استفاده از
Entity Framework 6 Code First
در
ASP.NET MVC 5

آموزش گام به گام
همراه با یک پروژه عملی

Tom Dykstra, Rick Anderson

ترجمه: مهندس محمد محمدی پیروز

انتشارات پندار پارس

فیپا

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ **www.pendarepars.com**

تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸ همراه: ۰۹۲۱۴۳۷۱۹۶۴ **info@pendarepars.com**

نام کتاب	: **استفاده از Entity Framework 6 Code First در MVC 5**	
ناشر	: **انتشارات پندار پارس**	
تالیف	: Tom Dykstra, Rick Anderson	
ترجمه	: **محمد محمدی پیروز**	
چاپ نخست	: **آبان ماه ۹۴**	
شمارگان	: **۵۰۰ نسخه**	
طرح جلد و صفحه‌آرایی	: **سارا یعسوبی**	
چاپ، صحافی	: **روز**	
قیمت	: **۲۱۰۰۰ تومان**	**شابک : ۳-۹۴-۶۵۲۹-۶۰۰-۹۷۸**

تقدیم به پدر و مادر مهربانم
و نیز به همسر وفادار و فرزند برومندم

"دیروز رویا است و فردا تنها یک خیال

اما خوب زندگی کردن در امروز

هر دیروزی را رویایی از شادمانی

و هر فردایی را خیالی از امید می‌سازد.

از این رو، به خوبی به امروز بنگرید."

یکی از اشعار سنتی هند

پیش‌گفتار

این روزها نقش وب به عنوان بستری با اهمیت برای انجام فعالیت‌های آنلاین، مانند به اشتراک گذاری منابع، تجارت، حضور در شبکه‌های احتماعی و بسیاری اعمال دیگر بر هیچ کس پوشیده نیست. مایکروسافت، یکی از بزرگ‌ترین و نام آورترین شرکت‌ها در تولید و ایجاد بسترهای مورد نیاز برای توسعه‌گران، با ارائه‌ی ابزارها و بسترهای لازم برای تولید برنامه‌های مبتنی بر وب است.

ASP.NET MVC یکی از جدیدترین چهارچوب‌هایی است که با توجه کامل به امکان استفاده از جدیدترین استانداردهای وب، و قابلیت سفارشی سازی و مفاهیم واقعی پروتکل HTTP، توسط مایکروسافت عرضه شده است و هر روز پیش از پیش مورد توجه قرار می‌گیرد.

تمرکز اصلی این کتاب بر نحوه به کارگیری Entity Framework در برنامه‌های تحت وب تولید شده با چهارچوب ASP.NET MVC است. بر همین اساس خوانندگان محترم می‌بایست آشنایی مختصری با ASP.NET MVC داشته باشند[1].

Entity Framework یکی از قدرتمندترین Object-Relational Mapping های دسترسی به منابع داده است. به وسیله Entity Framework، بدون اینکه نیاز به عملیات مستقیم در پایگاه داشته باشید، می‌توانید با انواع مختلف پایگاه داده همچون SQL Server، Sqlite و ... کار کنید. این تکنولوژی، چند ویژگی مهم دارد که توجه بسیاری از برنامه نویسان را به خود جلب کرده است:

کار کردن با Entity Framework بسیار ساده است. حتی یادگیری آن ده‌ها برابر ساده‌تر از مدل‌های مشابه مانند ADO.NET است.
سرعت آن در دسترسی به داده‌ها و اجرای شیوه‌نامه‌ها بسیار بالاست و در پروژه‌های بزرگ نیز می‌توان از آن استفاده کرد.
پیاده‌سازی آن بسیار سریع است و در زمان صرفه جویی می‌شود.

---

[1]- با توجه به اینکه کتاب بیشتر به تشریح امکانات Entity Framework برای کار با پایگاه داده در چهارچوب برنامه‌های تحت وب تولید شده به وسیله ASP.NET MVC پرداخته است، به شما خواننده‌ی گرامی پیشنهاد می‌کنم پیش از مطالعه این کتاب، کتاب مرجع کامل ASP.NET MVC 5.2 (یا نسخه MVC 4 آن) نوشته آقای بهروز راد، از همین ناشر، را مطالعه نمایید[مترجم].

پایگاه داده‌ی شما در قالب کلاس‌های شئ‌گرا، با دیگر اجزای پروژه یکپارچه می‌شود و افزون بر خوانایی بالا در کدنویسی، عملیات ارتقا و به‌روزرسانی نرم‌افزاری را سرعت می‌بخشد و از بروز بسیاری از خطاها جلوگیری می‌کند. در برخی از حالت‌ها نیاز به طراحی پایگاه داده ندارید! زیرا با استفاده از امکانات EF خود برنامه، پایگاه داده را از روی مدل یا کلاس‌های شما می‌سازد.

در واقع Entity Framework تکنولوژی توسعه یافته ADO.Net است که فاصله بین برنامه نویسی شئ‌گرایی و بانک اطلاعاتی رابطه‌ای را پر می کند. این فاصله معمولا با نام عدم تطابق شناخته می‌شود و یک تکنیک برنامه نویسی برای تبدیل ارتباطات در دیتابیس به مفاهیم Object Oriented در برنامه نویسی است. در واقع می‌توان گفت که کلاس‌ها را به Tableها مَپ می‌کند. وقتی می‌خواهید به پایگاه داده دسترسی پیدا کنید، یا اطلاعاتی را ذخیره کنید، این کارها را مستقیما بر روی اشیاء (آبجکت‌تان) انجام می‌دهید.

