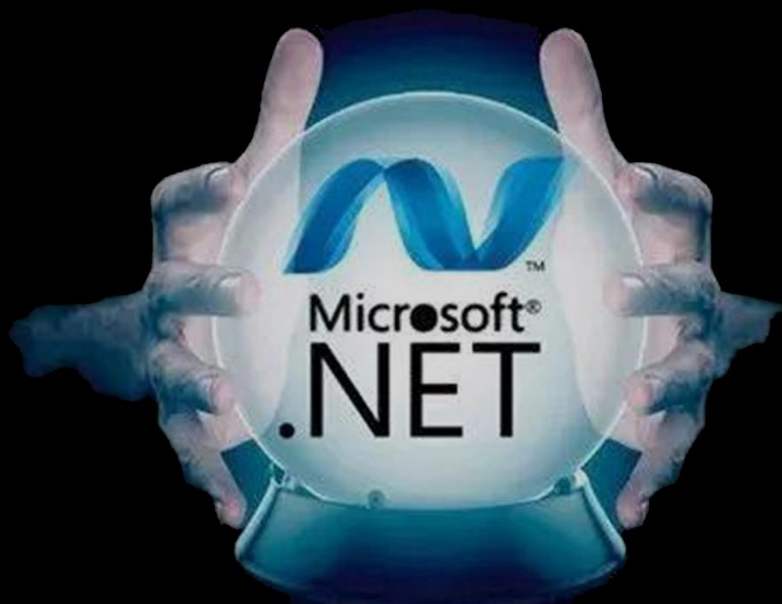


آدام فریمن

آموزش حرفه ای

ASP.NET Core 6



جلد چهارم:

اجرای پروژه مثال

مروّت گیوی

ترجمه  
Parjomiac

آدام فریمن

**آموزش حرفه‌ای ASP.NET Core 6**

**جلد چهارم: اجرای پروژه مثال**

ترجمه

مرّوت گیوی

انتشارات ترجمک

تابستان ۱۴۰۲

# آموزش حرفه‌ای ASP.NET CORE 6

## توسعه وب با ASP.NET Core

### توجه :

کتاب حاضر ترجمه **مرّوت گیوی** از انتشارات ترجمک می‌باشد. فایل کتاب حاوی اطلاعات DRM (مدیریت حقوق دیجیتال) است. وقتی برای اولین بار فایل را باز می‌کنید، کد شناسایی کتاب به همراه آدرس IP سیستم شما ذخیره شده و زمانیکه آنلاین شوید، به سرور انتشارات ترجمک انتقال می‌یابد.

خواهشمند است به حقوق مترجمان و گروه فنی ترجمک احترام گذاشته و از توزیع بدون مجوز فایل کتاب اجتناب نمایید. شما با خرید و دانلود این کتاب موافقت نموده‌اید که اطلاعات فایل DRM به سرور انتشارات ترجمک انتقال یابد و در صورت محرز شدن نقض حقوق صاحب اثر، کلیه خسارات حاصله در طی فرآیند حقوقی و مطابق قانون حمایت حقوق مؤلفان و مصنفان و هنرمندان و ناشران جمهوری اسلامی (مصوب دوازده اسفند ۱۳۶۵ یا بعد از آن) از شما دریافت شود.

از اینکه با عرضه مقرون به صرفه کتاب‌های الکترونیک و شکوفایی انتشارات ترجمک همیاری می‌کنید، سپاسگزاریم.

**انتشارات ترجمک**

**برنامه نویسی و وب**



عنوان و نام پدیدآور	: آموزش حرفه‌ای 6 ASP.NET Core/آدم فریمن؛ ترجمه مروت گیوی.
مشخصات نشر	: همدان: ترجمک، ۱۴۰۲
مشخصات ظاهری	: جلد ۴: اجرای پروژه مثال
شابک	: دوره: ۱-۵۳-۷۸۵۵-۶۲۲-۹۷۸؛ جلد ۴: ۷-۵۱-۷۸۵۵-۶۲۲-۹۷۸
موضوع	: مایکروسافت دات‌نت فریم‌ورک، Microsoft .NET Framework، وبگاه‌ها - برنامه‌های تالیفی، Web sites -- Authoring programs، علوم کامپیوتر، Computer science، نرم‌افزار - مهندسی، Software engineering
مندرجات	: جلد ۱: ساخت وبسایت تجارت الکترونیک، ۲: کار با داده، ۳: ویژگی‌های پیشرفته
عنوان دیگر	: آموزش کاربردی 6 ASP.NET Core MVC همراه با پیاده‌سازی کامل یک پروژه با MVC و صفحات Razor
رده بندی کنگره	: QA76/76
رده بندی دیویی	: 005/276
شماره کتابشناسی ملی	: ۹۳۰۱۹۶۲

## شناسنامه کتاب



نام کتاب: آموزش حرفه‌ای 6 ASP.NET Core: جلد چهارم: اجرای پروژه مثال

ترجمه: مروت گیوی

ناشر: انتشارات ترجمک

صفحه آرایشی: انتشارات ترجمک

طراحی جلد: محمدحسین گیوی

نوبت چاپ: چاپ اول، ۱۴۰۲

قیمت: ۷۵۰۰۰ تومان

چاپ: گروه نشر الکترونیک ترجمک

شابک دوره: ۱-۵۳-۷۸۵۵-۶۲۲-۹۷۸

شابک جلد: ۷-۵۱-۷۸۵۵-۶۲۲-۹۷۸

تلفن تماس: ۰۹۱۸۱۵۰۶۱۰۰

تارنمای اینترنتی: <https://tarjomac.ir>

ISBN : 978-622-7855-53-1



ISBN : 978-622-7855-51-7



## مقدمه

کتاب آموزش حرفه‌ای ASP.NET Core 6 آدام فریمن که اکنون چاپ نهم خود را منتشر کرده است یکی از منابع اصلی و قدرتمند فراگیری توسعه وب و نوشتن اپلیکیشن‌های وب کامل و عملیاتی است.

به منظور تسهیل تهیه کتاب بر حسب سطح برنامه نویسی و نیاز هر فرد، کتاب به چهار جلد براساس بخش‌های کتاب اورجینال تقسیم و منتشر شده است. مخاطبان این کتاب برنامه نویسان و توسعه دهندگان وب است که دانش پایه تولید وبسایت و زبان C# را داشته و می‌خواهند کارهای خود را تقویت و به روز رسانی کنند.

