌ی پایانی، کدی پاره پاره و بدفرم بود.

۱-۳ فهم ASP.NET Core

در سال ۲۰۱۵ میکروسافت خیر از مسیرهای جدیدی برای ASP.NET و MVC داد، که منجر به ASP.NET Core MVC، یعنی موضوع مورد بررسی این کتاب شد.

ASP.NET Core بر پایه‌ی NET Core استوار است که نگارشی از NET، مستقل از سیستم عامل و بدون اینترفیس برنامه‌نویسی ویندوز است. ویندوز هنوز هم سیستم عاملی برتر به حساب می‌آید؛ ولی برنامه‌های وب نه تنها روز به روز از کاربرد و محبوبیت بیشتری برخوردار می‌شوند، بلکه بر روی پلتفرم‌های دیگری مانند فضای ابری (Cloud) هم باید بتوانند میزبانی (Host) شوند. میکروسافت با این کار، گسترده‌ی کارکرد NET را افزایش داده است؛ به این معنی که می‌توان برنامه‌های کاربردی ASP.NET Core را بر روی بازه‌ی گسترده‌ای از محیط‌های مختلف میزبانی کرد (عدم وابستگی به IIS، م، هم اینک، می‌توانید پروژه‌های وب ASP.NET Core را برای Linux یا macOS هم تولید کنید.

ASP.NET Core، در مقایسه با MVC نخستین، ساده‌تر است و برخلاف آن، هیچ ارتباطی با پروژه‌های فرم‌های وب ندارد و از آنجا که بر پایه‌ی NET Core بنا شده است، ایجا پروژه‌های وب را بر روی بسیاری از سیستم‌عامل‌های گوناگون پشتیبانی کرده و امکان میزبانی در محیط‌های مختلف را داراست.

ASP.NET Core همه‌ی امکانات ASP.NET MVC را بر پایه‌ی پلتفرم جدید ASP.NET Core، فراهم می‌کند. افزون بر این که شامل همه‌ی کارآیی‌های اینترفیس کاربری وب است، روش‌های طبیعی‌تری برای ایجاد محتوای پیچیده پیشنهاد می‌کند و امکان می‌دهد که بسیاری از کارهای کلیدی مربوط به توسعه، مانند آزمایش‌های واحد، به سادگی انجام شوند.

۱-۳-۱ مزایای اصلی ASP.NET Core

در بخش‌هایی که در ادامه آورده می‌شوند، نشان خواهیم داد که چگونه پلتفرم جدید MVC، بر پروژه‌های قدیمی فرم‌های وب و MVC نخستین، برتری یافته است.

۱-۳-۱-۱ معماری MVC

ASP.NET Core MVC از الگوی مدل-نما-کنترلر برای ایجاد پروژه‌های وب استفاده می‌کند. در اینجا باید بین الگوی معماری MVC و پیاده‌سازی ASP.NET Core MVC از این الگو، تفاوت قائل شویم. الگوی MVC جدید نیست و به سال ۱۹۷۸ و پروژه‌ی Smalltalk بازمی‌گردد و از آن زمان تا کنون، به دلایلی که در ادامه آورده می‌شوند، در ایجاد پروژه‌های وب محبوبیت زیادی پیدا کرده است.

- برخورد کاربر با برنامه‌ای کاربردی که از الگوی MVC استفاده کرده است، مسیری طبیعی را طی می‌کند؛ کاربر عملی را انجام می‌دهد و در پاسخ به آن، برنامه با تغییر مدل داده‌ها، نمای جدیدی را نمایش می‌دهد. این چرخه به همین صورت ادامه پیدا می‌کند. این روش برای برنامه‌های وب، به عنوان مجموعه‌ای از درخواست‌ها و پاسخ‌های HTTP، مناسب است.
- در برنامه‌های کاربردی وب با فن‌آوری‌های گوناگونی مانند پایگاه‌های داده، HTML و کد اجرایی، سروکار پیدا می‌کنیم که در عمل، به صورت لایه‌های جدای پروژه پیاده‌سازی می‌شوند. الگوی به دست آمده از این ترکیب با تفکری که MVC بر پایه‌ی آن ایجاد شده، هماهنگی دارد.

از آنجا که ASP.NET Core MVC الگوی MVC را پیاده می‌کند، در مقایسه با پروژه‌های قدیمی فرم‌های وب، موضوع جداسازی لایه‌های پروژه را به خوبی انجام می‌دهد. در فصل سوم، این معماری را با جزئیات بیشتری بررسی خواهیم کرد.

