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 : فیپا
یادداشت : عنوان اصلی:.Pro ASP.NET Core MVC, 6th ed, 2016
یادداشت : 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 دات‌نت 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
رده بندی کنگره : ۷۶/۷۶QAالف۱۳۹۶ ۴ف۳۳م/
رده بندی دیویی : ۰۰۵/۲۷۶
شماره کتابشناسی ملی : ۴۹۹۱۴۷۳

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، 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 : ۳-۵۳-۸۲۰۱-۶۰۰-۹۷۸**

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 مایکروسافت، 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 دردسرهای 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 مستقیما به آدرس پست الکترونیکی خودم، nanavijobmail@yahoo.com در میان بگذارند.

نادر نبوی

پاییز سال ۱۳۹۶

فهرست

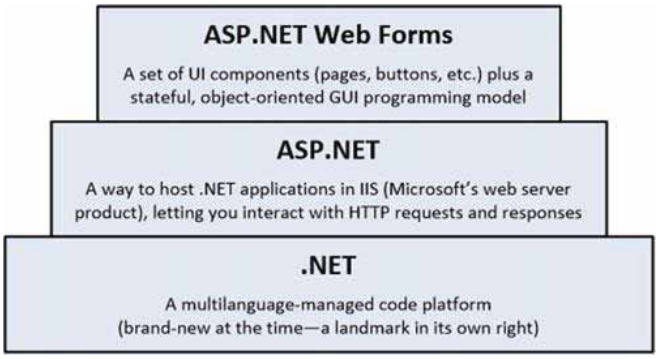
فصل یکم

آشنایی با ASP.NET Core MVC

ASP.NET Core MVC فریم ورک توسعه‌ی برنامه‌های کاربردی[1] مایکروسافت است که اثربخشی و سازماندهی مناسب معماری مدل-نما-کنترلر[2] 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[3] 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

---

[1] Application DevelopmentFramework

[2] Model-View-Controller (MVC)

[3] صفحات منتقل شده در این پروتوکل Stateless هستند. یعنی برخلاف فرم‌های ویندوز، وقتی در وب از صفحه‌ای به صفحه‌ی دیگر حرکت می‌کنید، عملا همه‌ی اطلاعات صفحه‌ی پیشین را از دست می‌دهید، م.

سمت وب[1]، 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 Hnadlers) 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[2]: روش کدنویسی فرم‌های وب، ظاهرا 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 (مثلا 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

---

[1] 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.

[2] Separation of Concerns

تولید شـده با اسـتانداردهای وب همخوانی نداشـت و به سـختی می‌توانسـت با CSS شکل‌دهی شود. افزون بر این، IDهای ایجاد شده در کد HTML با جاوا اسکریپت قابل دستیابی نبودند.

- آزمـایش نـاپـذیری: طراحـان فرم‌های وب نمی‌توانسـتند پیش‌بینی کنند که زمانی آزمایش کد، بخش مهمی از توسـعه‌ی هر نرم‌افزاری می‌شـود. معماری ارائه شـده توسـط آنها در فرم‌های وب، به هیچ روی مناسب روش‌های امروزی آزمایش کد نیست.

باید یادآور شـد که توسعه‌ی فرم‌های وب چندان هم بد نبود. مایکروسافت تا جای ممکن برای هماهنگی با اسـتانداردهای وب و آسـان کردن برنامه‌نویسـی آن، کوشـش کرد و حتی بعدها از برخی از ویژگی‌های نخسـتین ASP.NET MVC در معماری فرم‌های وب هم اسـتفاده کرد. هنوز هم اگر نیاز سـریع 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 سـال ۲۰۰۷، مایکروسـافت، برای پاسـخ‌گویی به انتقادهای فرم‌های وب و محبوب بودن برخی روش‌های توسـعه‌ی جدید مانند Ruby، پلتفرم توسـعه‌ی[1] 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، سـرویس‌های REST، آزمایش‌پذیری کد و تسـلیم بر این باور بود که برنامه‌نویسـان باید بدون وضعیت (Stateless) بودن ذاتی HTTP را بپذیرند.

