آدام فریمن

آموزش حرفه ای
ASP.NET Core 6

جلد چهارم:

اجرای پروژه مثال

مروّت گیوی

ترجمک
arjomac

آدام فریمن

آموزش حرفه‌ای ASP.NET Core 6

جلد چهارم: اجرای پروژه مثال

ترجمه

مروّت گیوی

انتشارات ترجمک

تابستان ۱۴۰۲

آموزش حرفه‌ای ASP.NET CORE 6

توسعه وب با ASP.NET Core

توجه :

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

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 نموده اید 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 کتاب های الکترونیک و شکوفایی انتشارات ترجمک همیاری می کنید، سپاسگزاریم.

انتشارات ترجمک

برنامه نویسی و وب

عنوان و نام پدیدآور	:	آموزش حرفه‌ای ASP.NET Core 6/آدام 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	:	جلد۱: ساخت وبسایت تجارت الکترونیک، ۲: کار با داده، ۳: ویژگی های پیشرفته
عنوان دیگر	:	آموزش کاربردی Pro ASP.NET Core MVC 6 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 ASP.NET Core 6: جلد چهارم: اجرای پروژه مثال

ترجمه: مروت گیوی

ناشر: انتشارات ترجمک

صفحه آرایی: 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 اپلیکیشن های وب کامل و عملیاتی است.

به منظور تسهیل تهیه کتاب بر حسب سطح برنامه نویسی و نیاز هر فرد، کتاب به چهار جلد براساس بخش های 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 می خواهند کارهای خود را تقویت و به روز رسانی کنند.

نرم افزار ویژوال بیسیک ۲۰۲۳ و تکنولوژی دات نت ۶ در این کتاب استفاده شده و با پیروی گام به گام از فصول کتاب می توان اپلیکیشن وب کاملا حرفه ای و قدرتمند تولید کرد. امید است خوانندگان این مجموعه کتاب چهار جلدی بتوانند وبسایت های مورد نظر خود را تولید کنند. ساختار کتاب به صورت زیر است:

- جلد اول: ساخت وبسایت تجارت الکترونیک
- جلد دوم: کار با داده ها
- جلد سوم: ویژگی های پیشرفته
- جلد چهارم: اجرای پروژه نمونه

مترجم

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 ASP.NET Core for .NET 6 به روز شده است، برنامه های ناب‌تری را برای پلتفرم ASP.NET Core تولید خواهند کرد. کتاب شامل توضیحات مفصلی در مورد پلتفرم ASP.NET Core و اپلیکیشن فریمورک‌های 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.

ASPNET Core 6 آخرین تحول پلتفرم وب ASP.NET مایکروسافت است و یک فریمورک «آگونیست میزبان» و یک مدل برنامه‌نویسی با بهره‌وری بالا ارائه می‌کند که معماری کد تمیزتر، توسعه مبتنی بر تست و بسط پذیری قدرتمند را ترویج می‌کند.

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 می‌کند، به شما در مورد کامپوننت های میان‌افزار، سرویس‌های سرخود، درخواست پیوند مدل و موارد دیگر آموزش می‌دهد. 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 نسخه‌های محبوب قبلی پیروی می‌کند، اما همه چیز را برای نسخه جدید ASP.NET Core یعنی برای NET 6 بروز می‌کند و تمرکز مطلب را بسط می‌دهد 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 می‌توانید 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 شرکت‌ها 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 ۶ 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 نهایتا 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، تگ هلپرهای سفارشی و تگ هلپرهای 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

فصل	صفحه

فهرست مندرجات کتاب

عنوان	صفحه

فصل ۳۲

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.

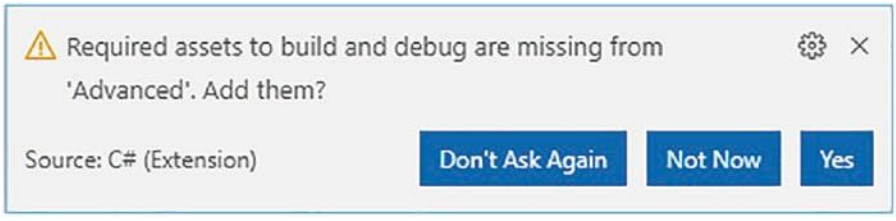
■ **نکته**: می‌توانید پروژه مثال این فصل (و سایر فصول کتاب) را از https://github.com/apress/pro-asp.net-core-6 دانلود کنید. برای نحوه دریافت کمک در صورت برخورد با مشکل در اجرای مثال ها به فصل ۱ مراجعه کنید.