در Entity Framework سه نوع مدل‌سازی وجود دارد:

۱. Database First Modeling: در این روش مدل ما از روی یک پایگاه داده موجود ایجاد می‌شود و می‌توان از طریق Entity Data Model Designer در Visual Studio تغییرات لازم را بر روی مدل انجام داد.

۲. Model First Modeling: در این روش ابتدا مدل توسط برنامه‌نویس در محیط Entity Data Model Designer ایجاد می‌شود، سپس به‌طور خودکار پایگاه داده و کدها و اسکریپ‌های موردنیاز از روی مدل ساخته می‌شوند.

۳. Code First Modeling: در این روش کلاس‌های معادل موجودیت‌ها (جداول) توسط توسعه‌گر نوشته می‌شوند (این کلاس‌ها POCO-Plain OLD CLR Objects نامیده می‌شوند) سپس EF بطور خودکار پایگاه داده و مدل را از روی این کلاس‌ها می‌سازد (موضوع اصلی این کتاب).

از آنجا که هیچ نوشته‌ای بدون اِشکال و بحث نیست، دوستان گرامی می‌توانند نظرات خود را مستقیما با اینجانب از طریق ایمیل piroozman@gmail.com در میان بگذارند و یا در صفحه اختصاصی کتاب در سایت انتشارات مطرح فرمایند.

از اینکه انتشارات پندارِ پارس و مدیر بزرگوار آن آقای حسین یعسوبی به من این فرصت را دادند که بتوانم کتاب خود را چاپ و در اختیار شما خواننده گرامی بگذارم نهایت سپاس و تشکر را دارم.

محمد محمدی پیروز

آبان ۱۳۹۴

فهرست

مقدمه

Entity Framework مجموعه‌ای از تکنولوژهای ADO.NET است که از توسعه برنامه‌های نرم‌افزاری داده‌گرا و شئ‌گرا پشتیبانی می‌کند. معماران و توسعه‌دهندگان برنامه‌های داده‌گرا، همواره در تلاش برای دستیابی به دو هدف بسیار متفاوت هستند. نخست آنکه آنها می‌بایست موجودیت‌ها، روابط بین آنها و منطق تجاری برنامه‌ها را مدل‌سازی کنند و دوم این که باید با مخازن داده‌ای مختلفی که برای ذخیره‌سازی و بازیابی داده‌ها استفاده می‌شود، کار کنند. داده‌ها ممکن است در چندین مخزن داده‌ی متفاوت ذخیره شوند که هر کدام پروتکل مختص خود را داشته باشند، حتی برنامه‌هایی که با یک مخزن داده کار می‌کنند باید بین نیازهای سامانه ذخیره‌سازی در برابر نیازهای نوشتن برنامه‌ای کارآمد و نگهداری موثر کدهای برنامه، توازنی برقرار کنند.

Entity Framework این توانایی را به توسعه‌دهندگان می‌دهد که با داده‌ها در چهارچوب اشیا و خواص مجموعه معینی، مانند مشتریان و آدرس آنها کار کنند، بدون این که بخواهند خودشان را با جداول پایگاه داده و ستون‌هایی که این داده‌ها در آنها ذخیره می‌شوند، درگیر کنند. توسعه‌دهندگان می‌توانند در هنگام روبرو شدن با داده‌ها، با استفاده از Entity Framework در سطح بالاتری از انتزاع برنامه نویسی کرده و برنامه‌های کاربردی داده‌گرا و شئ‌گرا را ایجاد و نگهداری کنند. برنامه‌های تولید شده با استفاده از Entity Framework در مقایسه با برنامه‌های قدیمی کدهای کمتری دارد زیرا Entity Framework مولفه‌ای از .NET Framework. است، بنابراین برنامه نویسی با Entity Framework سریع‌تر و کارآمدتر خواهد بود.

ASP.NET MVC بستری برای ایجاد برنامه‌های مبتنی بر وب است که توسط مایکروسافت ارائه شده و از کارایی و نظم موجود در معماری MVC (Model View Controller) بهره می‌برد.

ASP.NET MVC می‌تواند جایگزین کاملی برای ASP.NET Web Forms باشد و مزایای بسیاری را برای توسعه دهندگان وب به ارمغان آورده است. با توجه به این که تمرکز اصلی این کتاب بر روی Entity Framework جهت ایجاد پایگاه داده و تعامل با آن (ایجاد، به‌روزرسانی و حذف داده‌ها از پایگاه داده) می‌باشد، لذا به آموزش ASP.NET MVC کمتر پرداخته شده است. بنابراین به یک آشنایی ابتدایی با امکانات و تسهیلات ASP.NET MVC، نیازمند است.

اگر تا آخر کتاب با ما همراه باشید خواهید توانست:

- ✓ با نحوه بکارگیری Entity Framework در برنامه‌های تحت وب تولید شده در چهارچوب ASP.NET MVC کاملا آشنا شوید.
- ✓ Data Model های برنامه‌های خود را با Entity Framework ایجاد کنید.