نرم افزار ویزوال بیسیک ۲۰۲۳ و تکنولوژی دات نت ۶ در این کتاب استفاده شده و با پیروی گام به گام از فصول کتاب می‌توان اپلیکیشن وب کاملاً حرفه‌ای و قدرتمند تولید کرد. امید است خوانندگان این مجموعه کتاب چهار جلدی بتوانند وبسایت‌های مورد نظر خود را تولید کنند. ساختار کتاب به صورت زیر است:

- جلد اول: ساخت وبسایت تجارت الکترونیک
- جلد دوم: کار با داده‌ها
- جلد سوم: ویژگی‌های پیشرفته
- جلد چهارم: اجرای پروژه نمونه

## پیشگفتار

برنامه نویسان و طراحان وب حرفه ای با استفاده از راهنمایی های این کتاب پرفروش، که اکنون در نسخه نهم آن است و برای .NET 6 Core for ASP.NET به روز شده است، برنامه های نابتتری را برای پلتفرم ASP.NET Core تولید خواهند کرد. کتاب شامل توضیحات مفصلی در مورد پلتفرم ASP.NET Core و اپلیکیشن فریمورک های مورد پشتیبانی آن است.

این راهنمای بنیادی به زمینه چینی برای ASP.NET Core 6 می پردازد و ابزارها و تکنیک های مورد نیاز برای ساختن اپلیکیشن های وب مدرن و قابل توسعه را معرفی می کند. ویژگی ها و قابلیت های جدیدی مانند MVC، Razor Pages، Blazor Server و Blazor WebAssembly همراه با مثال هایی از کاربرد آنها پوشش داده شده است.

ASP.NET Core 6 آخرین تحول پلتفرم وب ASP.NET مایکروسافت است و یک فریمورک «آگونیزست میزبان» و یک مدل برنامه نویسی با بهره وری بالا ارائه می کند که معماری کد تمیزتر، توسعه مبتنی بر تست و بسط پذیری قدرتمند را ترویج می کند.

**آدام فریمن**، نویسنده این کتاب پیشروی بازار، کتاب را به طور کامل بازبینی کرده است و توضیح می دهد که چگونه از ASP.NET Core بیشترین بهره را ببرید. او با مباحث مقدماتی شروع می کند، به شما در مورد کامپوننت های میان افزار، سرویس های سرخود، درخواست پیوند مدل و موارد دیگر آموزش می دهد. با کسب دانش و اعتماد به نفس، او موضوعات پیچیده تر و ویژگی های پیشرفته تر، از جمله مسیریابی نقطه پایان و تزریق وابستگی را معرفی می کند. او آنقدر عمقی پیش می رود تا دانش مورد نیاز را به شما بدهد.

این کتاب از همان قالب و سبک نسخه های محبوب قبلی پیروی می کند، اما همه چیز را برای نسخه جدید ASP.NET Core یعنی برای NET 6 بروز می کند و تمرکز مطلب را بسط می دهد تا تمام پلتفرم ASP.NET Core را در بر بگیرد. یک مطالعه موردی کامل بصورت یک اپلیکیشن وب ASP.NET Core ارائه شده که می توانید به عنوان یک الگو برای پروژه های خود از آن استفاده کنید.

کد منبع این کتاب را می توانید در <https://github.com/Apress/pro-asp.net-core-6> بیابید.

## آنچه خواهید آموخت:

- کاوش کل پلتفرم ASP.NET Core
  - اعمال ویژگی های جدید ASP.NET Core 6 به محیط توسعه خود
  - نحوه ایجاد خدمات وب RESTful، اپلیکیشن های وب و اپلیکیشن های سمت کلاینت
  - راه اندازی سریع و موثر مدل های برنامه نویسی جدید بر پایه دانش موجود خود
- آدام فریمن یک متخصص IT با تجربه است که پست مدیر ارشد را در طیف وسیعی از شرکت ها داشته است و اخیراً به عنوان مدیر ارشد فناوری و مدیر عامل یک بانک جهانی خدمت کرده است. او که اکنون بازنشسته شده است، وقت خود را صرف نوشتن و دویدن در مسافت های طولانی می کند.

## مخاطبان این کتاب

مخاطبان این کتاب توسعه دهندگان وب با دانش اولیه توسعه وب و C# است که می خواهند آخرین پیشرفت ها و عملکردهای ASP.NET Core را در پروژه های خود بگنجانند.

## اطلاعات کتابشناختی کتاب اورجینال

**عنوان کتاب:** ASP.NET Core 6 پرو: ساخت اپلیکیشن های وب آماده کلاود با استفاده از صفحات MVC، بلیزور و ریزور

Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages

نویسنده: آدام فریمن (Adam Freeman)

کد DOI: 10.1007/978-1-4842-7957-1

ناشر: انتشارات ای پرس، برکلی، کالیفرنیا تاریخ انتشار: ۲۵ فوریه ۲۰۲۲

شابک (پرینت): 978-1-4842-7956-4 شابک (ایبوک): 978-1-4842-7957-1

نوبت چاپ: ۹ (9th edition) تعداد صفحات: XXXIII، ۱۲۵۳

## ساختار کتاب

کتاب ASP.NET Core 6 پرو: ساخت اپلیکیشن های وب آماده کلاود با استفاده از صفحات MVC، بلیزور و ریزور در چهار جلد تهیه شده است.

جلد اول کتاب همان بخش اول کتاب اورجینال می باشد که در ۱۱ فصل به مقدمات کار پرداخته است. در این بخش از پروژه مثال SportsStore استفاده شده است و مقدمات توسعه وب از قبیل آماده سازی محیط توسعه

آموزش داده شده است. ابتدا نحوه تهیه و نصب نرم افزارهای ویژوال استودیو و ویژوال استودیو کد آموزش داده شده است و طریقه نصب پکیج های میان افزار و فریمورک ها بحث شده است. سپس با استفاده از پروژه فروشگاه ورزشی به آموزش نحوه تولید صفحات، منو، سبد خرید و امنیت و انتقال اپلیکیشن پرداخته شده است.

جلد دوم کتاب با معرفی پلتفرم ASP.NET Core شروع می شود. سپس در ۶ فصل متوالی به آموزش نحوه مسیریابی، تزریق وابستگی، استفاده از ویژگی های خاص پلتفرم و نهایتاً کار با داده ها پرداخته شده است.

جلد سوم شامل فصل ۱۸ تا فصل ۳۱ است. در این جلد از کتاب که همان بخش سوم کتاب اورجینال است، ابتدا یک پروژه مثال ساخته می شود، سپس به وب سرویس ها، ویژگی های وب سرویس ها، کار با کنترلرها و نماها، استفاده از صفحات ریزور، کار با کامپوننت نما، نگ هلیپرهای سفارشی و نگ هلیپرهای سرخود، پیوند مدل، اعتبارسنجی مدل، استفاده از فیلتر و ساخت یک اپلیکیشن فرم پرداخته می شود.