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 ا 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 ۳۲-۲ 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 پکیج های NuGet به پروژه

مدل داده از Entity Framework Core برای ذخیره و کوئری داده ها در دیتابیس SQL Server LocalDB استفاده می کند. برای افزودن پکیج های NuGet برای Entity Framework Core، از یک خط فرمان PowerShell برای اجرای فرمان های ذکر شده در لیست ۳۲-۳ در پوشه پروژه Advanced استفاده کنید.

لیست ۳۲-۳ 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 ۳۲-۴ را اجرا کنید تا هر نسخه موجود از پکیج را حذف کرده و نسخه مورد نیاز این کتاب را نصب کنید.

لیست ۳۲-۴ 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[1] 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 ۳۲-۵ به آن اضافه کنید.

لیست ۳۲-۵ 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 ۳۲-۶ استفاده کنید.

لیست ۳۲-۶ 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; }
    }
}
```

[1] 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 ۳۲-۷ استفاده کنید.

لیست ۳۲-۷ 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 می کند که مقدار آن توسط دیتابیس زمانی که اشیاء جدید ذخیره می شوند، تخصیص داده می شود و ویژگی کلید خارجی را تعریف می کند که روابط بین کلاس‌ها را تعریف می کند. این ویژگی‌ها با ویژگی‌های ناوبری تکمیل می‌شوند که با متد Entity Framework Core Include برای ترکیب داده‌های مرتبط در کوئری ها استفاده می‌شوند.

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

لیست ۳۲-۸ 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 ویژگی هایی را تعریف می کند که برای کوئری از دیتابیس برای داده های Person، Department و Location استفاده می شود.

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",
```

```
                    Department = d4, Location = l3
                },
                new Person {
                    Firstname = "Marks", Surname = "Hays",
                    Department = d4, Location = l1
                },
                new Person {
                    Firstname = "Underwood", Surname = "Trujillo",
                    Department = d2, Location = l1
                },
                new Person {
                    Firstname = "Randall", Surname = "Lloyd",
                    Department = d3, Location = l2
                },
                new Person {
                    Firstname = "Guzman", Surname = "Case",
                    Department = d2, Location = l2
                });
            context.SaveChanges();
        }
    }
}
}
```

متد استاتیک SeedDatabase تضمین می کند که همه مهاجرت های معلق به دیتابیس اعمال شده است. اگر دیتابیس خالی باشد، داده های اولیه در آن بارگذاری می شود. Entity Framework Core نگاشت اشیاء در جداول دیتابیس را انجام می دهد و هنگامی که داده ها ذخیره می شوند، ویژگی های کلیدی به طور خودکار اختصاص داده می شود.

پیکربندی سرویس ها و میان افزارهای Entity Framework Core

تغییرات نشان داده شده در لیست ۳۲-۱۰ را در فایل Program.cs اعمال کنید، که Entity Framework Core را پیکربندی می کند و سرویس های DataContext را تنظیم می کند که در این قسمت از کتاب برای دسترسی به دیتابیس استفاده می شود.

لیست ۳۲-۱۰ آماده سازی سرویس ها و میان افزارها در فایل Program.cs از پوشه Advanced

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

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddDbContext<DataContext>(opts =>
{
    opts.UseSqlServer(builder.Configuration["ConnectionStrings:PeopleConnection"]
);
    opts.EnableSensitiveDataLogging(true);
});
var app = builder.Build();
```