- ✓ قابلیت‌های اساسی Create، Read، Update و Delete با استفاده از Entity Framework را پیاده‌سازی کنید.
- ✓ داده‌های نمایش داده شده در صفحات وب را صفحه‌بندی، فیلتر و مرتب کنید.
- ✓ دستورات SQL تولید شده به وسیله Entity Framework را رهگیری کنید و از شر خطاهای موقت تولید شده در هنگام کار با پایگاه داده‌های راه دور خلاص شوید.
- ✓ پایگاه داده و برنامه کاربردی تولید شده را با استفاده از تکنیک Code First Migrations مربوط به Entity Framework بر روی Window Azure مستقر کنید.
- ✓ مدل داده‌های مرتبط با سایر مدل‌های موجود در برنامه خود را خوانده و آنها را به‌روزرسانی کنید.
- ✓ با استفاده از Entity Framework مدل‌های برنامه‌نویسی ناهمگام را پیاده‌سازی کنید.
- ✓ با استفاده از Entity Framework و Stored Procedure ها عملیات درج، به‌روزرسانی و حذف داده‌ها از پایگاه داده را فرا بگیرید.
- ✓ همزمانی‌های عملیات مختلف درج، به‌روزرسانی و حذف از پایگاه داده را مدیریت کنید.
- ✓ با Entity Framework وراثت بین مدل‌های خود را پیاده‌سازی کنید.
- ✓ با برخی سناریوهای پیشرفته در Entity Framework آشنا شوید.
- ✓ خطاهای رایج استفاده از Entity Framework را شناخته و آنها را حل کنید.

افزون بر مطالب بالا، لینک‌های کاربردی و منابع بسیاری در کتاب برای تشریح کامل‌تر مطالب آورده شده است و دیگر تکنیک‌های مرتبط با موضوع، به شما توسعه‌گر محترم معرفی شده است.

فصل یکم

ایجاد یک Entity Framework Data Model

نمونه برنامه کاربردی تحت وب Contoso University به شما نشان می‌دهد که چگونه یک برنامه کاربردی ASP.NET MVC 5 با استفاده از Entity Framework 6 و Visual Studio 2013 ایجاد کنید. این کتاب، از گردش کار Code First استفاده می‌کند. برای اطلاعات در خصوص چگونگی انتخاب بین Code First، Database First و Model First به Entity Framework Development Workflows به نشانی زیر مراجعه کنید:

https://msdn.microsoft.com/en-us/library/ms178359.aspx#dbfmfcf

برنامه نمونه یک وب سایت برای دانشگاه Contoso فرضی است. این وب سایت دارای قابلیت‌هایی مانند پذیرش دانشجو، ایجاد واحدهای درسی (course) و تعیین استاد (instructor assignments) می باشد. این کتاب چگونگی ساخت نمونه برنامه کاربردی Contoso University را توضیح می‌دهد.

نسخه‌های نرم افزارهای استفاده شده در این کتاب

- Visual Studio 2013
- NET 4.5
- Entity Framework 6 (Entity Framework 6.1.0 NuGet package)
- Windows Azure SDK 2.2

این کتاب همچنین با Visual Studio 2013 Express for Web یا Visual Studio 2012 کار می‌کند. برای استقرار Windows Azure با Visual Studio 2012 به نسخه VS 2012 Windows Azure SDK نیاز است.

نسخه‌های آموزش

نسخه‌های پیشین این آموزش تحت عناوین EF 4.1/MVC 3 و Getting Started with EF 5 using MVC 4 به ترتیب در نشانی های زیر مراجعه کنید.

http://social.technet.microsoft.com/wiki/contents/articles/11608.e-book-gallery-for-microsoft-technologies.aspx#GettingStartedwiththeEntityFramework4.1usingASP.NETMVC

http://www.asp.net/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application

پرسش‌ها و پاسخ‌ها

لطفا به منظور بهبود این کتاب و تاثیر آن در یادگیری، در قسمت Comments در زیر این صفحات نسخه‌ی این کتاب در سایت ASP.NET بازخورد خود را فراموش نکنید. اگر سوالی دارید که به طور مستقیم مربوط به این کتاب نمی‌باشد می‌توانید آن را در صفحات مربوط به ASP.NET Entity Framework forum، Entity Framework and LINQ to Entities forum یا StackOverflow.com به ترتیب به نشانی های زیر مطرح نمایید. تشکر می‌کنیم.

http://forums.asp.net/1227.aspx

https://social.msdn.microsoft.com/Forums/en-US/home?forum=adodotnetentityframework

http://stackoverflow.com/

اگر با مشکلی مواجه شدید و نتوانستید آنرا حل کنید، عموما راه حل مسئله را می‌توانید با مقایسه کدهای خود و پروژه کاملی که دانلود کرده‌اید بیابید. برای برخی از خطاهای عمومی و اینکه چطور آنها را حل کنید، به بخش Common errors, and solutions or workarounds for them به نشانی زیر مراجعه کنید.

http://www.asp.net/mvc/overview/getting-started/getting-started-with-ef-using-mvc/advanced-entity-framework-scenarios-for-an-mvc-web-application#errors