معرفی MVC، تغییر دیگری را هم در سـیاسـت‌های پیشـین مایکروسـافت موجب شـد. پیش از آن، مایکروسـافت تاکید زیادی بر کنترل اجزای ایجاد کننده‌ی نرم‌افزار داشـت. ولی با معرفی MVC، و کاربرد ابزار متن‌بازی (Open Source) مانند jQuery در آن، مایکروسـافت سـعی کرد از بهترین‌های رقبای خود اسـتفاده کند و از همین روی، متن کد اصـلی MVC را در اختیار برنامه‌نویسـان به عنوان کاربران اصـلی MVC، قرار داد.

۱-۲-۱ مشکل پروژه‌های قدیمی MVC

در زمـان ایجـاد MVC برای مایکروسـافت 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 شـده بود، کار کند. بنابراین طراحان اولیه‌ی MVC مجبور به تغییر تنظیمات و پیکربندی‌هایی شـدند که ربطی به اجرای کد MVC نداشتند ولی به هر حال برای این که همه چیز به درستی کار کند، لازم بودند.

[1] Development Platform

با محبوب شدن MVC، مایکروسافت 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، مایکروسافت شروع به گسترش ASP.NET با ویژگی‌های جدیدی مانند سرویس‌ها 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 مایکروسافت خبر 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. مایکروسافت با این کار، گستره‌ی 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.

۱-۱-۳-۱ 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.

۲-۱-۳-۱ گسترش‌پذیری

ASP.NET Core و ASP.NET Core MVC شامل عناصر (Component) مستقلی هستند. این عناصر دارای مشخصه‌های روشنی بوده و معمولا 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، کاملا جایگزین کنید.

در فصل ۱۳ با این روش‌ها آشنا خواهید شد.

۳-۱-۳-۱ کنترل کامل بر HTML و HTTP

خروجی HTMLای که ASP.NET Core MVC ایجاد می‌کند، کاملا استاندارد است. برای شکل دادن به این HTML می‌توانید از سبک‌های CSS استفاده کنید. افزون بر این، می‌توانید از jQuery، Angualr 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.

۳-۱-۴-۱ آزمایش‌پذیری

از آنجا که در معماری ASP.NET Core MVC رابط کاربری، مدل داده‌ها و کد پردازش کننده 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 آزمایش متن‌باز 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[1] 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.

۳-۱-۵-۱ 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.

---

[1] 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 مسیردهی 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.

۱-۳-۱-۶ 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[1]، عبارت‌ها لاندا[2]، انواع پویا و بی‌نام[3] و کوئری آمیخته با زبان[4] (LINQ) استفاده می‌کند.

۱-۳-۱-۷ چند پلتفرمی[5]

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.

---

[1] Extension Methods
[2] Lamda Expressions
[3] Anonymous and Dynamic Types
[4] Language Integrated Query (LINQ)
[5] Cross Platform

۱-۳-۱-۷ متن باز بودن

بر خلاف فریم ورک‌های گذشته‌ی توسعه‌ی مایکروسافت، اکنون می‌توانید کد سورس ASP.NET Core و 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.

۱-۲ 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

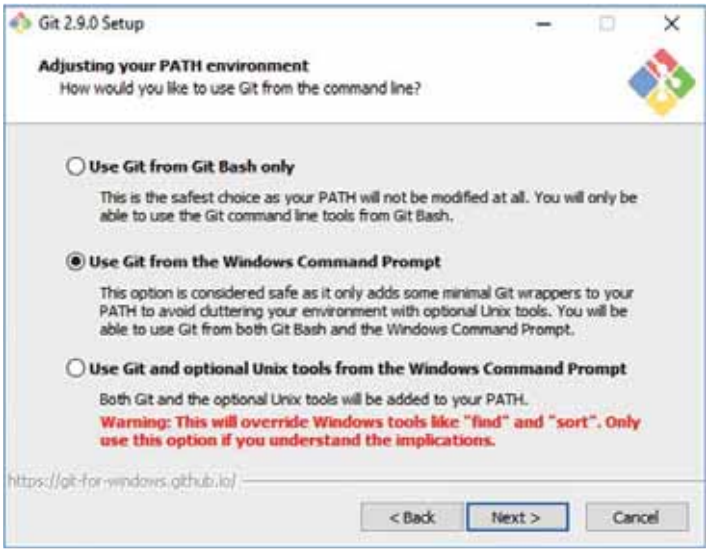
در مرحله‌ی بعد، برای نصب .NET Core باید به آدرس زیر مراجعه کنید:

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:

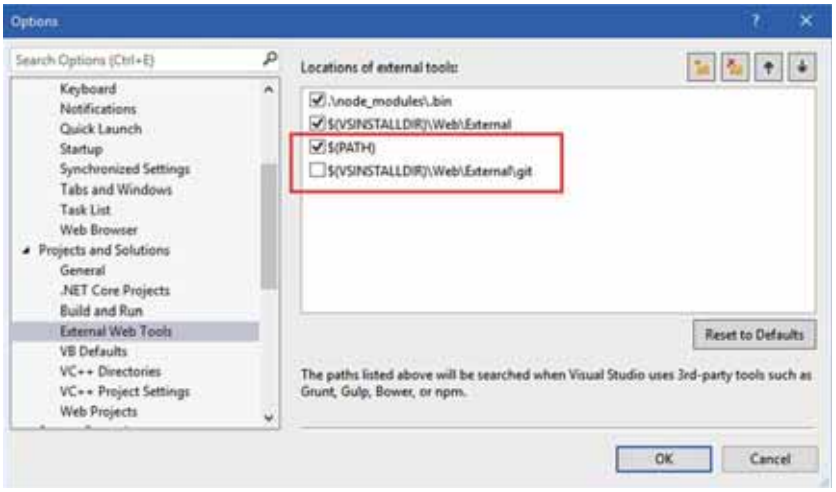
https://git-scm.com/download

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 ۱-۲ 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 ۱-۲

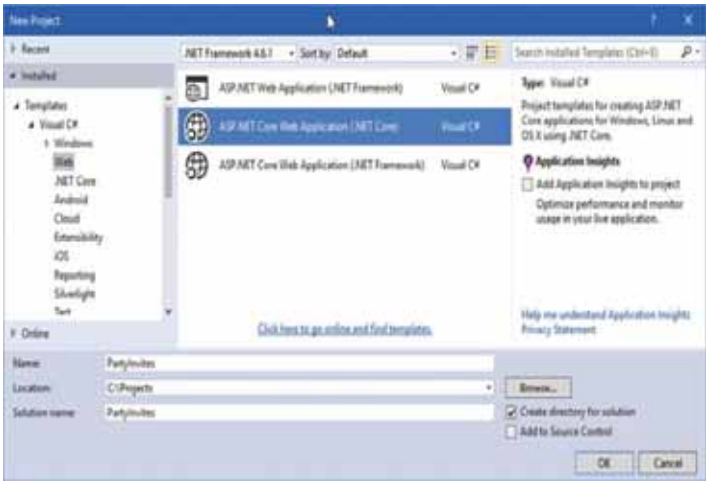
ویژوال استدیو را اجرا کنید و پس از انتخاب گزینه‌ی Options از فهرست Tools، همان‌گونه که در شکل ۲-۲ نشان داده شده است، با انتخاب External Web tools از بخش Projects and Solutions، گزینه‌ی $(VSINSTALLDIR)\Web\External\git item را از حالت انتخاب خارج کنید. به همین صورت مطمئن شوید که گزینه‌ی $(PATH) فعال است. 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

