به‌نام حضرت دوست

کاربرد Blazor و امکانات امنیتی در Pro ASP.NET Core

Adam Freeman

مهندس نادر نبوی

انتشارات پندار پارس

سرشناسه	:	فریمن، آدام، ۱۹۷۲ – م. Freeman, Adam
عنوان و نام پدیدآور	:	کاربرد Blazor و امکانات امنیتی در Pro ASP.NET Core/ آدام فریمن ؛ [ترجمه] نادر نبوی.
مشخصات نشر	:	تهران : پندار پارس ، ۱۴۰۲.
مشخصات ظاهری	:	xiii، ۳۳۸ص.: مصور(رنگی)، جدول.
شابک	:	978-622-7785-22-7
وضعیت فهرست نویسی	:	فیپا
یادداشت	:	عنوان اصلی: Pro ASP.NET core 6 : Develop cloud-ready web applications using MVC, blazor, and razor pages, 2022.
موضوع	:	مایکروسافت دات‌نت فریم‌ورک Microsoft .NET Framework
موضوع	:	نرم‌افزار مایکروسافت ؛ Microsoft software
موضوع	:	وبگاه‌ها -- برنامه‌های تالیفی ؛ Web sites -- Authoring programs علوم کامپیوتر ؛ Computer science نرم‌افزار -- مهندسی ؛ Software engineering
شناسه افزوده	:	نبوی، نادر، ۱۳۴۰–، مترجم
رده بندی کنگره	:	QA۷۶/۷۶
رده بندی دیویی	:	۲۷۶/۰۰۵
شماره کتابشناسی ملی	:	۹۴۳۸۳۵۹
اطلاعات رکورد کتابشناسی	:	فیپا

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ **www.pendarepars.com**
تلفن: ۶۶۵۷۲۳۳۵ – ۶۶۹۲۶۵۷۸ همراه: ۰۹۱۲۲۴۵۲۳۴۸ **info@pendarepars.com**

نام کتاب	:	**کاربرد Blazor و امکانات امنیتی در Pro ASP.NET Core**
ناشر	:	**انتشارات پندار پارس**
تألیف	:	**آدام فریمن**
ترجمه	:	**نادر نبوی**
چاپ نخست	:	**آذر ۱۴۰۲**
شمارگان	:	**۱۰۰ نسخه**
طرح جلد	:	**رامین شکرالهی**
چاپ، صحافی	:	**روز**
قیمت	:	**۳۰۰.۰۰۰ تومان** **شابک : ۷-۲۲-۷۷۸۵-۶۲۲-۹۷۸**

پیش‌گفتار مترجم

کتابی که در دست دارید، بخش چهارم و پایانی ویرایش نوزدهم کتاب Pro ASP.NET Core 6 نوشته Adam Freeman و شامل ۸ فصل است که در سال ۲۰۲۲ میلادی نگارش یافته است. این بخش به بررسی بلیزر به عنوان جایگزینی برای کاربرد جاوااسکریپت و همچنین ملاحظات امنیتی و تشخیص هویت در ASP.NET Core می‌پردازد. سه بخش نخست کتاب، با عنوان مرجع کاربردی آموزش Pro ASP.NET Core MVC 6 در بهار سال ۱۴۰۲ منتشر و در دسترس خوانندگان قرار گرفت که بر آموزش پایه تا پیشرفته فنآوری یادشده و بر موضوع محوری MVC و صفحات Razor تکیه داشت.

با استفاده از فناوری بلیزر، برنامه‌نویس نیازی به استفاده از جاوااسکریپت و سایر فریمورک‌های وابسته به آن برای انجام تعاملات سمت مشتری نخواهد داشت و همانطور که در خود کتاب، در معرفی آن گفته شده است: "پیشرفت و کاربرد روزافزون فریمورک‌های سمت مشتری جاوا اسکریپت، از آنجا که برنامه‌نویسان را مجبور به یادگیری زبان جدیدی می‌کند، چالشی برای برنامه‌نویسان C# محسوب می‌شود. یادگیری زبانی که از نظر نوشتار و ساختار تفاوت‌های بسیاری با C# به عنوان زبان اصلی برنامه‌نویسی دارد، کار ساده‌ای نیست.

راه حل بلیزر برای این مشکل، ایجاد امکان استفاده از C# برای برنامه‌نویسی سمت مشتری است". بلیزر دارای دو نگارش به نام‌های Blazor Server سرور بلیزر و Blazor WebAssembly یا وب اسمبلی بلیزر است. فصل‌های ۶ تا ۹ کتاب حاضر به بررسی سرور و فصل دهم به وب اسمبلی اختصاص دارد. دو فصل پایانی، فصل‌های یازده و دوازده، به بررسی مسائل مربوط به تشخیص و مدیریت هویت کاربران، ایجاد و حذف کاربران، تعیین و مدیریت نقش‌ها و موضوع مهم اعتبارسنجی اختصاص یافته است. برای کسانی که از پیش با ANC آشنایی داشته و تنها به دنبال یادگیری بلیزر و ملاحضات امنیتی هستند، این بخش، شامل هر هشت فصل آن، به همراه چهار فصل به عنوان مقدمه، به شکل کتاب حاضر در دسترس شما قرار دارد. با این هدف که کتاب حاضر به صورت مستقل قابل استفاده باشد، چهار فصل نخست به مرور سریع ASP.NET Core و چگونگی ایجاد پروژه‌ها و نیز، کار با صفحات Razor پرداخته که امیدوارم به عنوان مرجعی خلاصه و مفید، خواننده محترم را از رجوع به منابع دیگر بی نیاز کند.

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: https://github.com/apress/pro-asp.net-core-6

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

نادر نبوی

پاییز سال ۱۴۰۲

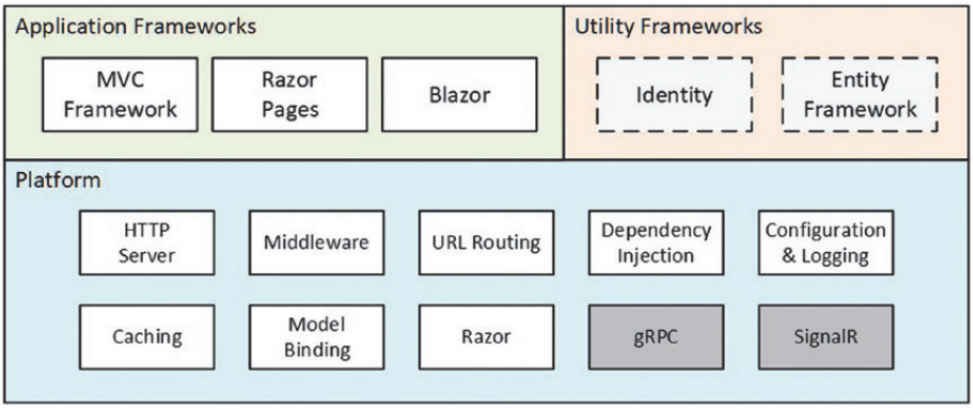
فهرست

فصل یکم

ASP.NET Core در عمل

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. ASP.NET برای نخستین بار در سال ۲۰۰۲ عرضه شد و از آن زمان تا کنون در اثر همه تغییرات و تحولاتی که به خود دیده، تبدیل به نگارش ۶ ANC شده، که موضوع بحث این کتاب است.

