عنوان: **MVC Scaffolding #1**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۲ ۱۳:۱۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, MVC, MVC Scaffolding

پیشنیازها

کل سری ASP.NET MVC
به همراه کل سری EF Code First

MVC Scaffolding چیست؟

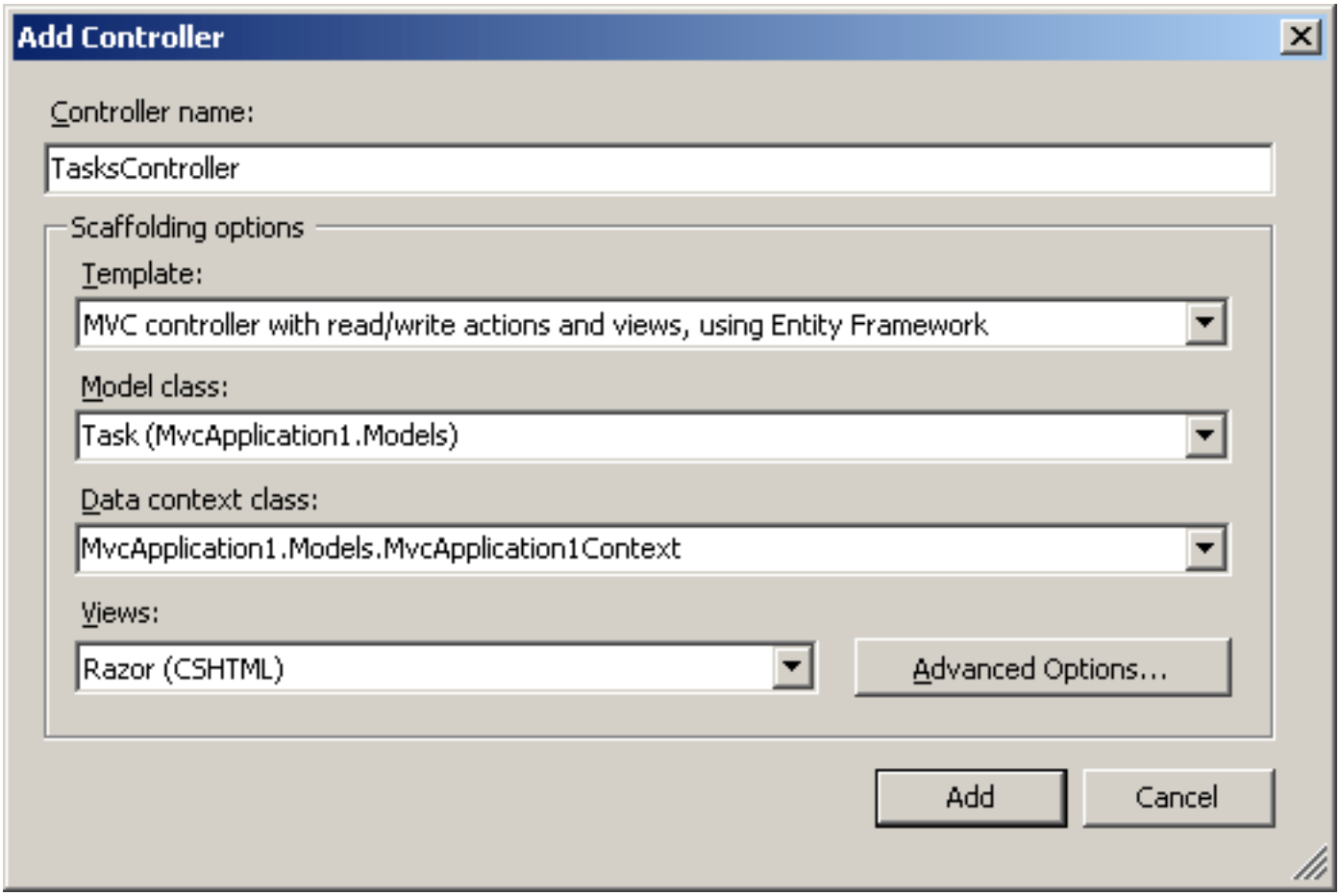
MVC Scaffolding ابزاری است برای تولید خودکار کدهای «اولیه» برنامه، جهت بالا بردن سرعت تولید برنامه‌های ASP.NET MVC مبتنی بر EF Code First.