جلد چهارم کتاب شامل فصل ۳۲ تا فصل ۳۹ است. در این جلد که بخش چهارم کتاب اورجینال است، ابتدا یک پروژه مثال ساخته می شود، سپس به بحث در مورد بلیزور سرور، بلیزور وب اسمبلی، ویژگی های پیشرفته بلیزور، استفاده از آیدنتیتی برای احراز هویت و تعیین مجوزها و نقش پرداخته می شود.

این کتاب شامل یک پکیج آنلاین است که کلیه کدهای مورد استفاده در کتاب، پروژه ها و رفع خطای کتاب در آن گنجانده شده است و از مخزن گیتهاب ناشر قابل دسترسی است.

آدام فریمن



## محتوای کتاب در یک نگاه

صفحه	فصل
۱	فصل ۳۲ - خلق پروژه مثال
۱۶	فصل ۳۳ - استفاده از بلیزور سرور - بخش ۱
۵۰	فصل ۳۴ - استفاده از بلیزور سرور - بخش ۲
۹۱	فصل ۳۵ - ویژگی های پیشرفته بلیزور
۱۲۹	فصل ۳۶ - فرم و داده بلیزور
۱۷۳	فصل ۳۷ - استفاده از وب اسمبلی بلیزور
۱۹۵	فصل ۳۸ - استفاده از ASP.NET CORE IDENTITY
۲۲۶	فصل ۳۹ - کاربرد ASP.NET CORE IDENTITY

## فهرست مندرجات کتاب

صفحه	عنوان
أ.....	مقدمه.....
ب.....	پیشگفتار.....
ج.....	ساختار کتاب.....
۱.....	<b>فصل ۳۲- خلق پروژه مثال.....</b>
۱.....	خلق پروژه.....
۱.....	وی استودیو.....
۲.....	افزودن پکیج های NuGet به پروژه.....
۳.....	افزودن مدل داده.....
۵.....	آماده سازی داده های اولیه.....
۶.....	پیکربندی سرویس ها و میان افزارهای Entity Framework Core.....
۷.....	خلق و اعمال مهاجرت.....
۸.....	اضافه کردن فریمورک BOOTSTRAP CSS.....
۸.....	پیکربندی سرویس ها و میان افزار.....
۹.....	ایجاد یک کنترلر و یک نما.....
۱۱.....	خلق صفحه ریزور.....
۱۴.....	اجرای اپلیکیشن مثال فصل.....
۱۴.....	خلاصه فصل.....
۱۶.....	<b>فصل ۳۳- استفاده از بلیزور سرور- بخش ۱.....</b>
۱۷.....	آماده شدن برای این فصل.....
۱۸.....	آشنایی با بلیزور سرور.....
۱۹.....	آشنایی با مزایای بلیزور سرور.....
۲۰.....	آشنایی با معایب بلیزور سرور.....
۲۰.....	انتخاب بین بلیزور سرور و Angular/React/Vue.js.....
۲۰.....	شروع به کار با بلیزور.....
۲۱.....	پیکربندی ASP.NET Core برای بلیزور سرور.....
۲۲.....	افزودن یک فایل جاوا اسکریپت بلیزور به لی اوت.....

۲۳	..... خلق فایل ایمپورت بلیزور
۲۳	..... خلق کامپوننت ریزور
۲۵	..... استفاده از کامپوننت ریزور
۲۸	..... آشنایی با پیام های اتصال بلیزور
۲۹	..... آشنایی با ویژگی های پایه کامپوننت های ریزور
۲۹	..... آشنایی با وقایع بلیزور و پیوند داده
۳۲	..... رسیدگی به وقایع چندین عنصر
۳۳	..... اجتناب از تله نام متد رسیدگی کننده
۳۵	..... پردازش وقایع بدون متد هندلر
۳۶	..... جلوگیری از وقایع پیش فرض و انتشار واقعه
۳۹	..... کار با پیوند داده
۴۲	..... خلق پیوند داده DateTime
۴۴	..... رها کردن قالب بندی تاریخ ها برای مرورگر
۴۵	..... استفاده از فایل های کلاس برای تعریف کامپوننت ها
۴۵	..... استفاده از کلاس Code-Behind
۴۷	..... تعریف کلاس کامپوننت ریزور
۴۹	..... خلاصه فصل

## فصل ۳۴- استفاده از بلیزور سرور - بخش ۲ ..... ۵۰

۵۰	..... آماده شدن برای این فصل
۵۱	..... ترکیب کامپوننت ها
۵۳	..... پیکربندی کامپوننت ها با خصیصه ها
۵۵	..... تنظیم و دریافت تنظیمات پیکربندی انبوه
۵۷	..... پیکربندی یک کامپوننت در نمای کنترلر یا صفحه ریزور
۵۹	..... ایجاد وقایع و پیوندهای سفارشی
۶۲	..... خلق پیوند مدل سفارشی
۶۵	..... نمایش محتوای دختر در یک کامپوننت
۶۵	..... محدود کردن استفاده مجدد عنصر
۶۷	..... ایجاد کامپوننت های تمپلیت
۶۹	..... استفاده از پارامترهای نوع ژنریک در کامپوننت های تمپلیت
۷۱	..... استفاده از کامپوننت تمپلیت ژنریک
۷۳	..... افزودن ویژگی ها به کامپوننت تمپلیت عمومی
۷۵	..... استفاده مجدد از کامپوننت تمپلیت عمومی
۷۷	..... پارامترهای آبشاری
۷۹	..... رسیدگی به خطاها
۷۹	..... رسیدگی به خطاهای اتصال

۸۲	رسیدگی به خطاهای شناسایی نشده اپلیکیشن
۸۵	استفاده از مرزهای خطا
۱۷	تعریف محتوای خطا درون محدوده
۸۱	بهبودی از استثناها
۹۰	خلاصه فصل

### فصل ۳۵- ویژگی های پیشرفته بلیزور ..... ۹۱

۹۲	آماده شدن برای این فصل
۹۳	استفاده از مسیریابی کامپوننت
۹۴	آماده سازی صفحه ریزور
۹۵	اضافه کردن مسیرها به کامپوننت ها
۹۷	تنظیم یک مسیر پیش فرض کامپوننت
۹۸	پیمایش بین کامپوننت های مسیریابی شده
۱۰۱	دریافت اطلاعات مسیریابی
۱۰۳	تعریف محتوای مشترک با استفاده از Layout
۱۰۴	اعمال یک لی اوت
۱۰۵	آشنایی با متدهای چرخه عمر کامپوننت
۱۰۶	آشنایی با چرخه عمر کامپوننت های مسیریابی شده
۱۰۸	استفاده از متدهای چرخه عمر برای وظایف آسنکرون
۱۱۰	مدیریت تعامل کامپوننت
۱۱۱	استفاده از رفرنس به کامپوننت های دختر
۱۱۴	تعامل با کامپوننت هایی از کد دیگر
۱۱۹	تعامل با کامپوننت ها از طریق جاوا اسکریپت
۱۱۹	فراخوانی یک تابع جاوا اسکریپت از یک کامپوننت
۱۲۲	نگهداری مرجع به عناصر HTML
۱۲۴	تحریر یک متد کامپوننت از داخل جاوا اسکریپت
۱۲۶	فراخوانی یک نمونه متد از یک تابع جاوا اسکریپت
۱۲۸	خلاصه فصل