همان‌گونه که در شکل ۱-۱ می‌بینید، ASP.NET Core (که از این پس در این کتاب آن را ANC خواهیم نامید) شامل پلت‌فرمی برای پردازش درخواست‌های HTTP، فریم‌ورک‌های اصلی برای ایجاد برنامه‌های کاربردی و فریم‌ورک‌های ثانویه‌ای برای ایجاد ویژگی‌های کمکی و سودمند[3] مورد نیاز برنامه‌نویس، است.



شکل ۱-۱

آشنایی با فریم‌ورک‌های MVC

در شروع کار با NET CORE. تعداد زیاد فریم‌ورک‌های موجود می‌تواند گیج کننده باشد. همان‌گونه که خواهید دید بسیاری از این فریم‌ورک‌ها تکمیل کننده‌ی یکدیگر بوده و مشکلات مختلفی را حل می‌کنند، گرچه برخی از آن‌ها یک مشکل را به روش‌های گوناگون پاسخ می‌دهند.

---

[1] Application DevelopmentFramework
[2] Model-View-Controller (MVC)
[3] Utility Framework

معرفی فریم‌ورک MVC

MVC مدت‌ها پیش از .NET CORE و 6 .NET.، در دورانی که ASP.NET استفاده می‌شد، ارائه شد. ASP.NET نخستین بار بر پایه روش توسعه صفحات وب (Web Pages) استوار بود که تلاش می‌کرد همان تجربه توسعه‌ی فرم‌های ویندوز را، در مورد صفحات وب بکار گیرد[۱]. ولی همان‌طور که می‌دانید این روش کارآمد نبود و مورد استقبال توسعه دهندگان وب قرار نگرفت. فریم‌ورک MVC در همان ایام بر پایه مدلی که بر ویژگی‌های HTTP و HTML استوار بود، ارائه شد.

MVC از الگوی مدل-نما-کنترلر برای ایجاد پروژه‌های وب استفاده می‌کند. الگوی MVC تاکید زیادی بر جداسازی دغدغه‌ها[۲] دارد که بر پایه آن، کارآیی‌های مختلف نرم‌افزار به صورت جداگانه و مستقل پیاده‌سازی می‌شوند. در این روش، کد مربوط به نمایش رابط کاربر در نما آورده می‌شود و مدل نماینده‌ی داده‌های برنامه است. کنترلرها، مسئول پردازش تقاضاهای رسیده و داده‌های مدل و افزون بر این، انتخاب و نمایش نمای مناسب به کاربر هستند. ولی اشکال کار در اینجا بود که نگارش‌های نخستین MVC بر پایه زیرساخت‌های ASP.NET ایجاد شده بودند (که از نظر کارکرد و معماری خیلی با MVC متفاوت است،م) و همین امر موجب پیدایش ویژگی‌های ناکارآمد و مشکل‌سازی شد. با حرکت به سوی .NET Core ، ASP.NET تبدیل به ANC شد و فریم‌ورک MVC بر پایه‌ی پلتفرم قابل توسعه‌ی جدید، دوباره از نو نوشته و ایجاد شد.

معرفی صفحات Razor

در برنامه‌های کاربردی ANC MVC، عنصر نرم‌افزاری موتور نما[۳]، مسئول تولید محتوایی است که برای کاربران فرستاده می‌شود. موتور پیش‌فرض نما، Razor است که فایل‌های HTML را برای فرامینی که محتوای پویا تولید می‌کنند، پردازش می‌کند.

یکی از مشکلات کار با MVC زمان زیادی است که باید پیش از تولید محتوای اصلی صفحات وب مورد استفاده‌ی کاربر، صرف آماده‌سازی پروژه شود. در حالی که در صفحات قدیمی ASP.NET با وجود همه مشکلاتی که داشتند، ایجاد برنامه‌های کاربردی ساده، بیش از چند ساعت وقت نمی‌برد.

در صفحات Razor، کد (مثلا به C#) با محتوای صفحه (HTML) در هم آمیخته می‌شوند. این روش همان سرعت ایجاد صفحات وب پیشین را بدون مشکلات خاص ASP.NET منسوخ شده، فراهم می‌کند. در این کتاب، صفحات Razor را در کنار MVC به کار خواهیم برد. در این روش، بخش‌های اصلی برنامه را با MVC نوشته و برای اهداف ثانویه مانند گزارش‌گیری، از صفحات Razor استفاده خواهیم کرد.

---

[۱] این به معنی سعی در پنهان کردن ویژگی‌های اصلی HTML و پروتکل HTTP بود که مهمترین آن‌ها بدون وضعیت، یا stateless بودن است. از این روی استفاده‌ی بیش از حد از کنترل‌هایی شبیه کنترل‌های ویندوز و سعی بر انتقال وضعیت این کنترل‌ها، در حجم وسیعی از داده، از یک صفحه به صفحه‌ی دیگر، عملا کار با ASP.NET را بسیار مشکل کرده بود.

[2] separation of concerns

[3] View Engine

معرفی Blazor

پیشرفت و کاربرد روزافزون فریم‌ورک‌های سمت مشتری[1] جاوا اسکریپت، از آنجا که برنامه‌نویسان را مجبور به یادگیری زبان جدیدی می‌کند، چالشی برای برنامه‌نویسان #C محسوب می‌شود. یادگیری زبانی که از نظر نوشتار و ساختار تفاوت‌های بسیاری با #C به عنوان زبان اصلی برنامه‌نویسی دارد، کار ساده‌ای نیست.