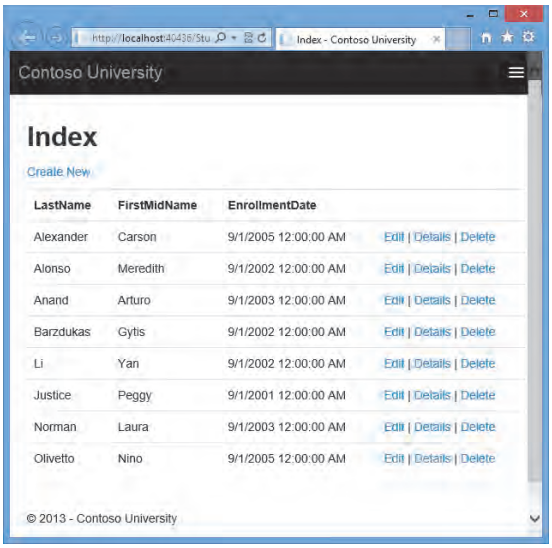
پروژه را دانلود کنید:

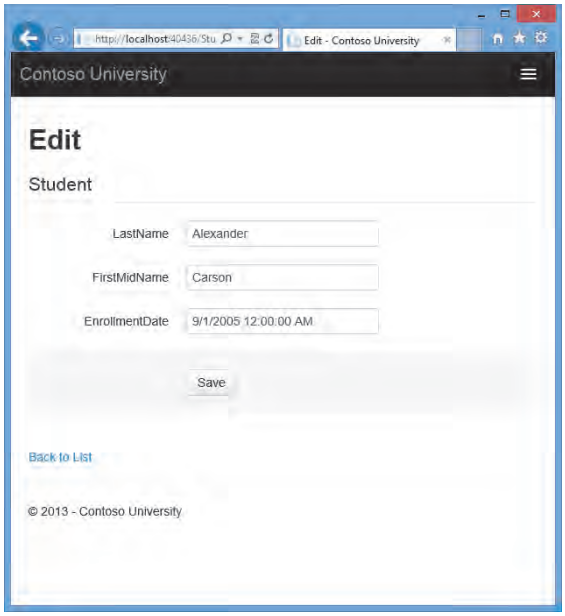
پروژه را از آدرس زیر دانلود کنید:

https://code.msdn.microsoft.com/ASPNET-MVC-Application-b01a9fe8

برنامه کاربردی تحت وب Contoso University

برنامه‌ای که در طی این کتاب خواهید ساخت، یک وب سایت دانشگاهی ساده می‌باشد. کاربران می‌توانند اطلاعات مربوط به دانشجویان، واحدهای درسی و اساتید را مشاهده کرده و به‌روزرسانی کنند.





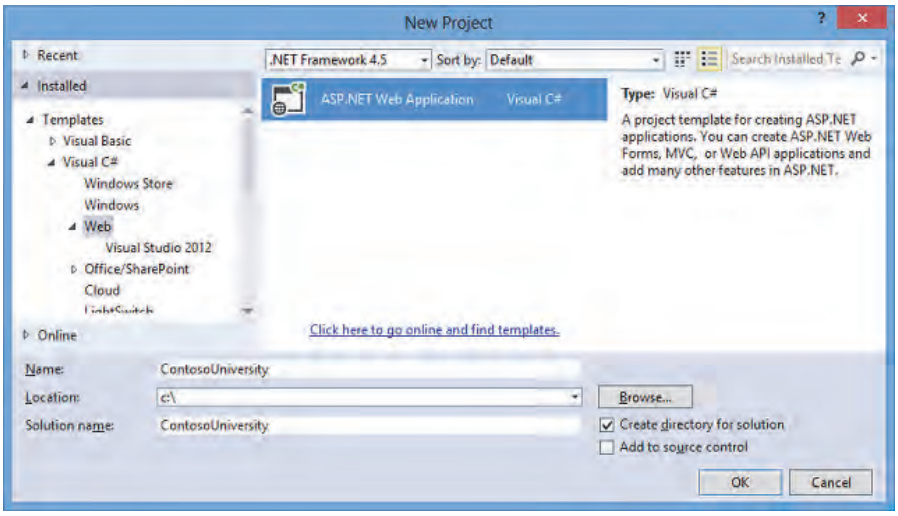
سبک UI این سایت به آنچه که به وسیله الگوهای درون ساختی تولید می‌شود نزدیک است، بنابراین در این کتاب می‌توانید به طور عمده بر روی چگونگی استفاده از Entity Framework تمرکز کنید.

پیش نیازها

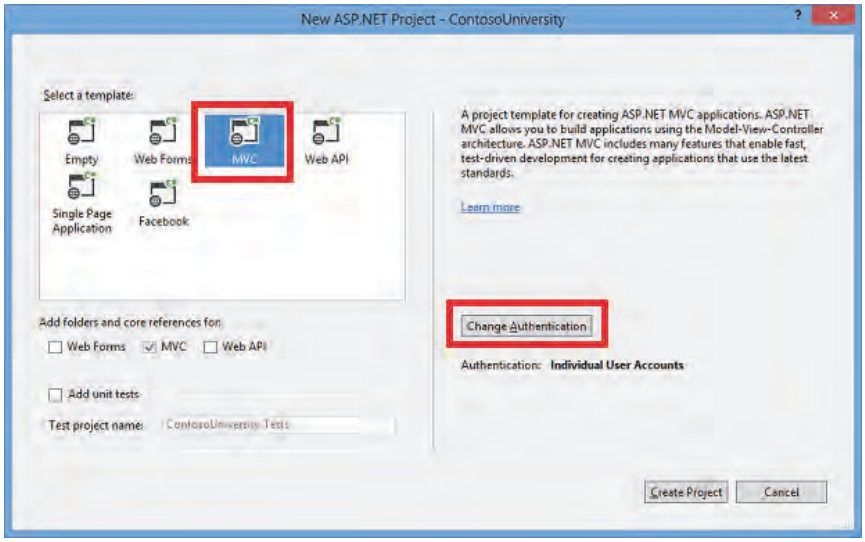
نسخه های نرم افزار در ابتدای این فصل را ببینید. Entity Framework 6 پیش نیاز نمی‌باشد زیرا نصب EF NuGet package قسمتی از این کتاب می‌باشد.