### فصل ۳۶- فرم و داده بلیزور ..... ۱۲۹

۱۳۰	آماده شدن برای این فصل
۱۳۳	حذف دیتابیس و اجرای اپلیکیشن مثال فصل
۱۳۴	استفاده از کامپوننت های فرم بلیزور
۱۳۶	ایجاد کامپوننت های فرم سفارشی
۱۴۰	اعتبارسنجی داده های فرم
۱۴۵	رسیدگی به وقایع فرم

۱۴۷	..... استفاده از ENTITY FRAMEWORK CORE با BLAZOR
۱۴۷	..... آشنایی با مسئله حوزه زمینه Entity Framework Core
۱۴۹	..... حذف تغییرات داده ذخیره نشده
۱۵۰	..... ایجاد حوزه های تزریق وابستگی جدید
۱۵۲	..... آشنایی با مشکل کوئری تکرار شونده
۱۵۵	..... مدیریت کوئری ها در یک کامپوننت
۱۵۹	..... اجتناب از تله کوئری همپوش
۱۶۰	..... انجام عملیات خلق، خواندن، بروزرسانی و حذف
۱۶۰	..... خلق کامپوننت List
۱۶۲	..... خلق کامپوننت Details
۱۶۳	..... خلق کامپوننت Editor
۱۶۶	..... بسط ویژگی های فرم بلیزور
۱۶۷	..... ایجاد یک محدوده اعتبارسنجی سفارشی
۱۷۰	..... ایجاد یک کامپوننت دکمه سابمیت فقط معتبر
۱۷۲	..... خلاصه فصل

## فصل ۳۷- استفاده از وب اسمبلی بلیزور ..... ۱۷۳

۱۷۴	..... آماده شدن برای این فصل
۱۷۶	..... حذف دیتابیس و اجرای اپلیکیشن
۱۷۷	..... نصب کردن BLAZOR WEBASSEMBLY
۱۷۷	..... ایجاد پروژه مشترک
۱۷۸	..... خلق پروژه Blazor WebAssembly
۱۷۸	..... آماده سازی پروژه ASP.NET Core
۱۷۸	..... اضافه کردن رفرنس های راهکار
۱۷۹	..... باز کردن پروژه ها
۱۷۹	..... تکمیل پیکربندی Blazor WebAssembly
۱۸۰	..... تنظیم URL پایه
۱۸۱	..... تنظیم مسیر پایه استاتیک دارایی وب
۱۸۲	..... تست کامپوننت های Placeholder
۱۸۲	..... خلق یک کامپوننت BLAZOR WEBASSEMBLY
۱۸۲	..... ایمپورت فضای نام مدل داده
۱۸۳	..... خلق یک کامپوننت
۱۸۵	..... ناوبری در کامپوننت Blazor WebAssembly
۱۸۵	..... دریافت داده در کامپوننت Blazor WebAssembly
۱۸۸	..... خلق یک Layout
۱۸۹	..... تعریف سبک های CSS

۱۹۰	.....تکمیل اپلیکیشن فرم BLAZOR WEBASSEMBLY
۱۹۰	.....خلق کامپوننت <i>Details</i>
۱۹۱	.....خلق کامپوننت <i>Editor</i>
۱۹۴	.....خلاصه فصل

## فصل ۳۸- استفاده از ASP.NET CORE IDENTITY ..... ۱۹۵

۱۹۶	.....آماده شدن برای این فصل
۱۹۷	.....آماده سازی پروژه برای ASP.NET CORE IDENTITY
۱۹۷	.....آماده کردن دیتابیس <i>ASP.NET Core Identity</i>
۱۹۸	.....پیکربندی رشته اتصال دیتابیس
۱۹۹	.....پیکربندی اپلیکیشن
۲۰۰	.....خلق و اعمال مهاجرت دیتابیس آیدنتیتی
۲۰۰	.....ریست کردن دیتابیس ASP.NET CORE IDENTITY
۲۰۰	.....خلق کردن ابزارهای مدیریت کاربر
۲۰۱	.....اسکفلد کردن ابزارهای مدیریت آیدنتیتی
۲۰۲	.....آماده شدن برای ابزارهای مدیریت کاربر
۲۰۳	.....برشمارش حساب های کاربری
۲۰۵	.....ایجاد کاربران
۲۰۸	.....اعتبارسنجی رمزهای عبور
۲۱۱	.....اعتبارسنجی جزئیات کاربر
۲۱۳	.....ویرایش کاربران
۲۱۶	.....حذف کاربران
۲۱۷	.....ایجاد ابزارهای مدیریت نقش
۲۱۹	.....آماده شدن برای ابزارهای مدیریت نقش
۲۱۹	.....برشمارش و حذف نقش ها
۲۲۱	.....خلق نقش ها
۲۲۲	.....تخصیص عضویت نقش
۲۲۵	.....خلاصه فصل

## فصل ۳۹- کاربرد ASP.NET CORE IDENTITY ..... ۲۲۶

۲۲۶	.....آماده شدن برای این فصل
۲۲۸	.....احراز هویت کاربران
۲۲۸	.....احراز هویت در مقابل تعیین مجوز
۲۲۸	.....خلق ویژگی لاگین
۲۳۱	.....محافظت از کوکی احراز هویت
۲۳۱	.....بازرسی کوکی <i>ASP.NET Core Identity</i>

۲۳۲	..... خلق صفحه خروج از سیستم.
۲۳۳	..... تست ویژگی احراز هویت
۲۳۴	..... فعال کردن میان افزار احراز هویت.
۲۳۶	..... در نظر گرفتن احراز هویت دو عاملی (دو مرحله ای).
۲۳۸	..... دادن مجوز دسترسی به نقاط پایان
۲۳۸	..... اعمال خصیصه تعیین مجوز
۲۳۹	..... فعال کردن میان افزار تعیین مجوز.
۲۴۰	..... خلق یک نقطه پایان دسترسی ممنوع.
۲۴۱	..... ایجاد داده های اولیه (بذر).
۲۴۳	..... تست کردن توالی احراز هویت
۲۴۴	..... تغییر دادن URL های احراز هویت
۲۴۵	..... دادن مجوز دسترسی به اپلیکیشن های بلیزور
۲۴۵	..... آشنایی با OAUTH و IDENTITYSERVER
۲۴۷	..... اجرای تعیین مجوز در کامپوننت های بلیزور
۲۴۹	..... نمایش محتوا برای کاربران مجاز
۲۵۱	..... وب سرویس های احراز هویت و تعیین مجوز
۲۵۴	..... ساختن یک کلاینت ساده جاوا اسکریپت
۲۵۶	..... محدود کردن دسترسی به وب سرویس
۲۵۷	..... استفاده از احراز هویت با کوکی
۲۶۰	..... استفاده از احراز هویت با توکن حامل
۲۶۱	..... آماده سازی اپلیکیشن
۲۶۱	..... خلق توکن ها
۲۶۴	..... احراز هویت با توکن ها
۲۶۶	..... محدود کردن دسترسی با توکن ها
۲۶۷	..... استفاده از توکن ها برای درخواست داده
۲۶۹	..... خلاصه فصل