راه حل Blazor برای این مشکل، ایجاد امکان استفاده از #C برای برنامه‌نویسی سمت مشتری است. Blazor دارای دو نگارش به نام‌های Blazor Server سرور بلیزر و Blazor WebAssembly بلیزر وب است. سرور بلیزیر بخشی از ANC بوده و با استفاده از ارتباطی که با سرور ANC ایجاد می‌کند (از طریق HTTP) کار می‌کند. همان‌گونه که می‌دانید سرور گفته شده دقیقا همان جایی است که کدهای سمت سرور #C اجرا می‌شوند. بلیزر وب که هنوز در مرحله‌ی تجربه و آزمایش قرار دارد، گامی فراتر رفته و سعی می‌کند کدهای #C را در مرورگر کاربر (یعنی سمت مشتری) اجرا کند.

معرفی فریم‌ورک کمکی[2]

دو فریم‌ورک دیگر که به صورت تنگاتنگی با Core کار می‌کنند، عبارتند از Entity Framework و ASP.NET Identity. همان‌گونه که می‌دانید، Entity Framework نرم‌افزار نگاشت شیءگرا–رابطه‌ای مایکروسافت برای ارائه‌ی داده‌های ذخیره شده در یک مدل رابطه‌ای (مثل SqlServer)، به صورت شیءگرا است[3]. از این فریم‌ورک در همه‌ی محیط‌های.NET می‌توان استفاده کرد. در این کتاب، از آن برای دسترسی به پایگاه داده، در پروژه‌های Core استفاده خواهیم کرد.

فریم‌ورک دوم، Entity، فریم‌ورکی است که برای اعتبارسنجی داده‌های لاگین کاربران، تعیین و محدودسازی سطوح دسترسی، مورد استفاده قرار می‌گیرد و در این کتاب هم دارای همین کاربرد است.

معرفی پلتفرم ANC

پلتفرم ANC شامل ویژگی‌های لازم برای دریافت و پردازش درخواست‌های HTTP و به دنبال آن، تولید پاسخ مناسب است. از مهم‌ترین اینها می‌توان از یک سرور HTTP، یک سیستم نرم‌افزاری میانی[4] برای پردازش درخواست‌های رسیده و سایر ویژگی‌هایی که فریم‌ورک برنامه به آنها نیازمند است، مانند مسیریابی[5] URL و موتور نمای[6] Razor نام برد.

---

[1] Client side
[2] Utility Framework
[3] ORM یا نرم افزاری که جداول رابطه‌ای پایگاه داده‌ای مانند SQL-Server را به شکل کلاس‌هایی که در یک سیستم شیءگرا از یکدیگر مشتق می‌شوند، در اختیار محیط برنامه‌نویسی .NET قرار می‌دهد،م.
[4] Middleware
[5] URL Routing
[6] Razor View Engine

سخنی در مورد ساختار کتاب

برای بهره‌برداری بهینه از کتاب، خواننده باید ضمن آشنایی با مفاهیم پایه‌ی توسعه‌ی وب، با کارکرد HTML و CSS آشنا بوده و علاوه بر اینها، دارای دانشی عملی در کار با C# باشد. از آنجا که تاکید کتاب بر کارکرد زبان C# با ANC است، تجربه‌ی کار با برنامه‌های توسعه‌ی سمت مشتری مانند JavaScript مورد نیاز نیست.

نرم‌افزار مورد نیاز برای مثال‌های کتاب

برای پیاده‌سازی مثال‌های کتاب (که اکیدا توصیه می‌شود) نیاز به یک ویرایشگر کد (خود ویژوال استدیو و یا نرم‌افزار Visual Studio Code)، کیت توسعه‌ی[1] .NET Core. و نگارش LocalDB از Sqlserver خواهید داشت. همه‌ی موارد گفته شده، از سایت مایکروسافت بدون پرداخت هزینه‌ای قابل دسترس هستند (و در زمان ترجمه‌ی کتاب، بدون نیاز به VPN و با آی پی ایران قابل دانلود هستند).

کتاب برای ویندوز نوشته شده است. خود من ویندوز ۱۰ را بکار بردم ولی هر نوع نگارشی از ویندوز که ویژوال استدیو و .NET Core. در آن قابل اجرا باشند، می‌تواند مورد استفاده قرار گیرد. ANC می‌تواند بر روی پلتفرم‌های دیگری غیر از ویندوز اجرا شود، ولی بیشتر مثال‌های کتاب به SQL Server LocalDB نیاز دارند، که مختص ویندوز است.

مطالب ارائه شده در کتاب

بخش نخست کتاب، شامل فصل‌های ۱ تا ۱۱، به معرفی ANC می‌پردازد. افزون بر آماده‌سازی محیط برنامه‌نویسی و توسعه‌ی مورد نیاز و ایجاد اولین پروژه، با مهمترین ویژگی‌های C# در کار با ANC آشنا خواهید شد. بخش زیادی از این فصل‌ها به پیاده‌سازی پروژه‌ای به نام فروشگاه ورزشی (SportsStore Project) می‌پردازد که در طی آن از مراحل آغازین طراحی مفاهیم پروژه، تا پیاده‌سازی و انتشار آن، با ویژگی‌های اصلی و مهم ANC و چگونگی ترکیب این ویژگی‌ها با هم و استفاده از آنها، آشنا خواهید شد.

در بخش دوم که شامل فصل‌های ۱۲ تا ۱۷ می‌شود، وارد جزئیات ویژگی‌های اصلی ANC خواهیم شد. ضمن آشنایی با چگونگی پردازش درخواست‌های HTTP، موضوعاتی مانند ایجاد و بکارگیری عناصر میانی (Middleware Components)، ایجاد مسیرها و مسیریابی (Routes)، تعریف و استفاده از سرویس‌ها و در پایان، کار با Entity Framework Core (که از این پس، به اختصار EF Core نامیده خواهد شد) را مورد بررس قرار خواهیم داد. و در آخر در بخش سوم، فصل‌های ۱۸ تا پایان کتاب، با چگونگی ایجاد انواع مختلف پروژه آشنا خواهید شد. مواردی مانند وب سرویس‌های REST (RESTful web services)، کاربرد Razor و کنترلرها در تهیه‌ی پروژه‌های HTML و ویژگی‌های مهمی مانند نماها (Views) و عناصر آنها (View Components) و تگ‌های کمکی (Tag Helpers) بررسی خواهند شد.

---

[1] .Net Core software development kit

فصل دوم

شروع به کار

بهترین راه برای آشنایی با یک فریم‌ورک توسعه‌ی نرم‌افزار، بکارگیری آن است. در این فصل با چگونگی آماده‌سازی و ایجاد یک محیط توسعه‌ی ANC و نحوه‌ی اجرای آن آشنا خواهید شد.