۱-۳-۱-۲ گسترش پذیری

ASP.NET Core MVC و ASP.NET Core MVC شامل عناصر (Component) مستقلی هستند. این عناصر دارای مشخصه‌های روشنی بوده و معمولاً از یک اینترفیس (Interface) و یا یک کلاس مجرد (Abstract Class) ارث‌بری کرده‌اند. به سادگی می‌توانید هر یک از این عناصر را با آنچه که خودتان پیاده‌سازی کرده‌اید، جایگزین کنید. برای هر عنصر، سه انتخاب در پیش رو دارید:

- پیاده‌سازی پیش‌فرض آن را، همان‌گونه که هست به کار برید که برای بسیاری از برنامه‌ها کافی است.
- کلاس دیگری از پیاده‌سازی پیش‌فرض مشتق کنید و رفتار آن را به دلخواه تغییر دهید.
- عنصر مورد نظر را، با ایجاد کلاس جدیدی که از اینترفیس یا کلاس مجرد نخستین ارث‌بری می‌کند، کاملاً جایگزین کنید.

در فصل ۱۳ با این روش‌ها آشنا خواهید شد.

۱-۳-۱-۳ کنترل کامل بر HTML و HTTP

خروجی HTML که ASP.NET Core MVC ایجاد می‌کند، کاملاً استاندارد است. برای شکل دادن به این HTML می‌توانید از سبک‌های CSS استفاده کنید. افزون بر این، می‌توانید از jQuery، Angular یا Bootstrap برای ایجاد عناصر پیچیده‌ی نما، مانند تقویم یا منوهای تو در تو استفاده کنید.

ASP.NET Core MVC با HTTP هماهنگ است. به این معنی که کنترل درخواست‌های ارسال شده از مرورگر به سرور را در دست دارید. این، امکان می‌دهد که تجربه‌ی کاربر از برنامه را آن‌گونه که می‌خواهید شکل دهید. کاربرد Ajax ساده شده است و همان‌طور که در فصل ۱۹ آمده است، به راحتی می‌توانید از سرویس‌های وب (Web Service) برای دریافت درخواست‌های ارسالی از مرورگر استفاده کنید.

۱-۳-۱-۴ آزمایش‌پذیری

از آنجا که در معماری ASP.NET Core MVC رابط کاربری، مدل داده‌ها و کد پردازش‌کننده به خوبی از هم جدا شده‌اند، زمینه‌ی خوبی برای اجرای آزمایش‌های واحد (Unit Tests) فراهم می‌شود. این کار را می‌توانید با هر یک از نرم‌افزارهای آزمایش‌متن‌باز موجود، مانند xUnit.net که در فصل ۷ معرفی خواهد شد، انجام دهید.

در این کتاب خواهید دید که چگونه می‌توان با استفاده از روش‌های شبیه‌سازی گوناگون برای ایجاد پیاده‌سازی‌های ساختگی^۱ از عناصر پروژه، کدهای آزمایشی ساده و روشنی برای کنترلرها و متدهای اکشن، نوشت.

آزمایش‌پذیری تنها به ایجاد آزمایش‌های واحد مربوط نمی‌شود. برنامه‌های کاربردی ASP.NET Core MVC با ابزاری که برای آزمایش خودکار رابط کاربری (UI) به کار می‌روند، به خوبی کار می‌کنند. بدون نیاز به دانستن ساختار عناصر HTML، CSS و یا IDهایی که برنامه ایجاد می‌کند، می‌توانید کدی بنویسید که برخورد کاربر با برنامه را شبیه‌سازی کند.

۱-۳-۱-۵ روش مسیریابی قوی

روش استفاده از URLها، با تکامل فن‌آوری وب، تغییر کرده است. آدرس‌هایی مانند:

/App_v2/User/Page.aspx?action=show%20prop&prop_id=82742

به ندرت پیدا می‌شوند و به جای آنها از آدرس‌های مشخص‌تری مانند:

/to-rent/chicago/2303-silver-street

استفاده می‌شود.