## خلق پروژه مثال

در این فصل شما یک پروژه مثال را برای استفاده در سراسر بخش ۴ کتاب خلق می کنید. این پروژه حاوی یک مدل داده است که با استفاده از کنترلرها و صفحات ریزور ساده نمایش داده می شود.

### خلق پروژه

خط فرمان پاورشل را از منوی استارت ویندوز باز کنید و فرمان ذکر شده در لیست ۳۲-۱ را اجرا کنید.

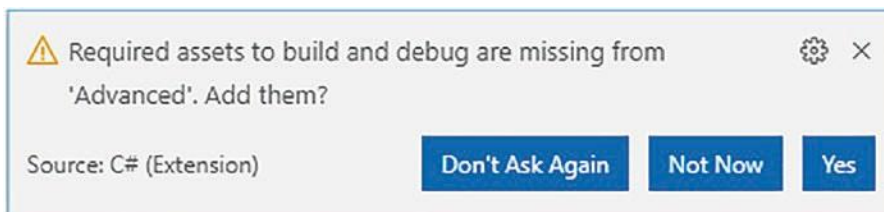
■ نکته: می توانید پروژه مثال این فصل (و سایر فصول کتاب) را از <https://github.com/apress/pro-asp.net-core> دانلود کنید. برای نحوه دریافت کمک در صورت برخورد با مشکل در اجرای مثال ها به فصل ۱ مراجعه کنید.

### لیست ۳۲-۱ خلق پروژه

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

### وی استودیو

اگر از ویژوال استودیو استفاده می کنید، فایل Advanced.sln | در پوشه Advanced باز کنید. اگر از ویژوال استودیو کد استفاده می کنید، پوشه Advanced را باز کنید. زمانی که از شما خواسته می شود که دارایی های مورد نیاز برای بیلد و دیباگ پروژه را نصب کند، دکمه Yes را کلیک کنید، همانطور که در شکل ۳۲-۱ نشان داده شده است.



شکل ۳۲-۱ اضافه کردن دارایی های (assets) به پروژه

فایل launchSettings.json را از پوشه WebApp/Properties باز کنید و پورت HTTP را تغییر داده و باز کردن مرورگر را غیرفعال کنید، همانطور که در لیست ۳۲-۲ نشان داده شده است.

**لیست ۲-۳۲** ست کردن پورت HTTP در فایل launchSettings.json از پوشه Properties

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

## افزودن پکیج‌های NuGet به پروژه

مدل داده از Entity Framework Core برای ذخیره و کوئری داده‌ها در دیتابیس SQL Server LocalDB استفاده می‌کند. برای افزودن پکیج‌های NuGet برای Entity Framework Core، از یک خط فرمان PowerShell برای اجرای فرمان‌های ذکر شده در لیست ۳-۳۲ در پوشه پروژه Advanced استفاده کنید.

### لیست ۳-۳۲ افزودن پکیج به پروژه

```
dotnet add package Microsoft.EntityFrameworkCore.Design --version 6.0.0
dotnet add package Microsoft.EntityFrameworkCore.SqlServer --version 6.0.0
```

اگر از مثال‌های فصل‌های قبلی پیروی نکرده‌اید، باید بسته ابزار گلوبال (همگانی) را که برای ایجاد و مدیریت مهاجرت‌های Entity Framework Core استفاده می‌شود، نصب کنید. فرامین قید شده در لیست ۳-۳۲ را اجرا کنید تا هر نسخه موجود از پکیج را حذف کرده و نسخه مورد نیاز این کتاب را نصب کنید.

## لیست ۴-۳۲ نصب پکیج گلوبال ابزار

```
dotnet tool uninstall --global dotnet-ef
dotnet tool install --global dotnet-ef --version 6.0.0
```

## افزودن مدل داده

مدل داده<sup>۱</sup> برای این اپلیکیشن شامل سه کلاس است که نمایانگر کارکنان، بخشی که در آن کار می کنند و مکان آنها است. یک پوشه Models ایجاد کنید و یک فایل کلاس به نام Person.cs با کد موجود در لیست ۵-۳۲ به آن اضافه کنید.

### لیست ۵-۳۲ محتوای فایل Person.cs از پوشه Models

```
namespace Advanced.Models
{
    public class Person
    {
        public long PersonId { get; set; }
        public string Firstname { get; set; } = String.Empty;
        public string Surname { get; set; } = String.Empty;
        public long DepartmentId { get; set; }
        public long LocationId { get; set; }
        public Department? Department { get; set; }
        public Location? Location { get; set; }
    }
}
```

یک فایل کلاس به نام Department.cs را به پوشه Models اضافه کنید و از آن برای تعریف کلاس نشان داده شده در لیست ۶-۳۲ استفاده کنید.

### لیست ۶-۳۲ محتوای فایل Department.cs از پوشه Models

```
namespace Advanced.Models
{
    public class Department
    {
        public long Departmentid { get; set; }
        public string Name { get; set; } = String.Empty;
        public IEnumerable<Person>? People { get; set; }
    }
}
```

<sup>1</sup> Data Model

یک فایل کلاس به نام Location.cs را به پوشه Models اضافه کنید و از آن برای تعریف کلاس نشان داده شده در لیست ۷-۳۲ استفاده کنید.

### لیست ۷-۳۲ محتوای فایل Location.cs از پوشه Models

```
namespace Advanced.Models
{
    public class Location
    {
        public long LocationId { get; set; }
        public string City { get; set; } = string.Empty;
        public string State { get; set; } = String.Empty;

        public IEnumerable<Person>? People { get; set; }
    }
}
```

هر یک از سه کلاس مدل داده یک ویژگی کلیدی را تعریف می‌کند که مقدار آن توسط دیتابیس زمانی که اشیاء جدید ذخیره می‌شوند، تخصیص داده می‌شود و ویژگی کلید خارجی را تعریف می‌کند که روابط بین کلاس‌ها را تعریف می‌کند. این ویژگی‌ها با ویژگی‌های ناوبری تکمیل می‌شوند که با متد Entity Framework Core Include برای ترکیب داده‌های مرتبط در کوئری‌ها استفاده می‌شوند.