انتخاب ویرایشگر کد و محیط توسعه

مایکروسافت دو ابزار اصلی برای کدنویسی معرفی می‌کند؛ ویژوال استدیو و ویژوال استدیو کد (Visual Studio Code). ویژوال استدیو ابزار معمول توسعه‌ی برنامه‌های NET. است که طیف وسیعی از راهکارها و ویژگی‌های مختلف را برای انواع برنامه‌های کاربردی NET. در اختیار قرار می‌دهد. مشکل آن، مصرف زیاد منابع (حافظه و پردازش) و کند بودن آن است. برخی از ویژگی‌های آن هم، گرچه برای کمک به برنامه‌نویس ایجاد شده‌اند، ولی می‌توانند مانعی بر سر راه توسعه باشند.

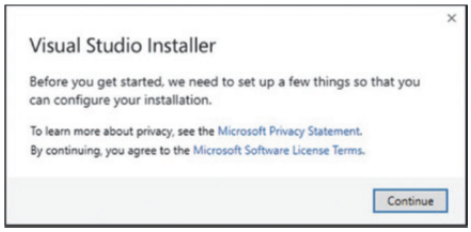
"ویژوال استدیو کد" نگارش سبکی از ویژوال استدیو است که همه‌ی امکانات لازم برای توسعه‌ی ANC را در اختیار دارد.

همه‌ی مثال‌های این کتاب دارای دستورات لازم برای بکارگیری هر یک از ادیتورهای گفته شده هستند و شما می‌توانید هر کدام را که ترجیح می‌دهید، نصب و استفاده کنید.

ویژوال استدیو ابزار بیشتری برای ایجاد انواع فایل‌های مورد نیاز NET Core. در اختیار می‌گذارد، بنابر این اگر در برنامه‌نویسی NET. مبتدی هستید، بهتر است که ویژوال استدیو را نصب کنید. به این ترتیب می‌توانید مطمئن باشید که به نتایج مورد نظر هر یک از مثال‌ها، به درستی، دست خواهید یافت.

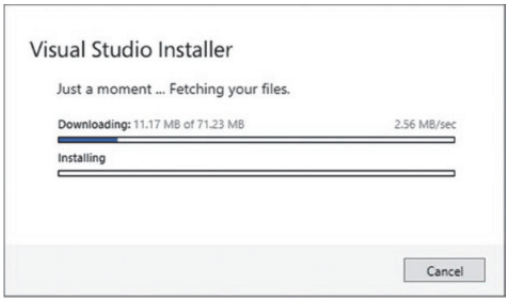
نصب ویژوال استدیو

برای ANC 6 نیاز به ویژوال استدیو ۲۰۲۲ دارید. در این کتاب از نگارش مجانی ویژوال استدیو (نگارش کامیونیتی Community Edition) استفاده شده که می‌توانید آن را از سایت www.visualstudio.com (با آی پی ایران و بدون نیاز به وی پی ان، در زمان ترجمه‌ی کتاب) دانلود کنید. پس از اقدام به دانلود و اجرای نصاب، با پنجره‌ی شکل ۲–۱ روبرو خواهید شد:



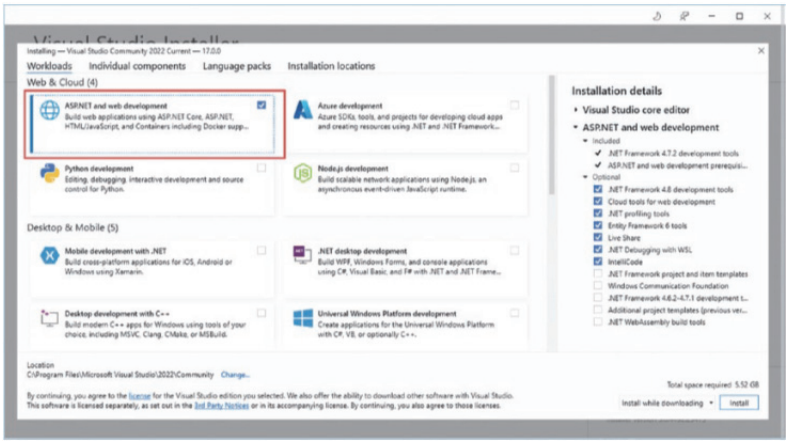
شکل ۲–۱

با کلیک بر روی "Continue" عمل دانلود فایل‌های لازم، مانند شکل ۲-۲، شروع خواهد شد:



شکل ۲-۲

با تکمیل دانلود، با گزینه‌هایی برای نصب روبرو می‌شوید. مطمئن شوید که مانند شکل ۲-۳، گزینه‌ی "ASP.NET and web development" حتما انتخاب شده باشد.



شکل ۲-۳

با توجه به شکل ۲-۴، با انتخاب بخش "Individual components"، حتما گزینه‌ی SQL Server Express 2019 LocalDB را انتخاب کنید. از این نرم‌افزار برای ذخیره‌سازی داده‌ها در مثال‌های آتی کتاب استفاده خواهیم کرد.

شکل ۲-۴

در پایان، کلیک بر روی دکمه‌ی install موجب دانلود فایل‌های لازم و نصب آنها خواهد شد. امکان دارد که برای تکمیل نصب، نیاز به ریست شدن کامپیوتر داشته باشید.

نصب .NET SDK

امکان دارد نگارشی از SDK که توسط نصب کننده‌ی ویژوال استدیو نصب می‌شود، همانی نباشد که مورد نیاز مثال‌های این کتاب است. بنابرین به آدرس زیر رفته و نگارش SDK 6.0.0 را دانلود و نصب کنید: https://dotnet.microsoft.com/download/dotnet-core/6.0 پس از نصب، پنجره‌ای جدید برای Command promt power shell ویندوز باز کرده و فرمان زیر را در آن وارد کنید:

```
dotnet --list-sdks
```

این فرمان، لیستی از SDK های نصب شده را نمایش می‌دهد. لیست زیر، نمایشی از ویندوزی است که .NET. برای نخستین بار در آن نصب شده است:

```
6.0.100 [C:\Program Files\dotnet\sdk]
```

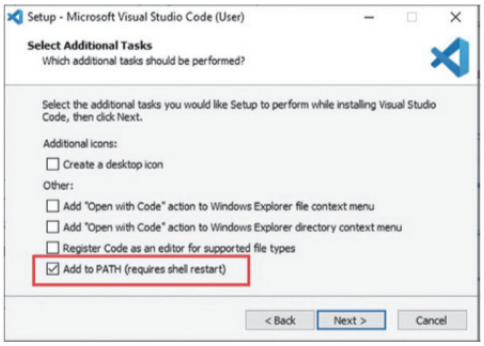
اگر از پیش، با نگارش‌های مختلفی از .NET. کار کرده باشید، امکان دارد لیست طولانی‌تری ببینید:

```
3.1.101 [C:\Program Files\dotnet\sdk]
5.0.100 [C:\Program Files\dotnet\sdk]
5.0.401 [C:\Program Files\dotnet\sdk]
6.0.100 [C:\Program Files\dotnet\sdk]
```

صرف نظر از هر تعداد SDk نصب شده، مطمئن شوید که حتما دارای نگارش 6.0.1xx هستید.