بررسی مقدماتی MVC Scaffolding

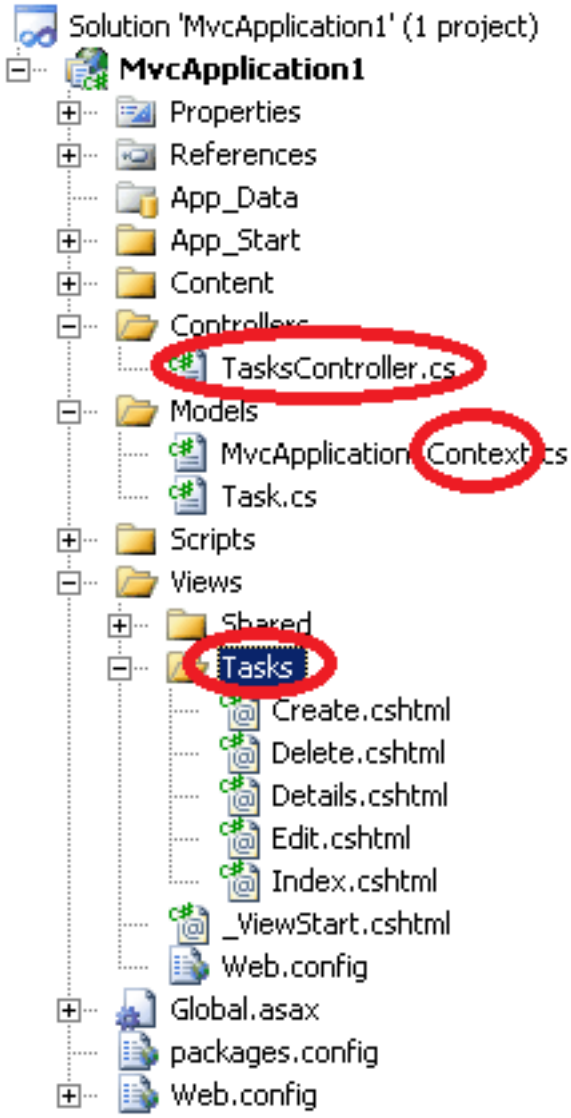
امکان اجرای ابزار MVC Scaffolding از دو طریق دستورات خط فرمان Powershell و یا صفحه دیالوگ افزودن یک کنترلر در پروژه‌های ASP.NET MVC وجود دارد. در ابتدا حالت ساده و ابتدایی استفاده از صفحه دیالوگ افزودن یک کنترلر را بررسی خواهیم کرد تا با کلیات این فرآیند آشنا شویم. سپس در ادامه به خط فرمان Powershell که اصل توانمندی‌ها و قابلیت‌های سفارشی MVC Scaffolding در آن قرار دارد، خواهیم پرداخت.
برای این منظور یک پروژه جدید MVC را آغاز کنید؛ ابزارهای مقدماتی MVC Scaffolding از اولین به روز رسانی ASP.NET MVC3 به بعد با VS.NET یکپارچه هستند.
ابتدا کلاس زیر را به پوشه مدل‌های برنامه اضافه کنید:

```
using System;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace MvcApplication1.Models
{
    public class Task
    {
        public int Id { set; get; }

        [Required]
        public string Name { set; get; }

        [DisplayName("Due Date")]
        public DateTime? DueDate { set; get; }

        [DisplayName("Is Complete")]
        public bool IsComplete { set; get; }

        [StringLength(450)]
        public string Description { set; get; }
    }
}
```

سپس بر روی پوشه Controllers کلیک راست کرده و گزینه Add controller را انتخاب کنید. تنظیمات صفحه ظاهر شده را مطابق شکل زیر تغییر دهید:



همانطور که ملاحظه می‌کنید در قسمت قالب‌ها، تولید کنترلرهایی با اکشن متدهای ثبت و نمایش اطلاعات مبتنی بر EF Code First انتخاب شده است. کلاس مدل نیز به کلاس Task فوق تنظیم گردیده و در زمان انتخاب DbContext مرتبط، گزینه new data context را انتخاب کرده و نام پیش فرض آن‌را پذیرفته‌ایم. زمانیکه بر روی دکمه Add کلیک کنیم، اتفاقات ذیل رخ خواهند داد:



الف) کنترلر جدید TasksController.cs به همراه تمام کدهای Insert/Update/Delete/Display مرتبط تولید خواهد شد.