^۱ Fake Implementations

چگونگی ساختار URL به دلایل زیادی اهمیت دارد. نخست این که موتورهای جست‌وجو به واژه‌های کلیدی موجود در آدرس‌های اینترنتی اهمیت می‌دهند. دیگر این که، بسیاری از کاربران اکنون معنی آدرس‌های اینترنتی را می‌دانند و ترجیح می‌دهند خودشان آن را در نوار آدرس مرورگر وارد کنند. افزون بر این، زمانی که شخصی معنی یک آدرس اینترنتی را به روشنی بفهمد، تمایل بیشتری برای سهیم شدن آن با دیگران و یا کاربرد لینک آن در صفحه‌ی وب خود خواهد داشت. مهم‌تر از همه‌ی آنها این است که کاربرد چنین URL‌هایی ساختار پوشه‌ها و فایل‌های پروژه را برای دیگران آشکار نمی‌کند و در صورت تغییر پیاده‌سازی ساختار برنامه، در مورد شکسته شدن آدرس‌ها و یا لزوم تغییر آنها، نگرانی نخواهید داشت.

استفاده از URL‌های روشن و قابل فهم، مانند آنچه که در بالا آمد، در فریم ورک‌های وب قدیمی مشکل بود. ولی ASP.NET Core MVC از روش آدرس‌دهی پیشرفته‌ای به نام مسیره‌ی URL (URL Routing) استفاده می‌کند، که به طور پیش‌فرض چنین URL‌هایی را به کار می‌برد. این امکان می‌دهد که کنترل خوبی بر ساختار آدرس‌های صفحات خود داشته باشید، و URL‌هایی با معنی و روشن در اختیار کاربران خود قرار دهید.

۶-۱-۳-۱ رابط برنامه‌نویسی قوی

از آنجا که ASP.NET Core MVC بر پایه‌ی NET Core بنا شده، از بسیاری از ویژگی‌های قدرت‌مند آن که برای برنامه‌نویسان C# آشنا هستند مانند، کاربرد await، متدهای توسعه یافته^۱، عبارت‌ها لاند^۲، انواع پویا و بی‌نام^۳ و کوئری آمیخته با زبان^۴ (LINQ) استفاده می‌کند.

۷-۱-۳-۱ چند پلتفرمی^۵

نگارش‌های گذشته‌ی ASP.NET، هم برای نوشتن برنامه نیازمند ویندوز بودند و هم برای میزبانی به سرور ویندوز نیاز داشتند. ASP.NET Core، هم برای برنامه‌نویسی و توسعه و هم برای انتشار به محیط ویژه‌ای وابسته نیست.

^۱ Extension Methods

^۲ Lamda Expressions

^۳ Anonymous and Dynamic Types

^۴ Language Integrated Query (LINQ)

^۵ Cross Platform

۱-۳-۱-۷ متن باز بودن

بر خلاف فریم ورک‌های گذشته‌ی توسعه‌ی میکروسافت، اکنون می‌توانید کد سورس ASP.NET Core MVC را دانلود کنید و حتی پس از انجام تغییرات و کامپایل، نگارش خودتان از آنها را به کار برید. هر دوی این نرم‌افزارها از آدرس <https://github.com/aspnet> قابل دانلود هستند.

۱-۴ نیازمندی‌ها

برای این که بهترین استفاده را از این کتاب ببرید، باید افزون بر آشنایی با مفاهیم پایه‌ی برنامه‌نویسی و توسعه‌ی وب، HTML و CSS، دانشی از برنامه‌نویسی با زبان C# داشته باشید.

اگر با برنامه‌نویسی سمت مشتری مانند جاوا اسکریپت به خوبی آشنایی ندارید، نگران نباشید. در این کتاب بر برنامه‌نویسی سمت سرور تمرکز خواهیم داشت و آنچه که در مورد جاوا اسکریپت لازم است را از کد مثال‌ها خواهید آموخت. در فصل چهارم، بررسی سریعی بر مهم‌ترین و جدیدترین ویژگی‌های C# خواهیم داشت. اگر تنها با برنامه‌نویسی سنتی C# کار کرده‌اید، توصیه می‌شود این فصل را به دقت بخوانید.

۱-۵ ساختار کتاب

این کتاب به دو بخش اصلی تقسیم می‌شود و هر بخش مطالب مرتبطی را مورد بررسی قرار می‌دهد.

بخش نخست در مورد آشنایی با ASP.NET Core MVC، ویژگی‌های کاربردی الگوی سه بخشی MVC را مورد بررسی قرار می‌دهد و همراه با ارائه‌ی روش‌های امروزی توسعه‌ی وب، ابزاری که C# برای رسیدن به این هدف در اختیار قرار می‌دهد را بررسی می‌کند.

در فصل دوم، همراه با ایجاد پروژه‌ای ساده، با مفاهیم پایه‌ی MVC آشنا خواهید شد. ادامه‌ی بخش نخست کتاب به پیاده‌سازی پروژه‌ای واقعی به نام SportsStore اختصاص پیدا کرده است که سعی بر پیاده‌سازی ویژگی‌های مهم ASP.NET Core MVC خواهد داشت.