نصب Visual Studio code

در صورتی که تصمیم بر استفاده از Visual Studio Code دارید، برنامه‌ی نصب آن را از سایت https://code.visualstudio.com دانلود کنید. با اجرای برنامه‌ی نصب، مطمئن باشید که گزینه‌ی Add to PATH مانند شکل ۲-۵ انتخاب شده باشد:



شکل ۲-۵

از آنجا که "ویژوال استدیو کد" دارای .NET SDK نیست، باید آن را بطور جداگانه دانلود کنید. با رفتن به آدرس https://dotnet.microsoft.com/download/dotnet-core/6.0 و انتخاب نگارش ۶.۰.۰ آن را دانلود کنید. پس از اجرای برنامه‌ی نصب کننده و پایان کار آن، پنجره‌ای جدید از خط فرمان ویندوز (PowerShell) باز کرده و فرمان زیر را در آن اجرا کنید:

```
dotnet --list-sdks
```

این فرمان، لیستی از SDK های نصب شده را نمایش می‌دهد. لیست زیر، نمایشی از ویندوزی است که .NET برای نخستین بار در آن نصب شده است:

```
6.0.100 [C:\Program Files\dotnet\sdk]
```

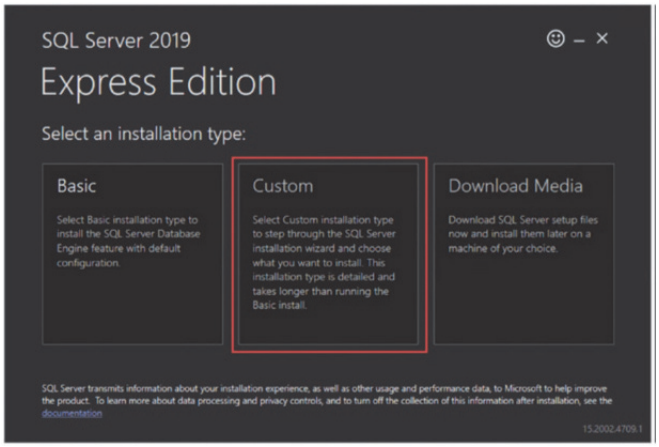
اگر از پیش با نگارش‌های مختلفی از .NET کار کرده باشید، امکان دارد لیست طولانی‌تری ببینید:

```
3.1.101 [C:\Program Files\dotnet\sdk]
5.0.100 [C:\Program Files\dotnet\sdk]
5.0.401 [C:\Program Files\dotnet\sdk]
6.0.100 [C:\Program Files\dotnet\sdk]
```

در اینجا هم، صرف نظر از هر تعداد SDk نصب شده، مطمئن شوید که حتما دارای نگارش 6.0.1xx هستید.

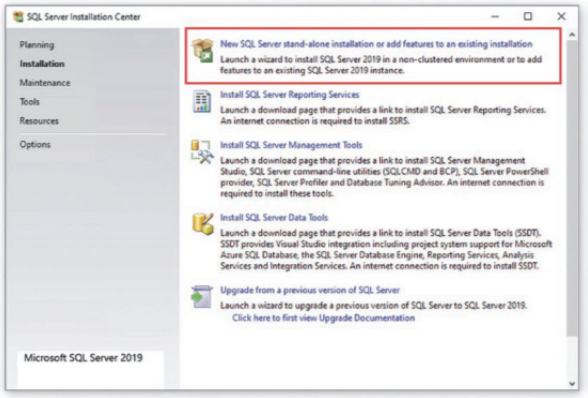
نصب SQL Server LocalDB

مثال‌های پایگاه داده‌ی این کتاب از LocalDB، که بخشی از نگارش SQL Express از SQL Server ، با کمترین پیکربندی است، استفاده می‌کنند. این نرم‌افزار، به صورت کاملا مجانی از آدرس https://www.microsoft.com/en-in/sql-server/sql-server-downloads قابل دانلود است. همان‌گونه که در شکل ۲-۶ می‌بینید، هنگام دانلود از بخش "Custom" استفاده کنید:



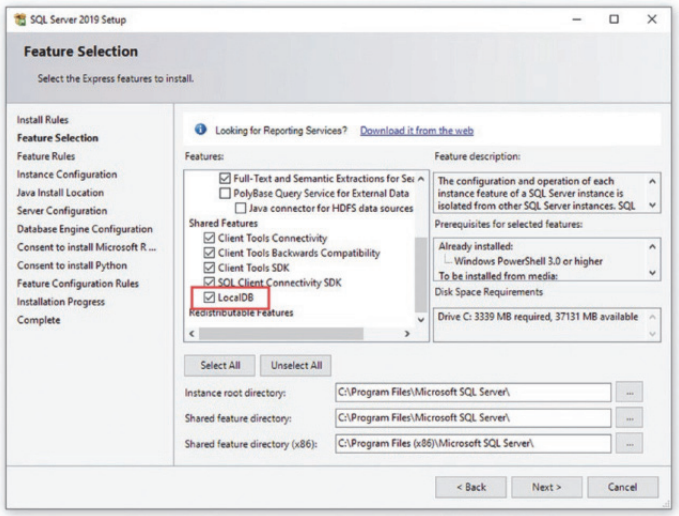
شکل ۲-۶

در ادامه، باید محلی را برای دانلود فایل‌های نصب شده بر روی کامپیوتر خود انتخاب کنید. پس از کلیک بر روی دکمه‌ی "Install"، عمل دانلود شروع می‌شود. وقتی پنجره‌ای مانند شکل ۲–۷ ظاهر شد، گزینه‌ی نخست را برای ایجاد وهله‌ی جدیدی از SQL Server، انتخاب کنید.