ب) کلاس DbContext خودکاری به نام MvcApplication1Context.cs در پوشه مدل‌های برنامه ایجاد می‌گردد تا کلاس Task را در معرض دید EF Code first قرار دهد. (همانطور که عنوان شد یکی از پیشنیازهای بحث Scaffolding آشنایی با EF Code first است)

ج) در پوشه Views\Tasks، پنج View جدید را جهت مدیریت فرآیندهای نمایش صفحات Insert، حذف، ویرایش، نمایش و غیره تهیه می‌کند.

د) فایل وب کانفیگ برنامه جهت درج رشته اتصالی به بانک اطلاعاتی تغییر کرده است. حالت پیش فرض آن استفاده از SQL CE است و برای استفاده از آن نیاز است قسمت 15 سری EF سایت جاری را پیشتر مطالعه کرده باشید (به چه اسمبلی‌های دیگری مانند System.Data.SqlServerCe.dll برای اجرا نیاز است و چطور باید اتصال به بانک اطلاعاتی را تنظیم کرد)

معایب:

کیفیت کد تولیدی پیش فرض قابل قبول نیست:

- DbContext در سطح یک کنترلر وهله سازی شده و الگوی Context Per Request در اینجا بکارگرفته نشده است. واقعیت یک برنامه ASP.NET MVC کامل، داشتن چندین Partial View تغذیه شونده از کنترلرهای مختلف در یک صفحه واحد است. اگر قرار باشد به ازای هر کدام یکبار DbContext وهله سازی شود یعنی به ازای هر صفحه چندین بار اتصال به بانک اطلاعاتی باید برقرار شود که سربار زیادی را به همراه دارد. (قسمت 12 سری EF سایت جاری)

- اکشن متدها حاوی منطق پیاده سازی اعمال CRUD یا همان Create/Update/Delete هستند. به عبارتی از یک لایه سرویس برای

خلوت کردن اکشن متدها استفاده نشده است.
- از ViewModel تعریف شده‌ای به نام Task هم به عنوان Domain model و هم ViewModel استفاده شده است. یک کلاس متناظر با جداول بانک اطلاعاتی می‌تواند شامل فیلدهای بیشتری باشد و نباید آن‌را مستقیما در معرض دید یک View قرار داد (خصوصا از لحاظ مسایل امنیتی).

مزیت‌ها:
قسمت عمده‌ای از کارهای «اولیه» تهیه یک کنترلر و همچنین Viewهای مرتبط به صورت خودکار انجام شده‌اند. کارهای اولیه‌ای که با هر روش و الگوی شناخته شده‌ای قصد پیاده سازی آن‌ها را داشته باشید، وقت زیادی را به خود اختصاص داده و نهایتا آنچنان تفاوت عمده‌ای هم با کدهای تولیدی در اینجا نخواهند داشت. حداکثر فرم‌های آن‌را بخواهید با jQuery Ajax پیاده سازی کنید یا کنترل‌های پیش فرض را با افزونه‌های jQuery غنی سازی نمائید. اما شروع کار و کدهای اولیه چیزی بیشتر از این نیست.

نصب بسته اصلی MVC Scaffolding توسط NuGet

بسته اصلی MVC Scaffolding را با استفاده از دستور خط فرمان Powershell ذیل، از طریق منوی Tools، گزینه Library package manager و انتخاب Package manager console می‌توان به پروژه خود اضافه کرد:

```
Install-Package MvcScaffolding
```