در بخش دوم کتاب، که بخش پیشرفته‌ی آن را تشکیل می‌دهد، به بررسی زیرساخت‌ها و کارکردهای درونی ویژگی‌هایی که از آنها در پروژه‌ی SportsStore استفاده شده است، خواهیم پرداخت. همراه با شرح چگونگی کارکرد هر یک از این ویژگی‌ها، پیکربندی‌های مختلف نیز بررسی می‌شوند.

فصل دوم

ایجاد نخستین پروژه MVC

بهترین راه برای درک یک فریم‌ورک توسعه‌ی نرم‌افزار، تجربه‌ی استفاده از آنست. در این فصل، یک پروژه‌ی ساده‌ی ورود داده با استفاده از MVC ایجاد خواهیم کرد. برای این که درک خوبی از چگونگی ایجاد و عملکرد چنین پروژه‌ای داشته باشید، کار را به صورت گام به گام پیش خواهیم برد. همچنین برای ساده نگاه داشتن کار، در این مرحله، از تشریح برخی جزئیات، پرهیز خواهیم کرد.

۲-۱ نصب ویژوال استدیو

در این کتاب از نگارش ۲۰۱۵ ویژوال استدیو استفاده می‌کنیم که همه‌ی امکانات لازم برای استفاده از ASP.NET MVC را در اختیارمان قرار می‌دهد. نسخه مجانی نرم‌افزار را می‌توانید از آدرس www.visualstudio.com دانلود کنید. در هنگام نصب ویژوال استدیو حتما گزینه‌ی Web developer Tools را انتخاب کرده باشید.

اگر نسخه‌ی نصب شده‌ای از ویژوال استدیو در اختیار دارید، به احتمال قوی باید به‌روزرسانی ۳ (Update 3) را انجام دهید. در صورتی که چنین است، برای انجام این به‌روزرسانی به آدرس زیر مراجعه کنید:

<http://go.microsoft.com/fwlink/?LinkId=691129>

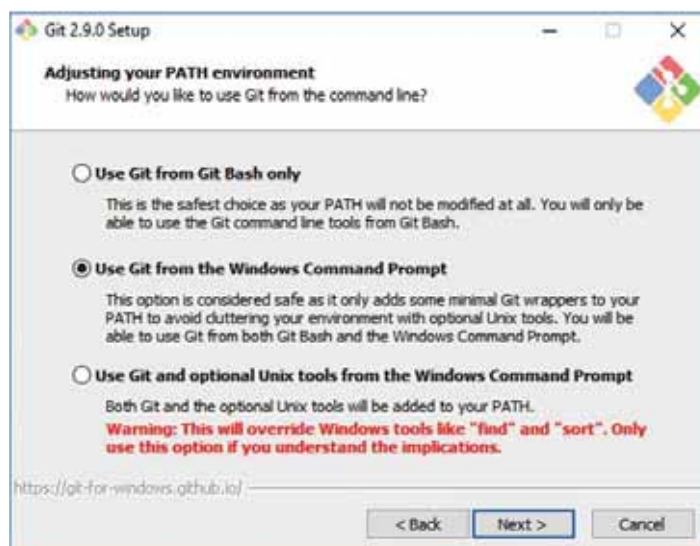
در مرحله‌ی بعد، برای نصب .NET Core باید به آدرس زیر مراجعه کنید:

<https://go.microsoft.com/fwlink/?LinkId=817245>

انجام این کار حتی برای نصب‌های جدید ویژوال استدیو هم الزامی است. آخرین گام، نصب ابزاری به نام git است که از آدرس زیر قابل دستیابی است:

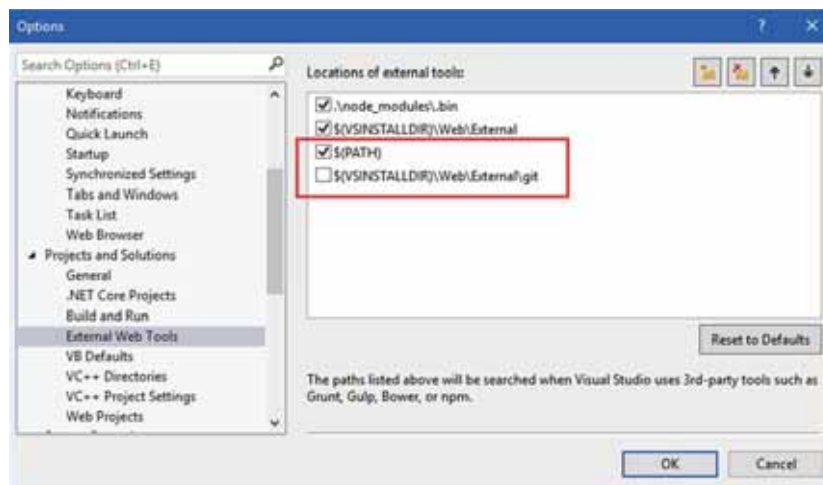
<https://git-scm.com/download>

ویژوال استدیو نگارشی از git را نصب می‌کند که متاسفانه در هنگام استفاده با ابزار دیگر، از جمله Bower که در فصل ۶ به آن خواهیم پرداخت، نتایج نادرستی ایجاد می‌کند. هنگام نصب git، همان‌طور که در شکل ۲-۱ نشان داده شده است، دقت کنید که ابزار مورد گفتگو حتما به متغیر محیطی PATH اضافه شود. در این صورت ویژوال استدیو به راحتی قادر به یافتن آن خواهد بود.



شکل ۲-۱

ویژوال استدیو را اجرا کنید و پس از انتخاب گزینه‌ی Options از فهرست Tools، همان‌گونه که در شکل ۲-۲ نشان داده شده است، با انتخاب External Web tools از بخش Projects and Solutions، گزینه‌ی `$(VSINSTALLDIR)\Web\External\git` item را از حالت انتخاب خارج کنید. به همین صورت مطمئن شوید که گزینه‌ی `$(PATH)` فعال است. با انجام کارهای بالا می‌توانید مطمئن باشید که نسخه git نصب شده توسط شما، به جای نسخه‌ی پیش‌فرض ویژوال استدیو، اجرا خواهد شد.



شکل ۲-۲

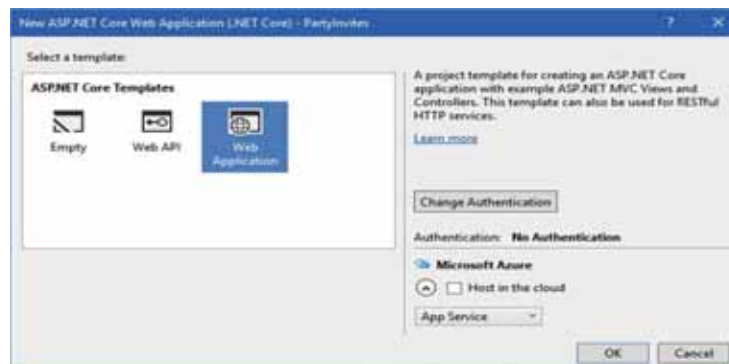
۲-۲ ایجاد پروژه جدید MVC

کار را با ایجاد پروژه‌ی جدیدی از نوع ASP.NET Core MVC در محیط ویژوال استدیو شروع می‌کنیم (شکل ۲-۳). اکنون از فهرست File گزینه‌ی New Project را انتخاب کنید. در این وضعیت، کادر گفتگوی پروژه‌ی جدید، گشوده می‌شود. در صورتی که از پنل سمت چپ، ابتدا Templates و پس از آن، Visual C# را انتخاب کنید، در بخش Web، نوع پروژه‌ی مورد نظرمان، یعنی ASP.NET Core Web Application را خواهید دید.



شکل ۲-۳

نام پروژه را PartyInvites گذاشته و همان‌گونه که در شکل ۲-۳ نشان داده شده است، توجه کنید که گزینه‌ی Add Application Insights to Project انتخاب نشده باشد. پس از کلیک بر دکمه‌ی OK، همان‌گونه که در شکل ۲-۴ نشان داده شده است کادر گفتگو، در مورد محتویات پروژه پرسش خواهد کرد.



شکل ۲-۴

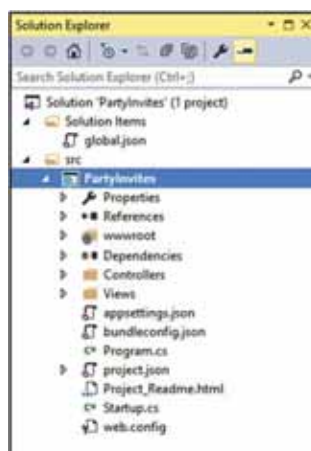
هر یک از سه نوع الگوی پروژه‌ی نشان داده شده در شکل، پروژه‌ای با محتوای آغازین متفاوتی را ایجاد می‌کند. برای کار این فصل، الگوی Web Application را انتخاب کنید.