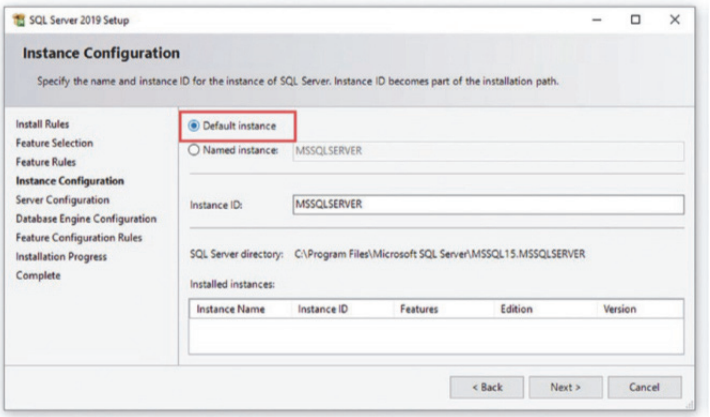
شکل ۲–۷

در ادامه، همه‌ی گزینه‌های پیش‌فرض را همانطور که نشان داده می‌شوند، قبول کنید. هنگامی که به پنجره‌ی انتخاب ویژگی‌های مورد نظر برای نصب، مانند شکل ۲–۸ می‌رسید، مطمئن باشید که گزینه‌ی مربوط به LocalDB انتخاب شده باشد. افزون بر این، ممکن است بخواهید گزینه‌های مربوط به R و Python را از حالت انتخاب خارج کنید که به هر حال، کاربردی در این کتاب نخواهند داشت (و البته به زمان زیادی برای دانلود و نصب نیاز دارند).



شکل ۲–۸

در صفحه‌ی انتخاب چگونگی پیکربندی وهله‌ی نصب شده، مانند شکل ۲–۹، گزینـه‌ی "Default instance" را انتخاب کنید:



شکل ۲–۹

در پنجره‌های بعدی، گزینه‌های پیش‌فرض را قبول کنید. پس از پایان نصب، آخرین آپدیت SQL Server را دانلود و نصب کنید. در هنگام نگارش کتاب، آخرین آپدیت در آدرس زیر قابل دسترس است:

https://support.microsoft.com/en-us/topic/kb5005679-cumulative-update-13-for-sql-server-2019-c1be850-460a-4be4-a569-fe11f0adc535

البته دسترسی به این آدرس، با جستجوی KB5005679 در جستجوگری مانند گوگل شاید راحت‌تر باشـد. دقت کنید که در زمان مطالعه‌ی این کتاب، ممکن است نگارش‌هـای جدیـدتری از ایـن آپـدیت در دسـترس باشد.

ایجاد یک پروژه‌ی ANC

سرراست‌ترین راه برای ایجاد پروژه، استفاده از خط فرمان است. پس از باز کردن یک پنجره‌ی جدید خط فرمان ویندوز (Powershell) و حرکت به درایو و پوشه‌ی مورد نظرتان برای ایجاد پروژه، فرامین زیـر را مانند لیست ۲–۳ وارد کنید:

لیست ۲–۳ کد ایجاد پروژه و سالوشن

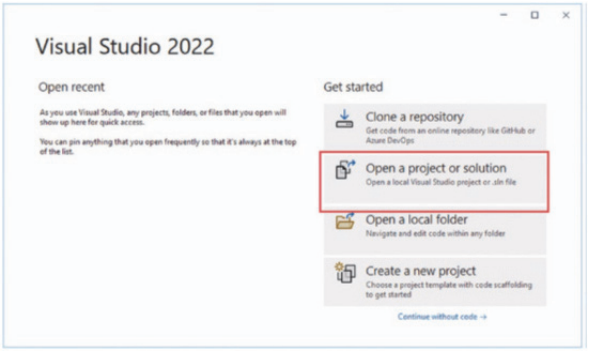
```
dotnet new globaljson --sdk-version 6.0.100 --output FirstProject
dotnet new mvc --no-https --output FirstProject --framework net6.0
dotnet new sln -o FirstProject
dotnet sln FirstProject add FirstProject
```

نخستین فرمان، پس از ایجاد پوشه‌ی FirstProject، فایلی به نام global.json بـه همـراه نگارشـی از .NET. که به کار خواهد رفت (۶.۰.۱۰۰)، را به آن اضافه می‌کند. این کار موجب حصول اطمینـان از دسـتیابی بـه

نتایج درست در اجرای مثال‌های کتاب خواهد شد. فرمان دوم، پروژه‌ی جدیدی به نام FirstProject، با استفاده از الگوی MVC ایجاد می‌کند. الگوی MVC (Template)، یکی از چندین الگوی ارائه شده برای ایجاد پروژه‌های جدید در ANC است. این الگو، پروژه‌ای ایجاد می‌کند که به طرز مناسبی برای فریم‌ورک MVC پیکربندی شده است. اگر در حال حاضر چیزی از MVC یا به طور کلی الگوهای مختلف ASP.NET نمی‌دانید، نگران نباشید؛ تا پایان کتاب با جزئیات کار MVC به طور کامل آشنا خواهید شد. دو فرمان بعدی به ترتیب، ابتدا یک سالوشن ایجاد کرده و پس از آن، پروژه را به سالوشن اضافه می‌کنند (اسامی سالوشن و پروژه یکسان هستند).

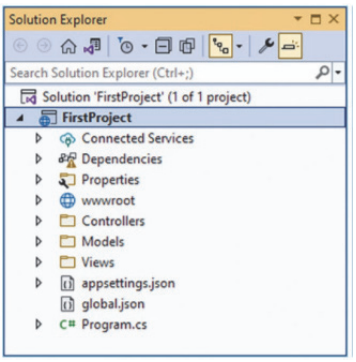
باز کردن پروژه در ویژوال استدیو

پس از اجرای ویژوال استدیو، مانند شکل ۲–۱۰، بر روی "Open a project or solution" کلیک کنید:



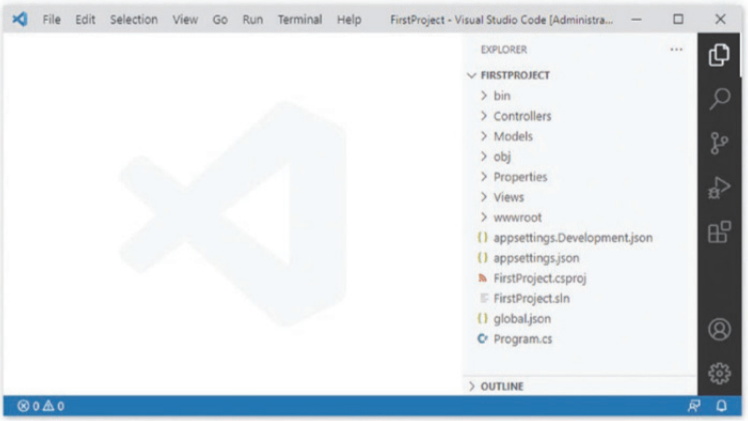
شکل ۲–۱۰

پس از ورود به پوشه‌ی FirstProject و انتخاب فایل FirstProject.sln (فایل سالوشن پروژه)، آن را با کلیک بر روی دکمه‌ی Open، باز کنید. ویژوال استدیو، پروژه را مانند شکل ۲–۱۱ باز کرده و محتویات آن را نمایش می‌دهد. همان‌گونه که قبلا گفته شد، فایل‌های این پروژه بر پایه‌ی الگو یا قالب MVC ایجاد شده‌اند.