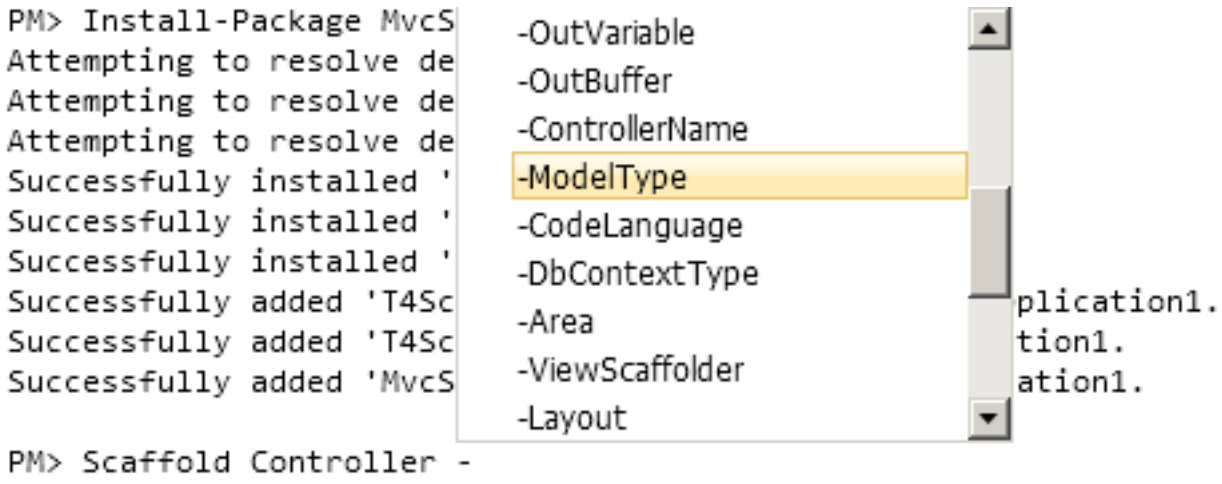
اگر به مراحل نصب آن دقت کنید یک سری وابستگی را نیز به صورت خودکار دریافت کرده و نصب می‌کند:

```
Attempting to resolve dependency 'T4Scaffolding'.
Attempting to resolve dependency 'T4Scaffolding.Core'.
Attempting to resolve dependency 'EntityFramework'.
Successfully installed 'T4Scaffolding.Core 1.0.0'.
Successfully installed 'T4Scaffolding 1.0.8'.
Successfully installed 'MvcScaffolding 1.0.9'.
Successfully added 'T4Scaffolding.Core 1.0.0' to MvcApplication1.
Successfully added 'T4Scaffolding 1.0.8' to MvcApplication1.
Successfully added 'MvcScaffolding 1.0.9' to MvcApplication1.
```

از مواردی که با T4 آغاز شده‌اند در قسمت‌های بعدی برای سفارشی سازی کدهای تولیدی استفاده خواهیم کرد.
پس از اینکه بسته MvcScaffolding به پروژه جاری اضافه شد، همان مراحل قبل را که توسط صفحه دیالوگ افزودن یک کنترلر انجام دادیم، اینبار به کمک دستور ذیل نیز می‌توان پیاده سازی کرد:

```
Scaffold Controller Task
```

نوشتن این دستور نیز ساده است. حروف sca را تایپ کرده و دکمه tab را فشار دهید. منویی ظاهر خواهد شد که امکان انتخاب دستور Scaffold را می‌دهد. یا برای نوشتن Controller نیز به همین نحو می‌توان عمل کرد.
نکته و مزیت مهم دیگری که در اینجا در دسترس می‌باشد، سوئیچ‌های خط فرمانی است که به همراه صفحه دیالوگ افزودن یک کنترلر وجود ندارند. برای مثال دستور Scaffold Controller را تایپ کرده و سپس یک خط تیره را اضافه کنید. اکنون دکمه tab را مجددا بفشارید. منویی ظاهر خواهد شد که بیانگر سوئیچ‌های قابل استفاده است.



برای مثال اگر بخواهیم دستور Scaffold Controller Task را با جزئیات اولیه کاملتری ذکر کنیم، مانند تعیین نام دقیق کلاس مدل و کنترلر تولیدی به همراه نام دیگری برای DbContext مرتبط، خواهیم داشت:

```
Scaffold Controller -ModelType Task -ControllerName TasksController -DbContextType TasksDbContext
```

اگر این دستور را اجرا کنیم به همان نتیجه حاصل از مراحل توضیح داده شده قبل خواهیم رسید؛ البته یا یک تفاوت: یک Partial View اضافه‌تر نیز به نام CreateOrEdit در پوشه Views\Tasks ایجاد شده است. این Partial View بر اساس بازخورد برنامه نویس‌ها مبنی بر اینکه Viewهای Edit و Create بسیار شبیه به هم هستند، ایجاد شده است.