همان‌گونه که در شکل ۲-۵ نشان داده شده است، با کلیک بر روی دکمه‌ی Change Authentication و اطمینان از اینکه گزینه‌ی No Authenticaion انتخاب شده است، پروژه را ایجاد کنید. در این پروژه نیازی به هیچ نوع خاصی از اعتبارسنجی نخواهیم داشت.



شکل ۲-۵

پس از بستن کادر بالا، در بازگشت به کادر پیشین، مطمئن شوید که گزینه‌ی Host in Cloud انتخاب نشده باشد. با کلیک بر روی OK پروژه‌ی PartyInvites را ایجاد کنید. پس از ایجاد پروژه و باز شدن محیط آن در ویژوال استودیو، فایل‌ها و پوشه‌های مختلفی را در مرورگر سالوشن (Solution Explorer) خواهید دید (شکل ۲-۶).



شکل ۲-۶

برای اجرای پروژه (با محتوای پیش‌فرضی که ویژوال استودیو در آن ایجاد کرده است)، از فهرست Debug گزینه‌ی Start debugging را انتخاب کنید (اگر پیامی در مورد فعال کردن دیباگ برنامه نمایان شد، بر روی OK کلیک کنید). با انجام این کار، ویژوال استودیو پس از کامپایل پروژه، برنامه‌ای به نام IIS Express را برای میزبانی آن، به کار می‌برد. پس از این کار، با باز کردن یکی از مرورگرهای اینترنت نصب شده بر

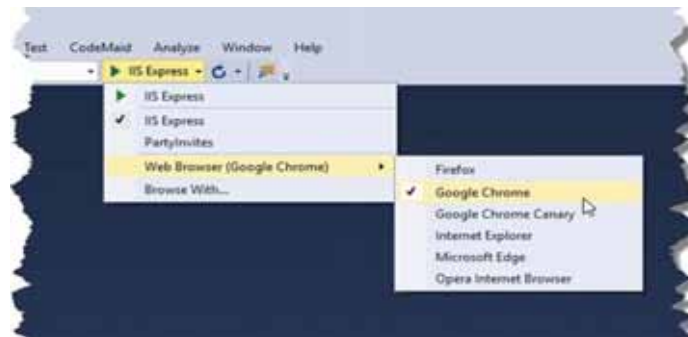
روی کامپیوتر، تقاضایی برای نمایش محتوای برنامه به IIS Express (برنامه‌ی میزبان) ارسال می‌کند (شکل ۲-۷).



شکل ۲-۷

در ادامه‌ی این فصل، برای ایجاد برنامه‌ی مورد نظرمان، محتوای پیش‌فرض ویژوال استدیو برای پروژه را، به آرامی با کدی مناسب، جایگزین خواهیم کرد. هم اینک برای پایان دادن به اجرای پروژه، یا پنجره‌ی مرورگر را ببندید و یا در محیط ویژوال، از فهرست Debug گزینه‌ی Stop debugging را انتخاب کنید.

همان‌گونه که در شکل ۲-۸ می‌بینید، با استفاده از دکمه‌ی IIS Express در نوار ابزار، می‌توانید هر یک از مرورگرهای نصب شده در کامپیوتر را برای اجرای پروژه به ویژوال استدیو معرفی کنید.



شکل ۲-۸

۲-۲-۱ افزودن کنترلر به پروژه

در مدل برنامه‌نویسی MVC تقاضاهای رسیده (به سایت یا وب اپلیکیشن) توسط کنترلرها بررسی و پاسخ داده می‌شوند. در مدل پیاده شده توسط مایکروسافت، ASP.NET Core MVC، کنترلرها کلاس‌های معمولی C# هستند که از `Microsoft.AspNetCore.Mvc.Controller` که کلاس پایه‌ی کنترلر محسوب می‌شود، ارث‌بری کرده‌اند.

هر متدی با دسترسی عمومی (Public) در کنترلر، به عنوان یک متد اکشن (Action Method) شناخته می‌شود. این به معنی آن است که چنین متدی را می‌توانید از داخل یک صفحه وب، توسط یک URL فراخوانی کنید تا عمل (Action) ویژه‌ای را انجام دهد. بر اساس آنچه که در MVC مرسوم است، ویژوال استدیو کنترلرها را در پوشه‌ای به نام `Controllers` قرار می‌دهد. همان پوشه‌ای که در هنگام ایجاد پروژه به طور خودکار توسط ویژوال استدیو ایجاد شد.