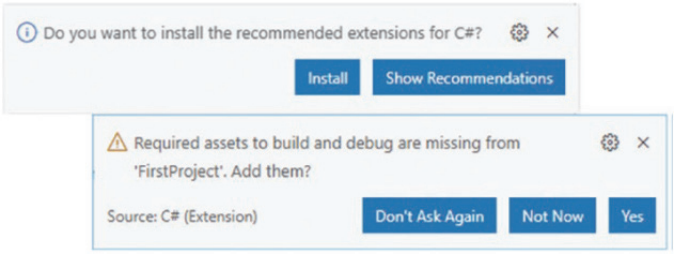
شکل ۲–۱۱

به عنوان روشی دیگر، برای این که پروژه را با ادیتور "ویژوال استدیو کد" باز کنید، پس از اجرای ادیتور، از منوی File گزینه‌ی Open Folder را انتخاب کنید. سپس به پوشه‌ی FirstProject بروید و در پایان، بر روی دکمه‌ی Select Folder کلیک کنید. پروژه مانند شکل ۲-۱۲ در محیط ادیتور باز شده و محتویات آن نمایش داده می‌شود.



شکل ۲-۱۲

پروژه‌ای که برای نخستین بار در "ویژوال استدیو کد" باز می‌شود، نیاز به پیکربندی‌های دیگری هم دارد. به عنوان اولین گام، بر روی فایل Program.cs در پنجره‌ی کاوشگر (Explorer Pane) کلیک کنید تا باز شود. این کار، موجب نمایش پنجره‌ای مبنی بر درخواست تأیید برای افزودن ویژگی‌های مورد نیاز پروژه، مانند شکل ۲-۱۳ می‌شود. اگر هیچ پروژه‌ی C# تاکنون باز نکرده باشید، پنجره‌ای هم برای نصب ویژگی‌های لازم برای محیط C# باز خواهد شد (پنجره‌ی نخست در شکل ۲-۱۳).



شکل ۲-۱۳

آن طور که مناسب می‌دانید، بر روی دکمه‌ی Install یا Yes کلیک کنید. ویژوال استدیو اقدام به دانلود و نصب موارد مورد نیاز خواهد کرد.

اجرای برنامه ANC

اجرای برنامه‌ها، هم از طریق ویژوال استدیو و هم در محیط "ویژوال استدیو کد"، به طور مستقیم امکان‌پذیر است. با این حال در این کتاب از خط فرمان استفاده می‌کنیم که هم قابل اطمینان‌تر بوده و هم این که از نظر آموزشی کارآیی بیشتری دارد. در آینده، پس از یادگیری مطالب، خودتان به هر روشی که برایتان راحت‌تر است، برنامه‌ها را اجرا خواهید کرد.

در زمان ایجاد پروژه، فایلی هم به نام LaunchSettings.json در پوشه‌ی Properties ایجاد می‌شود. محتویات این فایل تعیین کننده‌ی پورت HTTP است که برای گوش‌سپردن به درخواست‌های HTTP به کار خواهد رفت. پس از باز کردن این فایل با دو بار کلیک بر روی نام آن، در پنجره‌ی کاوشگر، شماره‌ی پورت گفته شده را با توجه به لیست ۲-۴، به ۵۰۰۰ تغییر دهید:

لیست ۲-۴

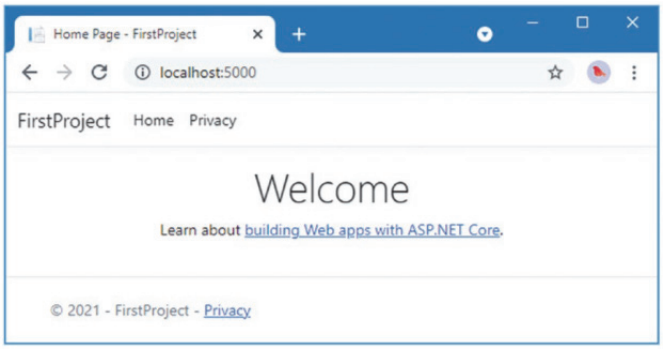
```
{
        "iisSettings": {
                "windowsAuthentication": false,
                "anonymousAuthentication": true,
                "iisExpress": {
                        "applicationUrl": "http://localhost:5000",
                        "sslPort": 0
                }
        },
        "profiles": {
                "FirstProject": {
                                "commandName": "Project",
                                "dotnetRunMessages": true,
                                "launchBrowser": true,
                                "applicationUrl": "http://localhost:5000",
                                "environmentVariables": {
                                        "ASPNETCORE_ENVIRONMENT": "Development"
                                }
                        },
        "IIS Express": {
                        "commandName": "IISExpress",
                        "launchBrowser": true,
                        "environmentVariables": {
                        "ASPNETCORE_ENVIRONMENT": "Development"
                        }
                }
        }
}
```

گرچه فقط آدرس موجود در بخش "profiles" بر روی ابزار خط فرمان NET. موثر است، در اینجا برای اطمینان بیشتر، هر دو مورد را تغییر داده‌ایم. پس از باز کردن پنجره‌ی جدید فرمان ویندوز (Powershell)، از طریق منوی شروع "start" ویندوز، به پوشه‌ی FirstProject که شامل فایل پروژه‌ی (.csproj) FirstProject است بروید و فرمان لیست ۲-۵ را برای اجرای برنامه وارد کنید.

لیست ۲-۵

```
dotnet run
```

فرمان dotnet run پروژه‌ی موجود در پوشه‌ی جاری را کامپایل و اجرا می‌کند. پس از اجرای برنامه، پنجره‌ی مرورگر را باز کنید و آدرس http://localhost:5000 را در نوار آدرس وارد کنید. این درخواست، پاسخ نشان داده شده در شکل ۲-۱۴ را ایجاد می‌کند:



شکل ۲-۱۴

در پایان، فشردن کلیدهای Control+C اجرای برنامه را متوقف می‌کند.