بهبود مقدماتی کیفیت کد تولیدی MVC Scaffolding

در همان کنسول پاروشل NuGet، کلید up arrow را فشار دهید تا مجددا دستور قبلی اجرا شده ظاهر شود. اینبار دستور قبلی را با سوئیچ جدید Repository (استفاده از الگوی مخزن) اجرا کنید:

```
Scaffold Controller -ModelType Task -ControllerName TasksController -DbContextType TasksDbContext -
Repository
```

البته اگر دستور فوق را به همین نحو اجرا کنید با یک سری خطای Skipping مواجه خواهید شد مبنی بر اینکه فایل‌های قبلی موجود هستند و این دستور قصد بازنویسی آن‌ها را ندارد. برای اجبار به تولید مجدد کدهای موجود می‌توان از سوئیچ Force استفاده کرد:

```
Scaffold Controller -ModelType Task -ControllerName TasksController -DbContextType TasksDbContext -
Repository -Force
```

اتفاقی که در اینجا رخ خواهد داد، بازنویسی کد بی‌کیفت ابتدایی همراه با وهله سازی مستقیم DbContext در کنترلر، به نمونه بهتری که از الگوی مخزن استفاده می‌کند می‌باشد:

```
public class TasksController : Controller
    {
        private readonly ITaskRepository taskRepository;

        // If you are using Dependency Injection, you can delete the following constructor
        public TasksController()
            : this(new TaskRepository())
        {
        }

        public TasksController(ITaskRepository taskRepository)
        {
```

```
            this.taskRepository = taskRepository;
        }
```

کیفیت کد تولیدی جدید مبتنی بر الگوی مخزن بد نیست؛ دقیقا همانی است که در هزاران سایت اینترنتی تبلیغ می‌شود؛ اما ... آنچنان مناسب هم نیست و اشکالات زیر را به همراه دارد:

```
public interface ITaskRepository : IDisposable
{
  IQueryable<Task> All { get; }
  IQueryable<Task> AllIncluding(params Expression<Func<Task, object>>[] includeProperties);
  Task Find(int id);
  void InsertOrUpdate(Task task);
  void Delete(int id);
  void Save();
}
```

اگر به ITaskRepository تولیدی دقت کنیم دارای خروجی IQueryable است؛ به این حالت leaky abstraction گفته می‌شود. زیرا امکان تغییر کلی یک خروجی IQueryable در لایه‌های دیگر برنامه وجود دارد و حد و مرز سیستم توسط آن مشخص نخواهد شد. بهتر است خروجی‌های لایه سرویس یا لایه مخزن در اینجا از نوع‌های IList یا IEnumerable باشند که درون آن‌ها از IQueryableها برای پیاده سازی منطق مورد نظر کمک گرفته شده است.

پیاده سازی این اینترفیس در حالت متد Save آن شامل فراخوانی context.SaveChanges است. این مورد باید به الگوی واحد کار (که در اینجا تعریف نشده) منتقل شود. زیرا در یک دنیای واقعی حاصل کار بر روی چندین موجودیت باید در یک تراکنش ذخیره شوند و قرارگیری متد Save داخل کلاس مخزن یا سرویس برنامه، مخزن‌های تعریف شده را تک موجودیتی می‌کند.

اما در کل با توجه به اینکه پیاده سازی منطق کار با موجودیت‌ها به کلاس‌های مخزن واگذار شده‌اند و کنترلرها به این نحو خلوت‌تر گردیده‌اند، یک مرحله پیشرفت محسوب می‌شود.

نظرات خوانندگان

نویسنده: حسین
تاریخ: ۱۳۹۱/۱۱/۰۲ ۱۳:۴۹

فوق العاده بود. اصلاً نمیدونستم که Scaffolding یه همچین قابلیت هایی هم داره.
ممنون.

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۰۲ ۱۴:۱۸

با تشکر
در کل از این روش در تولید پروژه‌های واقعی استفاده میشود ؟

نویسنده: سعید
تاریخ: ۱۳۹۱/۱۱/۰۲ ۱۶:۲۰

حداقل 4 بار در این متن کلمه «اولیه» بکار رفته؛ به همراه گیومه دورش. حتما دلیلی داشته ...

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۱/۱۱/۰۳ ۶:۰

دست مریزاد. خیلی مفید بود. ممنون

نویسنده: سعید رضایی
تاریخ: ۱۳۹۲/۱۲/۰۴ ۱۷:۲۱

با عرض سلام.
موقع نصب تو mvc4 خطای زیر رو میده
Unable to retrieve metadata for 'AhooraTech.Models.prod'. Unable to cast object of type
''System.Data.Entity.Core.Objects.ObjectContext' to type 'System.Data.Objects.ObjectContext'.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۴ ۱۷:۲۷

از EF 6 استفاده کردید؟ بله . فقط برای MVC 5 ابزار Scaffolding را جهت کار با EF 6 به روز کرده‌اند .