برای ایجاد کلاس زمینه برای Entity Framework Core که دسترسی به دیتابیس را فراهم می‌کند، فایلی به نام DataContext.cs را به پوشه Models اضافه کنید و کد نشان داده شده در لیست ۸-۳۲ را اضافه کنید.

### لیست ۸-۳۲ محتوای فایل DataContext.cs از پوشه Models

```
using Microsoft.EntityFrameworkCore;

namespace Advanced.Models {
    public class DataContext : DbContext {

        public DataContext(DbContextOptions<DataContext> opts)
            : base(opts) { }

        public DbSet<Person> People => Set<Person>();
        public DbSet<Department> Departments => Set<Department>();
        public DbSet<Location> Locations => Set<Location>();
    }
}
```

کلاس context ویژگی‌هایی را تعریف می‌کند که برای کوئری از دیتابیس برای داده‌های Department، Person و Location استفاده می‌شود.

## آماده سازی داده های اولیه

یک کلاس به نام SeedData.cs را به پوشه Models اضافه کنید و کد نشان داده شده در لیست ۳۲-۹ را اضافه کنید تا داده های اولیه (داده های موقت یا سید) را که برای پر کردن دیتابیس استفاده می شود، تعریف کنید.

### فهرست ۳۲-۹ محتوای فایل SeedData.cs از پوشه Models

```
using Microsoft.EntityFrameworkCore;

namespace Advanced.Models {
    public static class SeedData {

        public static void SeedDatabase(DataContext context) {
            context.Database.Migrate();
            if (context.People.Count() == 0 && context.Departments.Count() == 0
&&
                context.Locations.Count() == 0) {

                Department d1 = new Department { Name = "Sales" };
                Department d2 = new Department { Name = "Development" };
                Department d3 = new Department { Name = "Support" };
                Department d4 = new Department { Name = "Facilities" };

                context.Departments.AddRange(d1, d2, d3, d4);
                context.SaveChanges();

                Location l1 = new Location { City = "Oakland", State = "CA" };
                Location l2 = new Location { City = "San Jose", State = "CA" };
                Location l3 = new Location { City = "New York", State = "NY" };
                context.Locations.AddRange(l1, l2, l3);

                context.People.AddRange(
                    new Person {
                        Firstname = "Francesca", Surname = "Jacobs",
                        Department = d2, Location = l1
                    },
                    new Person {
                        Firstname = "Charles", Surname = "Fuentes",
                        Department = d2, Location = l3
                    },
                    new Person {
                        Firstname = "Bright", Surname = "Becker",
                        Department = d4, Location = l1
                    },
                    new Person {
                        Firstname = "Murphy", Surname = "Lara",
                        Department = d1, Location = l3
                    },
                    new Person {
                        Firstname = "Beasley", Surname = "Hoffman",
```



```
app.MapGet("/", () => "Hello World!");
var context = app.Services.CreateScope().ServiceProvider
    .GetRequiredService<DataContext>();
SeedData.SeedDatabase(context);

app.Run();
```

برای تعریف رشته اتصال که برای داده های اپلیکیشن استفاده می شود، تنظیمات پیکربندی نشان داده شده در لیست ۳۲-۱۱ را در فایل appsettings.json اضافه کنید. رشته اتصال باید در یک خط وارد شود.

**لیست ۳۲-۱۱** تعریف رشته اتصال در فایل appsettings.json از پوشه Advanced

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning",
      "Microsoft.EntityFrameworkCore": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "PeopleConnection":
    "Server=(localdb)\\MSSQLLocalDB;Database=People;MultipleActive ResultSets=True"
  }
}
```

علاوه بر رشته اتصال، لیست ۳۲-۱۰ جزئیات یادداشت برداری را برای Entity Framework Core افزایش می دهد تا کوئری های SQL ارسال شده به دیتابیس ثبت شوند.

## خلق و اعمال مهاجرت

برای ایجاد مهاجرتی که طرحواره دیتابیس را تنظیم می کند، از یک خط فرمان PowerShell برای اجرای فرمان ذکر شده در لیست ۳۲-۱۲ در پوشه پروژه Advanced استفاده کنید.

**لیست ۳۲-۱۲** خلق یک مهاجرت Entity Framework Core

---

```
dotnet ef migrations add Initial
```

---

پس از خلق مهاجرت، با استفاده از فرمان قید شده در لیست ۳۲-۱۳، آن را به دیتابیس اعمال کنید.

**لیست ۳۲-۱۳** اعمال مهاجرت به دیتابیس

---

```
dotnet ef database update
```

---

پیام های گزارش گیری که توسط برنامه نمایش داده می شود، دستورات SQL را نشان می دهد که به دیتابیس ارسال می شوند.

■ توجه داشته باشید اگر نیاز به ریست دیتابیس دارید، دستور `dotnet ef database drop --force` و سپس دستور موجود در لیست ۳۲-۱۳ را اجرا کنید.

## اضافه کردن فریمورک Bootstrap CSS

به دنبال الگوی ایجاد شده در فصول قبلی، من از چارچوب Bootstrap CSS برای استایل دادن به عناصر HTML تولید شده توسط اپلیکیشن مثال استفاده خواهیم کرد. برای نصب بسته بوت استرپ، فرمان‌های ذکر شده در لیست ۳۲-۱۴ را در پوشه پروژه Advanced اجرا کنید. اجرای این دستورات به پکیج Library Manager متکی است.

### لیست ۳۲-۱۴ نصب فریمورک Bootstrap CSS

```
libman init -p cdnjs
libman install bootstrap@5.1.3 -d wwwroot/lib/bootstrap
```

اگر از ویژوال استودیو استفاده می‌کنید، می‌توانید بسته‌های سمت کلاینت را با کلیک راست روی آیتیم پروژه Advanced در Solution Explorer و انتخاب Client-Side Library ► Add از منوی پاپ آپ نصب کنید.

## پیکربندی سرویس‌ها و میان افزار

اپلیکیشن مثال در این قسمت از کتاب به درخواست‌ها با استفاده از کنترلرهای MVC و صفحات ریزور پاسخ می‌دهد. برای پیکربندی سرویس‌ها و میان افزارهایی که اپلیکیشن استفاده می‌کند، دستورات نشان داده شده در لیست ۳۲-۱۵ را به فایل Program.cs اضافه کنید.

### لیست ۳۲-۱۵ افزودن سرویس‌ها و میان‌افزار به فایل Program.cs از پوشه Advanced