ایجاد یک برنامه کاربردی تحت وب MVC

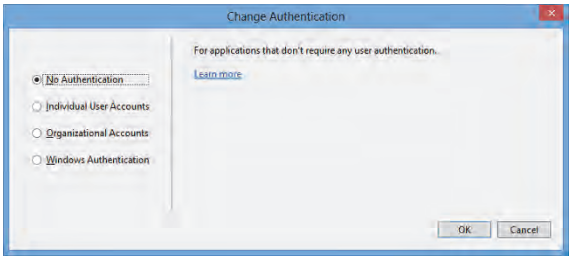
Visual Studio را باز کرده و یک پروژه C# Web به نام "ContosoUniversity" ایجاد کنید.



در پنچره New ASP.NET Project قالب MVC را انتخاب کرده و روی Change Authentication کلیک کنید.



در پنجره Change Authentication، گزینه No Authentication را انتخاب کرده و روی OK کلیک کنید. در این آموزش نیازی به ورود کاربران یا محدودیت‌های دسترسی بر اساس شخص ورود کننده نخواهید داشت.



به پنجره ASP.NET Project برگشته و برای ایجاد پروژه روی OK کلیک کنید.

تنظیم استایل سایت

چند تغییر ساده، منوی سایت(site menu)، جانمایی(layout) و صفحه خانگی(home page) را تنظیم خواهد کرد.

فایل Views\Shared_Layout.cshtml را باز کرده و تغییرات زیر را ایجاد کنید:

- تمامی گزینه‌های "MY ASP.NET Application"و "Application name" را به عبارت "Contoso University" تغییر دهید.

- ورودی‌های منو برای Students، Courses، Instructors و Departments را وارد کنید.

تغییرات به صورت رنگی مشخص شده است.

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>@ViewBag.Title - Contoso University</title>
   @Styles.Render("~/Content/css")
   @Scripts.Render("~/bundles/modernizr")
</head>
<body>
   <div class="navbar navbar-inverse navbar-fixed-top">
      <div class="navbar-inner">
         <div class="container">
            <button type="button" class="btn btn-navbar" datatoggle="collapse" data-target=".nav-
collapse">
               <span class="icon-bar"></span>
               <span class="icon-bar"></span>
               <span class="icon-bar"></span>
            </button>
            @Html.ActionLink("Contoso University", "Index", "Home", null, new { @class = "brand" })
            <div class="nav-collapse collapse">
               <ul class="nav">
                  <li>@Html.ActionLink("Home", "Index", "Home")</li>
                  <li>@Html.ActionLink("About", "About", "Home")</li>
                  <li>@Html.ActionLink("Students", "Index", "Student")</li>
                  <li>@Html.ActionLink("Courses", "Index", "Course")</li>
                   <li>@Html.ActionLink("Instructors", "Index", "Instructor")</li>
                   <li>@Html.ActionLink("Departments", "Index", "Department")</li>
               </ul>
            </div>
         </div>
      </div>
   </div>

   <div class="container">
      @RenderBody()
      <hr />
      <footer>
         <p>&copy; @DateTime.Now.Year - Contoso University</p>
```

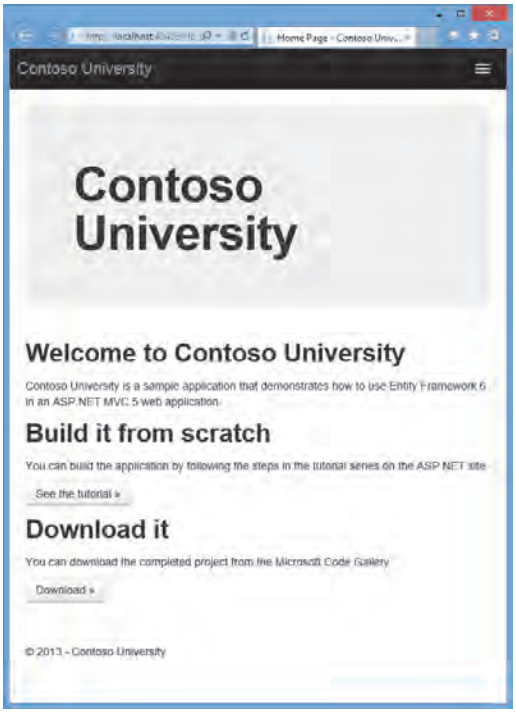
برای جایگزینی متنی که درباره ASP.NET and MVC است با متنی درباره این برنامه، کد زیر را به جای محتوی فایل Views\Home\Index.cshtml قرار دهید.