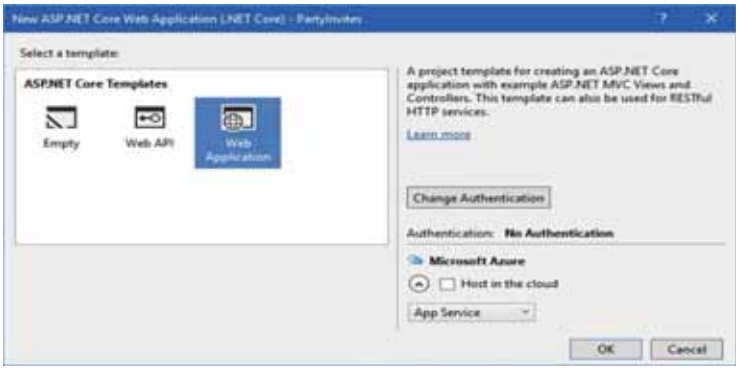
۲-۲ ایجاد پروژه جدید MVC

کار را با ایجاد پروژه‌ی جدیدی از نوع ASP.NET Core MVC در محیط ویژوال استدیو شروع می‌کنیم (شکل ۳-۲). 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 ۳-۲

نام پروژه را PartyInvites گذاشته و همان‌گونه که در شکل ۳-۲ 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 ۴-۲ نشان داده شده است کادر گفتگو، در مورد محتویات پروژه پرسش خواهد کرد.



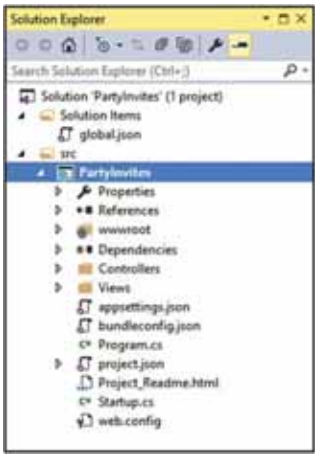
شکل ۴-۲

هر یک از سـه 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 شـکل ۵-۲ نشـان داده شـده اسـت، با کلیک بر روی دکمه‌ی Change Authentication و اطمینان از اینکه گزینه‌ی No Authenticaion انتخاب شـده اسـت، پروژه را ایجاد کنید. در این پروژه نیازی به هیچ نوع خاصی از اعتبارسنجی نخواهیم داشت.

شکل ۵-۲

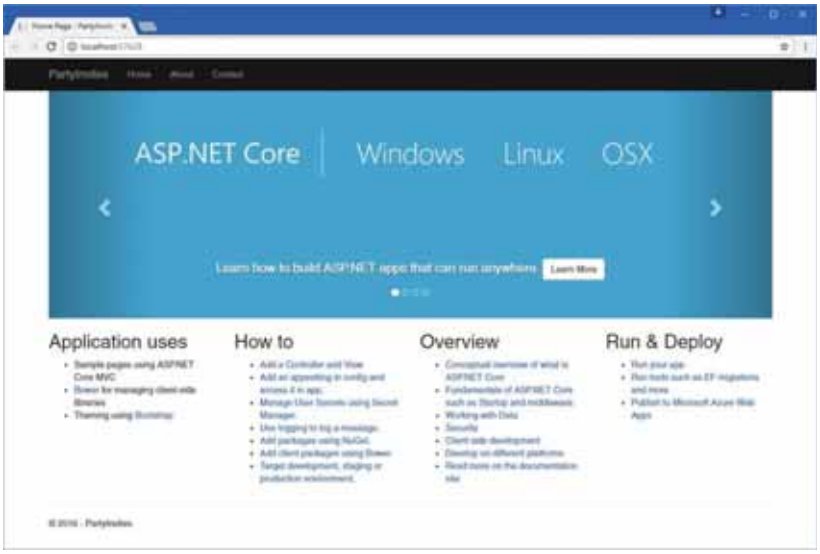
پس از بستن کادر بالا، در بازگشت به کادر پیشین، مطمئن شوید که گزینه‌ی Host in Cloud انتخاب نشده باشد. با کلیک بر روی OK پروژه‌ی PartyInvites را ایجاد کنید. پس از ایجاد پروژه و باز شدن محیط آن در ویژوال اسـتدیو، فایل‌ها و پوشـه‌های مختلفی را در مرورگر سالوشن (Solution Explorer)خواهید دید (شکل ۶-۲).