عنوان: **MVC Scaffolding #2**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۳ ۸:۱۰
آدرس: www.dotnettips.info
برچسب‌ها: Entity framework, MVC, MVC Scaffolding

از آنجائیکه اصل کار با MVC Scaffolding از طریق خط فرمان پاورشل انجام می‌شود، بنابراین بهتر است در ادامه با گزینه‌ها و سوئیچ‌های مرتبط با آن بیشتر آشنا شویم.
دو نوع پارامتر حین کار با MVC Scaffolding مهیا هستند:

الف) سوئیچ‌ها
مانند پارامترهای boolean عمل کرده و شامل موارد ذیل می‌باشند. تمام این پارامترها به صورت پیش فرض دارای مقدار false بوده و ذکر هرکدام در دستور نهایی سبب true شدن مقدار آن‌ها می‌گردد:
Repository: برای تولید کدها بر اساس الگوی مخزن
Force: برای بازنویسی فایل‌های موجود.
ReferenceScriptLibraries: ارجاعاتی را به اسکریپت‌های موجود در پوشه Scripts، اضافه می‌کند.
NoChildItems: در این حالت فقط کلاس کنترلر تولید می‌شود و از سایر ملحقات مانند تولید Viewها، DbContext و غیره صرفنظر خواهد شد.

ب) رشته‌ها
این نوع پارامترها، رشته‌ای را به عنوان ورودی خود دریافت می‌کنند و شامل موارد ذیل هستند:
ControllerName: جهت مشخص سازی نام کنترلر مورد نظر
ModelType: برای ذکر صریح کلاس مورد استفاده در تشکیل کنترلر بکار می‌رود. اگر ذکر نشود، از نام کنترلر حدس زده خواهد شد.
DbContext: نام کلاس DbContext تولیدی را مشخص می‌کند. اگر ذکر نشود از نامی مانند ProjectNameContex استفاده خواهد کرد.
Project: پیش فرض آن پروژه جاری است یا اینکه می‌توان پروژه دیگری را برای قرار دادن فایل‌های تولیدی مشخص کرد. (برای مثال هربار یک سری کد مقدماتی را در یک پروژه جانبی تولید کرد و سپس موارد مورد نیاز را از آن به پروژه اصلی افزود)
CodeLanguage: می‌تواند cs یا vb باشد. پیش فرض آن زبان جاری پروژه است.
Area: اگر می‌خواهید کدهای تولیدی در یک ASP.NET MVC area مشخص قرار گیرند، نام Area مشخصی را در اینجا ذکر کنید.
Layout: در حالت پیش فرض از فایل layout اصلی استفاده خواهد شد. اما اگر نیاز است از layout دیگری استفاده شود، مسیر نسبی کامل آن‌را در اینجا قید نمائید.

یک نکته:
نیازی به حفظ کردن هیچکدام از موارد فوق نیست. برای مثال در خط فرمان پاورشل، دستور Scaffold را نوشته و پس از یک فاصله، دکمه Tab را فشار دهید. لیست پارامترهای قابل اجرای در این حالت ظاهر خواهند شد. اگر در اینجا برای نمونه Controller انتخاب شود، مجددا با ورود یک فاصله و خط تیره و سپس فشردن دکمه Tab، لیست پارامترهای مجاز و همراه با سوئیچ کنترلر ظاهر می‌گردند.

MVC Scaffolding و مدیریت روابط بین کلاس‌ها

مثال قسمت قبلی بسیار ساده و شامل یک کلاس بود. اگر آن‌را کمی پیچیده‌تر کرده و برای مثال روابط one-to-many و many-to-many را اضافه کنیم چطور؟

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

```
namespace MvcApplication1.Models
{
    public class Task
    {
        public int Id { set; get; }

        [Required]
        public string Name { set; get; }

        [DisplayName("Due Date")]
        public DateTime? DueDate { set; get; }

        [ForeignKey("StatusId")]
        public virtual Status Status { set; get; } // one-to-many
        public int StatusId { set; get; }

        [StringLength(450)]
        public string Description { set; get; }

        public virtual ICollection<Tag> Tags { set; get; } // many-to-many
    }

    public class Tag
    {
        public int Id { set; get; }

        [Required]
        public string Name { set; get; }

        public virtual ICollection<Task> Tasks { set; get; } // many-to-many
    }

    public class Status
    {
        public int Id { set; get; }

        [Required]
        public string Name { set; get; }
    }
}
```

کلاس Task تعریف شده اینبار دارای رابطه many-to-many با برچسب‌های مرتبط با آن است. همچنین یک رابطه one-to-many با کلاس وضعیت هر Task نیز تعریف شده است. به علاوه نکته تعریف « کار با کلیدهای اصلی و خارجی در EF Code first » نیز در اینجا لحاظ گردیده است.