```
@{ ViewBag.Title = "Home Page"; }
<div class="jumbotron">
   <h1>Contoso University</h1>
</div>
```

```
<div class="row">
   <div class="col-md-4">
      <h2>Welcome to Contoso University</h2>
      <p>Contoso University is a sample application that demonstrates how to use Entity Framework 6 in
an ASP.NET MVC 5 web application.</p>
   </div>
   <div class="col-md-4">
      <h2>Build it from scratch</h2>
      <p>You can build the application by following the steps in the tutorial series on the ASP.NET
site.</p>
      <p><a class="btn btn-default" href="http://www.asp.net/mvc/tutorials/getting-started-with-ef-
usingmvc/">See the tutorial &raquo;</a></p>
   </div>

   <div class="col-md-4">
      <h2>Download it</h2>
      <p>You can download the completed project from the Microsoft Code Gallery.</p>
      <p><a class="btn btn-default" href="http://code.msdn.microsoft.com/ASPNET-MVC-
Applicationb01a9fe8">Download &raquo;</a></p>
   </div>
</div>
```

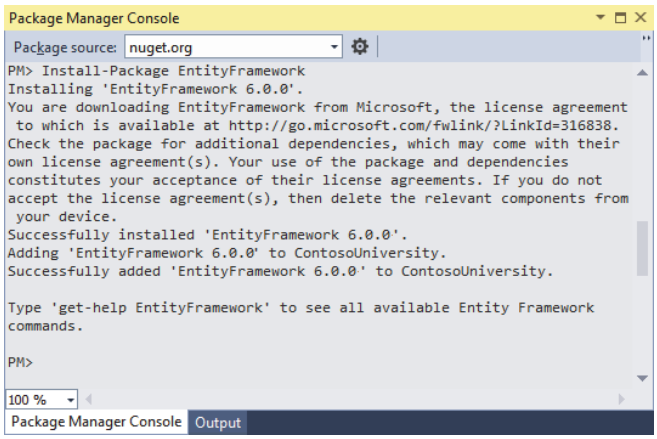
کلیدهای ترکیبی CTRL+F5 را برای اجرای سایت فشار دهید. صفحه Home با منوی اصلی را می‌بینید.



نصب Entity Framework 6

از منوی Tools روی Library Package Manager کلیک کنید و سپس روی گزینه Package Manager Console کلیک کنید. در پنجره Package Manager Console دستور زیر را وارد کنید.

Install-Package EntityFramework

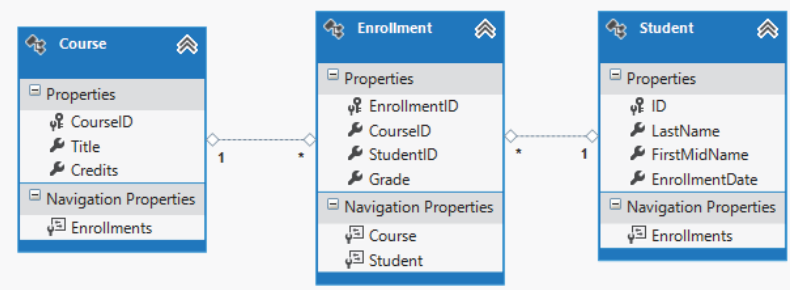


تصویر نصب نسخه 6.0.0 را نشان می دهد، اما NuGet آخرین نسخه Entity Framework را نصب خواهد کرد (به استثنای نسخه های Release).

این مرحله یکی از چند مرحله‌ای است که در این کتاب آنرا به صورت دستی انجام می‌دهید، اما می‌توانید با استفاده از ویژگی ASP.NET MVC scaffolding آنرا به طور خودکار انجام دهید. آنها را به صورت دستی انجام دهید تا بتوانید مراحل مورد نیاز برای استفاده از Entity Framework را مشاهده کنید. بعدها از Scaffolding برای ایجاد کنترل‌کننده‌ها (Controllers) و نماها (Views) استفاده خواهید کرد. روش جایگزین این است که اجازه دهیم Scaffolding به طور خودکار EF NuGet package را نصب کرده، کلاس‌های Database Context و رشته اتصال را ایجاد کند. وقتی این مراحل را فرا گرفتید، از این مراحل صرفنظر کرده و پس از اینکه کلاس‌های موجودیت (Entity) خود را ایجاد کردید، ساخت کنترل‌کننده‌های MVC را به Visual Studio Scaffold خواهید سپرد.

ایجاد Data Model

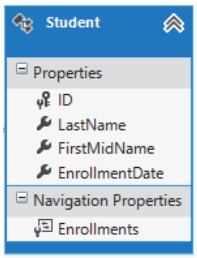
کلاس‌های موجودیت را برای برنامه Contoso University ایجاد خواهید کرد. با سه موجودیت زیر شروع خواهید کرد:



رابطه بین دو موجودیت Student و Enrollment و رابطه بین دو موجودیت Courses و Enrollment از نوع یک به چند است. به عبارت دیگر یک دانشجو می‌تواند در هر تعداد واحد درسی که می‌خواهد ثبت نام کند و یک درس می‌تواند هر تعداد دانشجو داشته باشد. در بخش‌های زیر کلاسی را برای هر یک از این موجودیت ها ایجاد خواهید کرد.

نکته: اگر بخواهید پیش از ایجاد تمامی این کلاس‌های موجودیت پروژه را کامپایل کنید، با خطاهای کامپایل مواجه خواهید شد.

موجودیت Student



در پوشه Models یک کلاس با نام Student.cs ایجاد کرده و کدهای زیر را در آن قراردهید:

```
using System;
using System.Collections.Generic;

namespace ContosoUniversity.Models
{
    public class Student
    {
        public int ID { get; set; }
        public string LastName { get; set; }
        public string FirstMidName { get; set; }
        public DateTime EnrollmentDate { get; set; }
```