شکل ۶-۲

برای اجرای پروژه (با محتوای پیش‌فرضی که ویژوال استدیو در آن ایجاد کرده است)، از فهرست Debug گزینـه‌ی Start debugging را انتخاب کنید (اگر پیامی در مورد فعال کردن دیباگ برنامه نمایان شـد، بر روی OK کلیک کنید). با انجام این کار، ویژوال استدیو 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

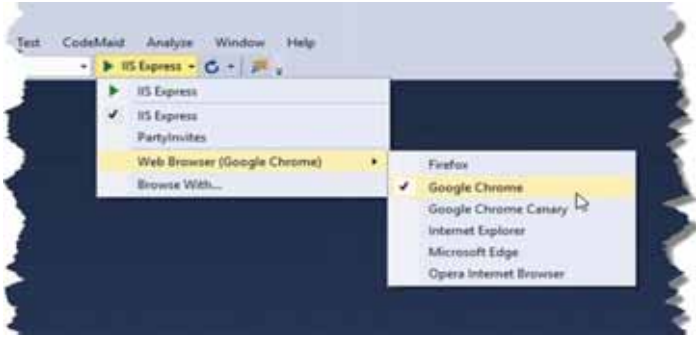
روی کامپیوتر، تقاضـــایی برای نمایش محتوای برنامه به IIS Express (برنامه‌ی میزبان) ارســـال می‌کند (شکل ۷-۲).



شکل ۷-۲

در ادامه‌ی این فصـل، برای ایجاد برنامه‌ی مورد نظرمان، محتوای پیش‌فرض ویژوال اسـتدیو برای پروژه را، به آرامی با کدی مناسـب، 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 شـکل ۸-۲ 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 ۸-۲

۱-۲-۲ 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 داری 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 ۱-۲ 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 Home

```
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 را، به همان روش پیشین اجرا کنید.

مرورگر یک تقاضای[1] 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 ۹-۲).

شکل ۹-۲

در شکل بالا، ویژوال استدیو مرورگر را به پورت ۵۷۶۲۸ هدایت کرده است. از آنجا که ویژوال استدیو 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

[1] 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[1]، نماها[2] و کنترلرها[3]، برنامه‌های کاربردی MVC از روش آدرس‌دهی ASP.NET برای مربوط کردن URLها به کنترلرها و متدهای آنها (اکشن‌ها) استفاده می‌کنند. مسیر[4] قانونی است که معین می‌کند یک درخواست چگونه پاسخ داده شود. ویژوال 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 پورتی 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

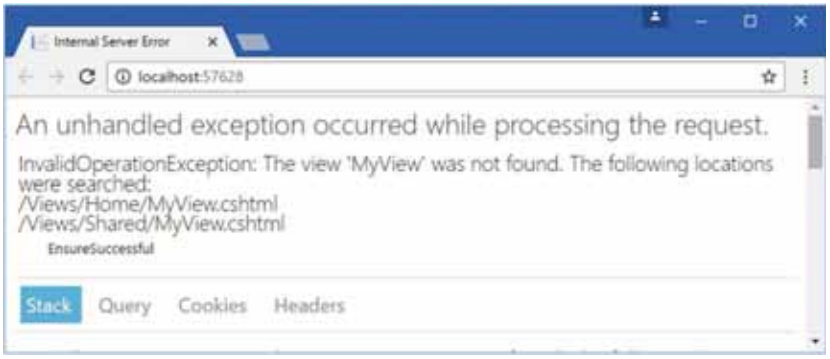
۱-۳-۲ ایجاد نما

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

لیست ۳-۲: ویرایش کنترلر برای نمایش صفحه وب

```
using Microsoft.AspNetCore.Mvc;
namespace PartyInvites.Controllers {
  public class HomeController : Controller {
    public ViewResult 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 ۱۰-۲ برخورد خواهید کرد.



شکل ۱۰-۲

پیام خطای بالا نه تنها نشان دهنده‌ی این است که برنامه به دنبال یافت 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.