با باز کردن پوشه‌ی `Controllers` در مرورگر سالوشن متوجه می‌شوید که ویژوال استدیو کنترلری پیش‌فرض به پروژه اضافه کرده است. نام فایل کنترلر `HomeController` است. فایل کلاس کنترلر همیشه دارای یک پیشوند (در اینجا، `Home`) و سپس واژه‌ی `Controller` است. با مشاهده‌ی چنین نامی متوجه می‌شوید که فایل مورد نظر باید دارای کنترلری به نام `Home` باشد (کنترلر پیش‌فرض در پروژه‌های MVC ویژوال استدیو). برای باز کردن این فایل، دو بار بر روی آن کلیک کنید. ویژوال استدیو این فایل را با کدی که در لیست ۲-۱ نمایش داده شده است، برای ویرایش باز می‌کند.

لیست ۲-۱: کد فایل کنترلر پیش‌فرض ویژوال استدیو

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace PartyInvites.Controllers {
    public class HomeController : Controller {
        public IActionResult Index() {
            return View();
        }
        public IActionResult About() {
            ViewData["Message"] = "Your application description page.";
            return View();
        }
        public IActionResult Contact() {
            ViewData["Message"] = "Your contact page.";
            return View();
        }
    }
}
```



```

    }
    public IActionResult Error() {
        return View();
    }
}
}

```

کد بالا را با کدی که در لیست ۲-۲ آمده است، جایگزین کنید. در اینجا، عبارت‌های `using` بدون استفاده و همه متدها به غیر از یکی از آنها حذف شده‌اند. همان‌گونه که می‌بینید، پیاده‌سازی متد `Index()` و نوع برگشتی آن هم تغییر کرده‌اند.

لیست ۲-۲: کد تغییر یافته کنترلر `HomeController`

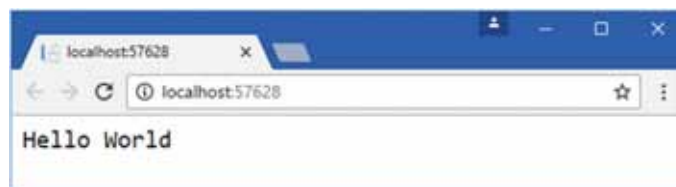
```