```
        public virtual ICollection<Enrollment> Enrollments { get;
set; }
    }
}
```

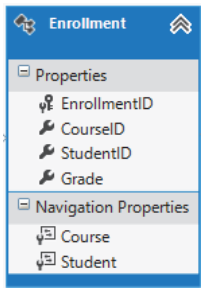
خصیصه ID در جدول پایگاه داده مربوط به این کلاس به یک ستون با کلید اولیه (Primary key) تبدیل خواهد شد. به طور پیش فرض، Entity Framework یک خصیصه با نام ID یا calssnameID را به عنوان کلید اولیه تعبیر خواهد کرد.

خصیصه Enrollments یک خصیصه ناوبری (navigation property) است. خصیصه‌های ناوبری، سایر موجودیت‌هایی را که مرتبط با این موجودیت هستند در خود نگهداری می‌کنند. در این مورد، خصیصه Enrollment موجود در موجودیت Student، تمامی موجودیت‌های Enrollment را که با موجودیت مرتبط هستند در خود نگهداری خواهد کرد. به عبارت دیگر اگر یک ردیف دانشجوی معینی در پایگاه داده دو ردیف Enrollment مرتبط با خود را داشته باشد (ردیف‌هایی که حاوی مقدار کلید اولیه Student در ستون کلید خارجی StudentID خود می‌باشند)، خصیصه ناوبری Enrollment در موجودیت Student دارای دو موجودیت Enrollment خواهد بود.

خصیصه‌های ناوبری اصولا به طور مجازی (Virtual) تعریف می‌شوند که بتوانند از مزیت‌های ویژه‌ی Entity Framework مانند بارگذاری تنبل (lazy loading) استفاده کنند (بارگذاری تنبل را در فصل هفتم کتاب تحت عنوان بارگذاری داده‌های مرتبط تنبل، حریص و صریح خواهید خواند).

اگر یک خصیصه ناوبری بتواند چندین موجودیت را در خود نگهداری کند (به عنوان یک رابطه چند به چند یا یک به چند)، آن باید نوعی List باشد که بتوان آنها را اضافه کرده، حذف نمود یا به‌روزرسانی کرد، مانند ICollection.

موجودیت Enrollment



در پوشه Models، فایل Enrollment.cs را ایجاد کرده و کدهای زیر را با کدهای آن جایگزین کنید:

```
namespace ContosoUniversity.Models
{
    public enum Grade
```

```
    {
        A, B, C, D, F
    }
    public class Enrollment
    {
        public int EnrollmentID { get; set; }
        public int CourseID { get; set; }
        public int StudentID { get; set; }
        public Grade? Grade { get; set; }
        public virtual Course Course { get; set; }
        public virtual Student Student { get; set; }
    }
}
```

خصیصه EnrollmentID به کلید اولیه تبدیل خواهد شد، این موجودیت به جای الگوی ID (آنطور که در موجودیت Student دیدید) از الگوی classnameID استفاده کرده است. معمولا شما یک الگو را انتخاب خواهید کرد و آنرا در تمامی Data Model های خود به کار خواهید بست. در اینجا ما از هر دو نوع استفاده کردیم که بتوانید از هر کدام از الگوها که خواستید استفاه کنید. بعدها در همین کتاب خواهید دید که چطور استفاده از ID بدون classname برای پیاده سازی سلسله مراتب در Data Model آسان‌تر خواهد بود.

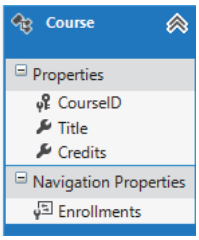
خصیصه Grade یک enum است. نماد پرسش پس از تعریف نوع Grade نشان می‌دهد که خصیصه Grade یک nullable است. نمره‌ای که null است متفاوت از یک نمره صفر خواهد بود – null به معنای این است که نمره مشخص نیست یا هنوز نمره‌ای برای آن درس تعیین نشده است.

خصیصه StudentID یک کلید خارجی است و خصیصه ناوبری مرتبط با آن Student می‌باشد. یک موجودیت Enrollment با یک موجودیت Student مرتبط است، از این رو خصیصه تنها می‌تواند یک موجودیت Student را در خود نگه‌دارد (برخلاف خصیصه ناوبری Student.Enrollments که پیش از این دیدید که می‌تواند بیش از یک موجودیت Enrollment را در خود نگه دارد).

خصیصه CourseID یک کلید خارجی است و خصیصه ناوبری منطبق با آن Course می‌باشد. یک موجودیت Enrollment با یک موجودیت Course در ارتباط است.

Entity Framework یک خصیصه را وقتی به عنوان یک خصیصه کلید خارجی تفسیر خواهد کرد که نامگذاری آن به صورت <navigation property name><primary key property name> باشد (برای نمونه، StudentID برای خصیصه ناوبری Student از آنجا که کلید اولیه موجودیت Student، ID است). همچنین خصیصه‌های کلید خارجی را می‌توان به شکل ساده <primary key property name> نامگذاری کرد (برای مثال CourseID از آنجا که کلید اولیه موجودیت Course، CourseID است).

موجودیت Course



در پوشه Models، فایل Course.cs را ایجاد کرده، کدهای زیر را در آن قرار دهید.

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;

namespace ContosoUniversity.Models
{
    public class Course
    {
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int CourseID { get; set; }
        public string Title { get; set; }
        public int Credits { get; set; }

        public virtual ICollection<Enrollment> Enrollments { get; set; }
    }
}
```

خصیصه Enrollments یک خصیصه ناوبری است. یک موجودیت Course می‌تواند با هر تعداد موجودیت Enrollment در ارتباط باشد.