```
using Microsoft.EntityFrameworkCore;
using Advanced.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();
builder.Services.AddRazorPages();
builder.Services.AddDbContext<DataContext>(opts =>
{
    opts.UseSqlServer(builder.Configuration["ConnectionStrings:PeopleConnection"]);
});
builder.Services.EnableSensitiveDataLogging(true);
});

var app = builder.Build();
//app.MapGet("/", () => "Hello World!"); app.UseStaticFiles();
app.MapControllers();
app.MapControllerRoute("controllers",
"controllers/{controller=Home}/{action=Index}/{id?}");
app.MapRazorPages();
```



```
var context = app.Services.CreateScope().ServiceProvider
    .GetRequiredService<DataContext>();
SeedData.SeedDatabase(context);
app.Run();
```

علاوه بر نگاشت مسیر کنترلرها، من مسیری اضافه کرده‌ام که با مسیرهای URL که با کنترلرها شروع می‌شوند مطابقت دارد، که پیروی از مثال‌های فصول بعدی را آسان‌تر می‌کند، زیرا آنها بین کنترلرها و صفحات ریزور جابجا می‌شوند. این همان قراردادی است که در فصل‌های قبلی اتخاذ کردم، و مسیرهای URL را که با /pages شروع می‌شوند به صفحات ریزور هدایت می‌کنم.

## ایجاد یک کنترلر و یک نما

برای نمایش داده‌های برنامه با استفاده از یک کنترلر، پوشه‌ای به نام Controllers در پوشه پروژه Advanced ایجاد کنید و یک فایل کلاس به نام HomeController.cs با محتوای نشان داده شده در لیست ۱۶-۳۲ به آن اضافه کنید.

### لیست ۱۶-۳۲ محتوای فایل HomeController.cs از پوشه Controllers

```
using Advanced.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace Advanced.Controllers
{
    public class HomeController : Controller
    {
        private DataContext context;
        public HomeController(DataContext dbContext)
        {
            context = dbContext;
        }
        public IActionResult Index([FromQuery] string selectedCity)
        {
            return View(new PeopleListViewModel
            {
                People = context.People
                .Include(p => p.Department).Include(p => p.Location),
                Cities = context.Locations.Select(l => l.City).Distinct(),
                SelectedCity = selectedCity
            });
        }
    }
    public class PeopleListViewModel
    {
        public IEnumerable<Person> People { get; set; } =
        Enumerable.Empty<Person>();
        public IEnumerable<string> Cities { get; set; } =
        Enumerable.Empty<string>();
        public string SelectedCity { get; set; } = String.Empty;
    }
}
```

```

        public string GetClass(string? city) =>
            SelectedCity == city ? "bg-info text-white" : "";
    }
}

```

برای فراهم کردن نما برای کنترلر، پوشه Views/Home را ایجاد کنید و یک نمای ریزور به نام Index.cshtml با محتوای نشان داده شده در لیست ۱۷-۳۲ به آن اضافه کنید.

### لیست ۱۷-۳۲ محتوای فایل Index.cshtml از پوشه Views/Home

```

@model PeopleListViewModel

<h4 class="bg-primary text-white text-center p-2">People</h4>

<table class="table table-sm table-bordered table-striped">
    <thead>
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Dept</th>
            <th>Location</th>
        </tr>
    </thead>
    <tbody>
        @foreach (Person p in Model?.People ?? Enumerable.Empty<Person>())
        {
            <tr class="@Model?.GetClass(p.Location?.City)">
                <td>@p.PersonId</td>
                <td>@p.Surname, @p.Firstname</td>
                <td>@p.Department?.Name</td>
                <td>@p.Location?.City, @p.Location?.State</td>
            </tr>
        }
    </tbody>
</table>

<form asp-action="Index" method="get">
    <div class="form-group">
        <label for="selectedCity">City</label>
        <select name="selectedCity" class="form-control">
            <option disabled selected>Select City</option>
            @foreach (string city in Model?.Cities ??
Enumerable.Empty<string>())
            {
                <option selected="@ (city == Model?.SelectedCity)"> @city
            </option>
            }
        </select>
    </div>
    <button class="btn btn-primary mt-2" type="submit">Select</button>

```

```
</form>
```

برای فعال کردن تگ هلیپرها و افزودن فضاهای نامی که به‌طور پیش‌فرض در نماها در دسترس خواهند بود، یک فایل ایمپورت نمای ریزور با نام `_ViewImports.cshtml` را با محتوای نشان‌داده‌شده در فهرست ۳۲-۱۸ به پوشه Views اضافه کنید.

### لیست ۳۲-۱۸ محتوای فایل `_ViewImports.cshtml` از پوشه Views

```
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@using Advanced.Models
@using Advanced.Controllers
```

برای تعیین لی‌اوت پیش‌فرض برای نماهای کنترلر، یک فایل استارت نمای ریزور به نام `_ViewStart.cshtml` با محتوای نشان‌داده‌شده در لیست ۳۲-۱۹ به پوشه Views اضافه کنید.

### لیست ۳۲-۱۹ محتوای فایل `_ViewStart.cshtml` از پوشه Views

```
{
    Layout = "_Layout";
}
```

برای خلق لی‌اوت یا چیدمان، پوشه `Views/Shared` را ایجاد کنید و یک لی‌اوت ریزور به نام `_Layout.cshtml` با محتوای نشان‌داده‌شده در لیست ۳۲-۲۰ به آن اضافه کنید.

### لیست ۳۲-۲۰ محتوای فایل `_Layout.cshtml` از پوشه `Views/Shared`

```
<!DOCTYPE html>
<html>

<head>
    <title>@ViewBag.Title</title>
    <link href="/lib/bootstrap/css/bootstrap.min.css" rel="stylesheet" />
</head>

<body>
    <div class="m-2"> @RenderBody()
    </div>
</body>

</html>
```

## خلق صفحه ریزور

برای نمایش داده‌های برنامه با استفاده از صفحه ریزور، پوشه `Pages` را ایجاد کنید و یک صفحه ریزور به نام `Index.cshtml` با محتوای نشان‌داده‌شده در لیست ۳۲-۲۱ به آن اضافه کنید.

### لیست ۳۲-۲۱ محتوای فایل `Index.cshtml` از پوشه Pages

```
@page "/pages"
```

```

@model IndexModel
<h4 class="bg-primary text-white text-center p-2">People</h4>
<table class="table table-sm table-bordered table-striped">
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Dept</th>
      <th>Location</th>
    </tr>
  </thead>
  <tbody>
    @foreach (Person p in Model.People)
    {
      <tr class="@Model.GetClass(p.Location?.City)">
        <td>@p.PersonId</td>
        <td>@p.Surname, @p.Firstname</td>
        <td>@p.Department?.Name</td>
        <td>@p.Location?.City, @p.Location?.State</td>
      </tr>
    }
  </tbody>
</table>