using Microsoft.AspNetCore.Mvc;
namespace PartyInvites.Controllers {
    public class HomeController : Controller {
        public string Index() {
            return "Hello World";
        }
    }
}

```

متد `Index()` رشته‌ای به شکل `Hello World` از نوع `string` بازمی‌گرداند. دوباره پروژه راه، به همان روش پیشین اجرا کنید.

مرورگر یک تقاضای^۱ `HTTP` به سرور ارسال می‌کند. بر اساس پیکربندی پیش‌فرض `MVC` این تقاضا با متد `Index()` پاسخ داده می‌شود. این متد را `action method` یا به اختصار، `action` می‌گویند و خروجی این متد در پاسخ به تقاضای یاد شده، برای مرورگر فرستاده می‌شود (شکل ۲-۹).



شکل ۲-۹

در شکل بالا، ویژگی‌های مرورگر را به پورت ۵۷۶۲۸ هدایت کرده است. از آنجا که ویژگی‌های مرورگر همیشه این پورت را به شکل تصادفی انتخاب می‌کند، شماره‌ی آن در کامپیوتر شما می‌تواند متفاوت باشد. افزون بر این با دقت در نوار وظیفه، بخش اعلانات، متوجه آیکن کوچکی در رابطه با `IIS Express`

^۱ HTTP Request

می‌شوید. این برنامه، به عنوان سرویس دهنده‌ی وب، نگارش سبک شده‌ای از برنامه اصلی IIS نصب شده در ویندوز است.

۲-۲-۲ بررسی و فهم مسیرها

افزون بر مدل‌ها^۱، نماها^۲ و کنترلرها^۳، برنامه‌های کاربردی MVC از روش آدرس‌دهی ASP.NET برای مربوط کردن URLها به کنترلرها و منتهای آنها (اکشن‌ها) استفاده می‌کنند. مسیر^۴ قانونی است که معین می‌کند یک درخواست چگونه پاسخ داده شود. ویژگی‌های استدیو در هنگام ایجاد پروژه، چند مسیر پیش‌فرض برای شروع کار، تولید می‌کند. هر یک از URLهای زیر که اکشنی در یک کنترلر اجرا می‌کنند، را می‌توانید به کار برید.

- /
- /Home
- /Home/Index

بنابراین هر بار مرورگری یکی از آدرس‌های `http://yoursite/` یا `http://yoursite/Home` را تقاضا کند، خروجی متد اکشن `Index()` را دریافت خواهد کرد. این موضوع را می‌توانید با تغییر آدرس موجود در مرورگر آزمایش کنید. هم اینک این آدرس به صورت `http://localhost:57628` ولی شاید با پورتهای متفاوت است. اگر رشته‌های `/Home` و یا `/Home/Index` را به دنباله‌ی آن اضافه کنید، همان نتیجه‌ی گذشته، `Hello World` را دریافت خواهید کرد.

وضعیت بالا نشان‌دهنده‌ی مثال خوبی از پذیرفتن پیکربندی پیش‌فرض ASP.NET Core MVC است. تنظیم پیش‌فرض در اینجا این است که، پروژه دارای کنترلری به نام `Home` (با نام فایل `HomeController`) بوده و این کنترلر نقطه‌ی شروع برنامه است (با اکشن `Index`). اگر این پیکربندی پیش‌فرض را دنبال نکرده بودیم، می‌بایست تنظیمات آدرس‌دهی را بر اساس کنترلری که ایجاد کرده بودیم (متفاوت از `Home`) تغییر می‌دادیم.

۲-۳ پردازش و نمایش صفحات وب

خروجی مثال گذشته به جای این که HTML باشد، رشته‌ی `Hello World` بود. برای این که در پاسخ درخواست مرورگر، پاسخ HTML تولید کنیم، باید نمایی داشته باشیم که برنامه را برای تولید چنین خروجی به درستی هدایت کند.

¹ Models

² Views

³ Controllers

⁴ Route

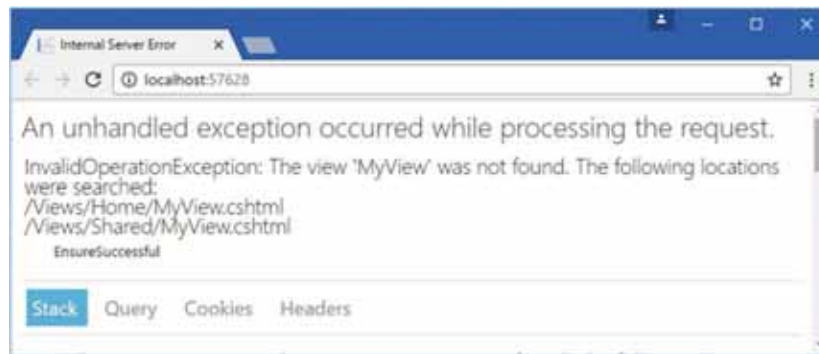
۲-۳-۱ ایجاد نما

نخستین گام، ویرایش اکشن Index به شکل نشان داده شده در لیست ۲-۳ است.

لیست ۲-۳: ویرایش کنترلر برای نمایش صفحه وب

```
using Microsoft.AspNetCore.Mvc;
namespace PartyInvites.Controllers {
    public class HomeController : Controller {
        public IActionResult Index() {
            return View("MyView");
        }
    }
}
```

متد اکشنی که خروجی آن شیئی از نوع `ViewResult` باشد، موجب پردازش یک نما می‌شود. در اینجا شیء `ViewResult` با فرخوانی متد `View()` و ارائه‌ی نام نما به صورت پارامتر متد `(MyView)`، تولید شده است. با اجرای برنامه، خواهید دید که MVC سعی بر یافتن این نما (که البته وجود ندارد) کرده و در نتیجه با پیام خطایی همانند شکل ۲-۱۰ برخورد خواهید کرد.



شکل ۲-۱۰

پیام خطای بالا نه تنها نشان دهنده‌ی این است که برنامه به دنبال یافتن نما بوده است، بلکه مشخص کننده‌ی محل‌های جست‌وجو نیز هست. نماها در پوشه‌ای به نام `Views` که دارای زیرشاخه‌هایی (پوشه فرعی سطوح پایین‌تر) است، ذخیره و دسته‌بندی می‌شوند. نام این پوشه‌ها همیشه با نام کنترلر یکی است. نماهایی که به کنترلر `Home` مربوط می‌شوند، در پوشه‌ای به همین نام ذخیره می‌شوند (`Views/Home`). توجه داشته باشید که هر متد اکشن داخل کنترلر، دارای یک نما در پوشه‌ی یاد شده خواهد بود. نماهایی که به کنترلر خاصی مربوط نیستند، در پوشه‌ای به آدرس `Views/Shared` ذخیره می‌شوند.