در قسمت‌های بعدی این کتاب در خصوص صفت DatabaseGenerated بیشتر صحبت خواهیم کرد. در واقع، این صفت به شما اجازه می‌دهد تا به جای اینکه پایگاه داده کلید اولیه را ایجاد کند، آنر در Course وارد کنید.

ایجاد Database Context

کلاس اصلی که مختصات قابلیت‌های Entity Framework برای یک Data Model تعیین می‌کند، کلاس Database Context می باشد. این کلاس را با اشتقاق از کلاس System.Data.Entity.DbContext ایجاد می‌کنید. در کدهای خود، مشخص خواهید کرد که کدام موجودیت در Data Model قرار دارند. همچنین می‌توانید رفتار Entity Framework معینی را سفارشی کنید. در این پروژه، کلاس SchoolContext نامگذاری خواهد شد.

برای ایجاد یک پوشه در پروژه ContosoUniversity، در Solution Explorer روی نام پروژه کلیک راست کرده و روی Add و سپس روی New Folder کلیک کنید. نام پوشه جدید را DAL (سر نام عبارت **D**ata **A**ccess **L**ayer) قرار دهید. در این پوشه یک کلاس جدید با نام SchoolContext.cs ایجاد کرده و کدهای الگو زیر را در آن قرار دهید:

```
using ContosoUniversity.Models;
using System.Data.Entity;
using System.Data.Entity.ModelConfiguration.Conventions;

namespace ContosoUniversity.DAL
{
    public class SchoolContext : DbContext
    {
        public SchoolContext() : base("SchoolContext")
        {
        }
        public DbSet<Student> Students { get; set; }
        public DbSet<Enrollment> Enrollments { get; set; }
        public DbSet<Course> Courses { get; set; }

        protected override void OnModelCreating(DbModelBuilder
modelBuilder)
        {

modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();
        }
    }
}
```

تعیین مجموعه‌های موجودیت

این کد برای هر مجموعه موجودیت، یک خصیصه DbSet ایجاد می‌کند. در واژه شناسی Entity Framework، یک مجموعه موجودیت، اصولا منطبق با یک جدول پایگاه داده است و یک موجودیت منطبق با یک ردیف در آن جدول.

می‌توانید عبارات DbSet<Enrollment> و DbSet<Course> را حذف کنید و آن همان کار را خواهد کرد. Entity Framework آنها را به طور ضمنی شامل خواهد بود، چون موجودیت Student به موجودیت Enrollment و موجودیت Enrollment به موجودیت Course ارجاع می‌کند.

تعیین رشته اتصال

نام رشته اتصال (که آنرا بعدا به فایل Web.config اضافه خواهید کرد) به سازنده (constructor) پاس داده شده است.

```
public SchoolContext() : base("SchoolContext")
{
}
```

همچنین می‌توانید رشته اتصال خود را به جای نام رشته ای که در فایل Web.config ذخیره شده است به آن پاس دهید. برای اطلاعات بیشتر درباره گزینه‌هایی برای مشخص کردن پایگاه داده برای استفاده، Entity Framework-Connection and Models را در نشانی زیر ببینید:

http://msdn.microsoft.com/en-us/data/jj592674

اگر یک رشته اتصال را مشخص نکنید یا نام آنرا به صراحت ذکر نکنید، Entity Framework فرض می‌کند که نام رشته اتصال هم نام کلاس می‌باشد. نام رشته اتصال پیش فرض در این مثال SchoolContext خواهد بود، همانند آنچه که صریحا تعیین کردید.

تعیین نام‌های مفرد برای جدول

عبارت modelBuilder.Conventions.Remove در متد OnModelCreating مانع از جمع بسته شدن نام جداول می‌شود. اگر از این متد استفاده نکنید، جداول تولید شده در پایگاه داده به جای اینکه به شکل مفرد (Student، Course و Enrollment) باشد به صورت جمع (Students، Courses و Enrollments) نامگذاری خواهند شد. توسعه دهندگان در خصوص اینکه آیا نام جداول باید جمع باشد یا مفرد توافق ندارند. این کتاب از شکل مفرد آن استفاده می‌کند، اما نکته مهم این است که بتوانید هرآنچه که خود ترجیح می‌دهید را انتخاب کنید. می‌توانید از کد مذکور استفاده کنید یا آنرا حذف نمایید.

تنظیم EF برای مقدار دهی اولیه پایگاه داده با داده‌های آزمایشی

هنگامی که برنامه را اجرا می‌کنید، Entity Framework می‌تواند یک پایگاه داده را ایجاد کند (یا اینکه حذف و دوباره ایجاد کند– drop and re-create). می‌توانید تعیین کنید که با هر بار اجرای برنامه این کار انجام شود یا اینکه تنها وقتی که مدل با پایگاه داده موجود ناهمگام (out of sync) است این اتفاق بیافتد. همچنین می‌توانید یک متد Seed بنویسید که پس از ایجاد پایگاه داده و به منظور پرکردن آن از داده های آزمایشی، Entity Framework آنرا به طور خودکار فراخوانی کند.