<form asp-page="Index" method="get">
  <div class="form-group">
    <label for="selectedCity">City</label>
    <select name="selectedCity" class="form-control">
      <option disabled selected>Select City</option>
      @foreach (string city in Model.Cities)
      {
        <option selected="@ (city == Model.SelectedCity)"> @city
      </option>
      }
    </select>
  </div>
  <button class="btn btn-primary mt-2" type="submit">Select</button>
</form>
@functions {
public class IndexModel : PageModel
{
  private DataContext context;
  public IndexModel (DataContext dbContext)
  {
    context = dbContext;
  }
  public IEnumerable<Person> People { get; set; } =
  Enumerable.Empty<Person> ();
  public IEnumerable<string> Cities { get; set; } =
  Enumerable.Empty<string> ();
  [FromQuery]

```

```

public string SelectedCity { get; set; } = String.Empty;
public void OnGet()
{
    People = context.People.Include(p => p.Department)
        .Include(p => p.Location);
    Cities = context.Locations.Select(l => l.City).Distinct();
}
public string GetClass(string? city) =>
    SelectedCity == city ? "bg-info text-white" : "";
}
}

```

برای فعال کردن تگ هلیپرها و افزودن فضاهای نامی که به‌طور پیش‌فرض در بخش نمای صفحات ریزور در دسترس خواهند بود، یک فایل ایمپورت نمای ریزور به نام `_ViewImports.cshtml` را به پوشه Pages با محتوای نشان داده شده در فهرست ۲۲-۳۲ اضافه کنید.

#### لیست ۲۲-۳۲ محتوای فایل `_ViewImports.cshtml` از پوشه Pages

```

@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@using Advanced.Models
@using Microsoft.AspNetCore.Mvc.RazorPages
@using Microsoft.EntityFrameworkCore

```

برای تعیین چیدمان پیش‌فرض برای صفحات ریزور، یک فایل استارت نمای ریزور به نام `_ViewStart.cshtml` را به پوشه Pages با محتوای نشان داده شده در فهرست ۲۳-۳۲ اضافه کنید.

#### لیست ۲۳-۳۲ محتوای فایل `_ViewStart.cshtml` از پوشه Pages

```

@{
    Layout = "_Layout";
}

```

برای خلق یک لی‌اوت یا چیدمان، یک فایل لی‌اوت ریزور به نام `_Layout.cshtml` را به پوشه Pages با محتوای نشان داده شده در فهرست ۲۴-۳۲ اضافه کنید.

#### لیست ۲۴-۳۲ محتوای فایل `_Layout.cshtml` از پوشه Pages

```

<!DOCTYPE html>
<html>
<head>
    <title>@ViewBag.Title</title>
    <link href="/lib/bootstrap/css/bootstrap.min.css" rel="stylesheet" />
</head>
<body>
    <div class="m-2">
        <h5 class="bg-secondary text-white text-center p-2">Razor Page</h5>
    @RenderBody()
    </div>
</body>

```

</html>

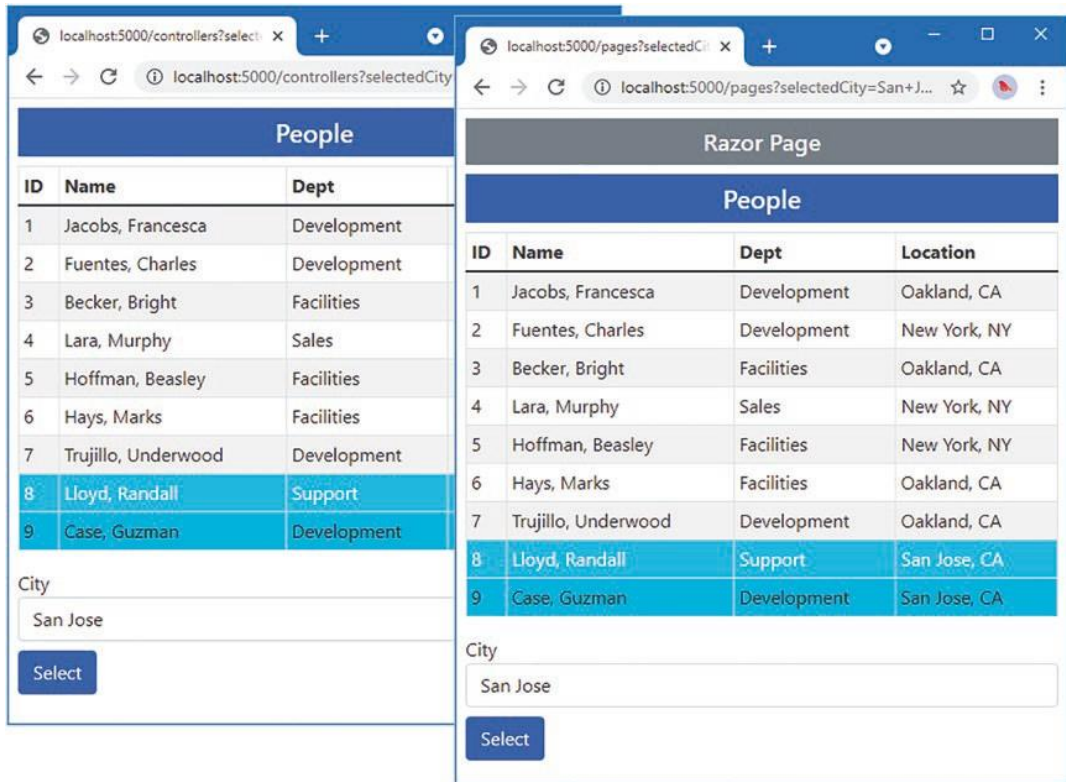
## اجرای اپلیکیشن مثال فصل

اپلیکیشن را با اجرای فرمان قید شده در لیست ۳۲-۲۵ در پوشه پروژه Advanced شروع کنید.

### لیست ۳۲-۲۵ اجرای اپلیکیشن مثال فصل

```
dotnet run
```

از مرورگر برای درخواست <http://localhost:5000/controllers> و <http://localhost:5000/pages> استفاده کنید. همانطور که در شکل ۳۲-۲ نشان داده شده است، یک شهر را با استفاده از عنصر select انتخاب کنید و روی دکمه Select کلیک کنید تا ردیف‌های جدول هایلایت شوند.



شکل ۳۲-۲۵ اجرای اپلیکیشن مثال فصل

## خلاصه فصل

در این فصل، من نشان دادم که چگونه می‌توان یک اپلیکیشن نمونه را ایجاد کرد که در این قسمت از کتاب استفاده می‌شود. این پروژه با تمپلیت Empty ایجاد شده است و شامل یک مدل داده است که به Entity Framework

Core متکی است و درخواست ها را با استفاده از یک کنترلر و یک صفحه ریزور رسیدگی می کند. در فصل بعد Blazor را معرفی می کنم.