آشنایی با Endpoint

در برنامه‌های ANC درخواست‌های ورودی توسط نقاط پایانی (Endpoints)، دریافت و پردازش می‌شوند. نقطه‌ی پایانی تولید کننده‌ی پاسخ مندرج در لیست ۲-۴، یک اکشن (Action) است. این اکشن، به عنوان یک متد استاندارد، به زبان C# نوشته شده است. هر اکشن در یک کنترلر (Controller) نوشته می‌شود که خود کلاسی مشتق شده از کلاس پایه‌ی Microsoft.AspNetCore.Mvc.Controller است.

به طور کلی، هر متد عمومی نوشته شده در یک کنترلر، یک اکشن است. این اکشن را می‌توان برای پردازش یک درخواست HTTP، فراخوانی کرد[۱]. روش معمول در پروژه‌های ANC این است که کلاس‌های کنترلر را در پوشه‌ای به نام Controllers قرار می‌دهند. در مثال حاضر، این پوشه از همان ابتدا توسط الگوی MVC که برای پروژه انتخاب کردید، ساخته شده است.

در اینجا نام کلاس کنترلر، HomeController.cs و نام خود کنترلر، Home است. بنابراین، فایل‌های کلاس‌های کنترلر، از یک نام دلخواه (در این مورد Home) و به دنبال آن کلمه‌ی Controller به صورت چسبیده به هم، استفاده می‌کنند. باید اشاره کرد که کنترلر Home، کنترلر پیش‌فرض در پروژه‌های ANC است.

[۱] ساده‌ترین درخواست HTTP، وارد کردن آدرس یک صفحه توسط کاربر است. در این وضعیت، اکشنی که به صورت یک متد در یک کنترلر نوشته شده، در ابتدایی‌ترین حالت، محتوای صفحه‌ی مورد نظر را نمایش خواهد داد.

اکنون فایل HomeController.cs را در پنجره‌ی کاوشگر ویژوال استدیو (یا ویژوال استدیو کد) پیدا، و بر روی آن کلیک کنید تا محتوایش مانند لیست ۲-۵ نمایش داده شود:

لیست ۲-۵

```
using System.Diagnostics;
using Microsoft.AspNetCore.Mvc;
using FirstProject.Models;
namespace FirstProject.Controllers;
public class HomeController : Controller {
    private readonly ILogger<HomeController> _logger;
    public HomeController(ILogger<HomeController> logger) {
     _logger = logger;
    }
    public IActionResult Index() {
     return View();
   }
    public IActionResult Privacy() {
     return View();
    }
    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
     NoStore = true)]
    public IActionResult Error() {
     return View(new ErrorViewModel { RequestId = Activity.Current?.Id
      ?? HttpContext.TraceIdentifier });
    }

}
```

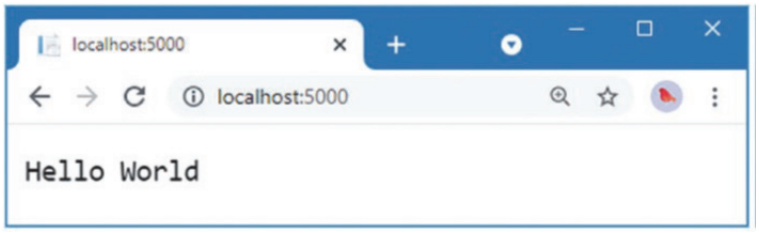
در گام بعد، محتوای این کلاس را با آنچه که در لیست ۲-۶ می‌بینید، تغییر دهید. در اینجا، همراه با حذف عبارت‌های using اضافی و همه‌ی متدها به غیر از یکی از آنها، نوع خروجی متد و محتوای آن هم تغییر کرده است:

لیست ۲-۶

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

}
```

در پی این تغییرات، در کنترلرHome تنها یک متد اکشن به نام Index تعریف شده است. این متد، عبارت Hello World را باز می‌گرداند. اکنون با استفاده از همان روش خط فرمان، پس از اجرای برنامه در پوشه‌ی FirstProject، آدرس http://localhost:5000 را در مرورگر وارد کنید. این درخواست شما، توسط متد اکشن Index در کنترلر Home پردازش و پاسخ داده می‌شود. رشته‌ی تولید شده توسط این متد، Hello World، پاسخ به درخواست HTTP است (شکل ۲-۱۵).



شکل ۲-۱۵

آشنایی با مفهوم مسیر[1]

اینک این سوال مطرح می‌شود که در پاسخ یک درخواست، کدام یک از اکشن‌ها، که درون کنترلرها دسته‌بندی شده‌اند، اجرا شود؟ در ANC سیستم مسیریابی[2] مسئول تعیین نقطه‌ی پایانی یا همان اکشنی است که باید به درخواست رسیده پاسخ دهد. مسیر یا Route، قانونی است که چگونگی پاسخ به درخواست را مشخص می‌کند. در هنگام ایجاد پروژه، یک مسیر پیش‌فرض برای شروع کار ایجاد شده است. با درخواست هر یک از URLهای زیر، همان اکشن Index در کنترلر Home، اجرا خواهد شد:

```
/
/Home
/Home/Index
```

بنابراین، وقتی مرورگری تقاضاهایی مانند http://yoursite/ یا http://yoursite/home را ارسال می‌کند، خروجی را از متد Index در کلاس HomeController دریافت می‌کند. این وضعیت را می‌توانید با تغییر آدرس مرورگر امتحان کنید. در حال حاضر، این آدرس http://localhost:5000 است که البته اگر از ویژوال استدیو استفاده می‌کنید، شماره‌ی پورت ممکن است متفاوت باشد. اگر رشته‌های Home/ یا Home/Index/ را به انتهای آدرس اضافه کنید، همان نتیجه‌ی Hello World را مشاهده خواهید کرد.

چگونگی پردازش HTML

خروجی مثال پیش فقط یک رشته متنی (Hello World) بود، این در حالی است که در بیشتر مواقع، خروجی متد اکشن باید صفحه‌ی HTML باشد. برای ایجاد پاسخ HTML در برابر درخواست مرورگر، نیاز به یک نما (View) داریم. نماها مشخص می‌کنند که ANC چگونه باید نتیجه‌ی تولید شده توسط متد Index را به شکل پاسخ HTML مناسب برای نمایش در مرورگر درآورد.

گام نخست، همان‌گونه که در لیست ۲-۷ نشان داده شده، تغییر مناسب در محتوای متد اکشن Index است. تغییرات به صورت پررنگ نشان داده شده‌اند.

لیست ۲-۷ پردازش و نمایش نما، در HomeController.cs در پوشه‌ی Controllers

```
using Microsoft.AspNetCore.Mvc;
```

[1] Route
[2] Routing System