```
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 یاداشت برداری را برای Entity Framework Core افزایش می‌دهد 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
```

پیام‌های گزارش‌گیری که توسط برنامه نمایش داده می‌شود، دستورات SQL را نشان می‌دهد که به دیتابیس ارسال می‌شوند.

■ 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 CSS Bootstrap برای استایل دادن به عناصر HTML تولید شده توسط اپلیکیشن مثال استفاده خواهم کرد. برای نصب بسته بوت استرپ، فرمان های 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 آیتم پروژه Advanced در Solution Explorer و انتخاب Client-Side Library ➤ Add از منوی پاپ آپ نصب کنید.

پیکربندی سرویس ها و میان افزار

اپلیکیشن مثال در این قسمت از کتاب به درخواست ها با استفاده از کنترلرهای MVC و صفحات ریزور پاسخ می دهد. برای پیکربندی سرویس ها و میان افزارهایی که اپلیکیشن استفاده می کند، 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"]
);
    opts.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 داده های برنامه با استفاده از یک کنترلر، پوشه ای به نام Controllers در پوشه پروژه Advanced ایجاد کنید و یک فایل کلاس به نام HomeController.cs با محتوای نشان داده شده در لیست ۳۲-۱۶ به آن اضافه کنید.

لیست ۳۲-۱۶ 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 ۳۲-۱۷ به آن اضافه کنید.

لیست ۳۲-۱۷ 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 هلپرها 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 نشان داده شده 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 نشان داده شده 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 داده های برنامه با استفاده از صفحه ریزور، پوشه Pages را ایجاد کنید و یک صفحه ریزور به نام Index.cshtml با محتوای نشان داده شده 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 هلپرها 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 ۳۲-۲۲ اضافه کنید.

لیست ۳۲-۲۲ 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 ۳۲-۲۳ اضافه کنید.

لیست ۳۲-۲۳ 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 ۳۲-۲۴ اضافه کنید.

لیست ۳۲-۲۴ 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>
```

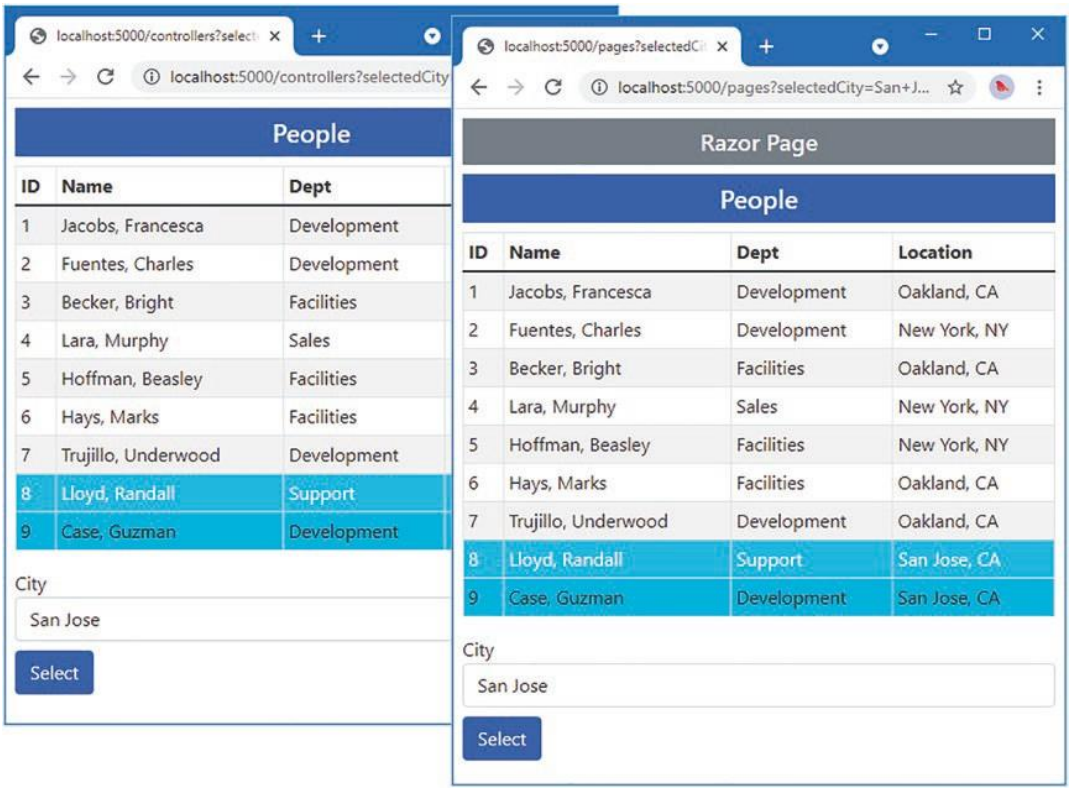
اجرای اپلیکیشن مثال فصل

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 ردیف های جدول هایلایت شوند.



شکل ۳۲-۲ اجرای اپلیکیشن مثال فصل

خلاصه فصل

در این فصل، من نشان دادم که چگونه می توان یک اپلیکیشن نمونه را ایجاد کرد که در این قسمت از کتاب استفاده می شود. 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.