در ادامه دستور تولید کنترلرهای Task، Tag و Status ساخته شده با الگوی مخزن را در خط فرمان پاورشل ویژوال استودیو صادر می‌کنیم:

```
PM> Scaffold Controller -ModelType Task -ControllerName TasksController -DbContextType TasksDbContext -
Repository -Force
PM> Scaffold Controller -ModelType Tag -ControllerName TagsController -DbContextType TasksDbContext -
Repository -Force
PM> Scaffold Controller -ModelType Status -ControllerName StatusController -DbContextType
TasksDbContext -Repository -Force
```

اگر به کارهایی که در اینجا انجام می‌شود دقت کنیم، می‌توان صرفه جویی زمانی قابل توجهی را شاهد بود؛ خصوصا در برنامه‌هایی که از ده‌ها فرم ورود اطلاعات تشکیل شده‌اند. فرض کنید قصد استفاده از ابزار فوق را نداشته باشیم. باید به ازای هر عملیات CRUD دو متد را ایجاد کنیم. یکی برای نمایش و دیگری برای ثبت. بعد بر روی هر متد کلیک راست کرده و Viewهای متناظری را ایجاد کنیم. سپس مجددا یک سری پیاده سازی «مقدماتی» تکراری را به ازای هر متد جهت ثبت یا ذخیره اطلاعات تدارک ببینیم.

اما در اینجا پس از طراحی کلاس‌های برنامه، با یک دستور، حجم قابل توجهی از کدهای «مقدماتی» که بعدها مطابق نیاز ما سفارشی سازی و غنی‌تر خواهند شد، تولید می‌گردند.

چند نکته:

- با توجه به اینکه مدل‌ها تغییر کرده‌اند، نیاز است بانک اطلاعاتی متناظر نیز به روز گردد. مطالب مرتبط با آن‌را در مباحث Migrations می‌توانید مطالعه نمائید.

- View تولیدی رابطه many-to-many را پشتیبانی نمی‌کند. این مورد را باید دستی اضافه و طراحی کنید: (^ و ^)

- رابطه one-to-many به خوبی با View متناظری دارای یک drop down list تولید خواهد شد. در اینجا لیست تولیدی به صورت خودکار با مقادیر خاصیت Name کلاس Status پر می‌شود. اگر این نام دقیقا Name نباشد نیاز است توسط ویژگی به نام DisplayColumn که بر روی نام کلاس قرار می‌گیرد، مشخص کنید از کدام خاصیت باید استفاده شود.

```
@Html.DropDownListFor(model => model.StatusId,
((IEnumerable<Status>)ViewBag.PossibleStatus).Select(option => new SelectListItem {
  Text = (option == null ? "None" : option.Name),
  Value = option.Id.ToString(),
  Selected = (Model != null) && (option.Id == Model.StatusId)
}), "Choose...")
@Html.ValidationMessageFor(model => model.StatusId)
```

تولید آزمون‌های واحد به کمک MVC Scaffolding

MVC Scaffolding امکان تولید خودکار کلاس‌ها و متدهای آزمون واحد را نیز دارد. برای این منظور دستور زیر را در خط فرمان پاورشل وارد نمائید:

```
PM> Scaffold MvcScaffolding.ActionWithUnitTest -Controller TasksController -Action ArchiveTask -
ViewModel Task
```

دستوری که در اینجا صادر شده است نسبت به حالت‌های کلی قبلی، اندکی اختصاصی‌تر است. این دستور بر روی کنترلری به نام TasksController، جهت ایجاد اکشن متدی به نام ArchiveTask با استفاده از کلاس ViewModel ایی به نام Task اجرا می‌شود.
حاصل آن ایجاد اکشن متد یاد شده به همراه کلاس TasksControllerTest است؛ البته اگر حین ایجاد پروژه جدید در ابتدای کار، گزینه ایجاد پروژه آزمون‌های واحد را نیز انتخاب کرده باشید. نام پروژه پیش فرضی که جستجوی می‌شود YourMvcProjectName.Test/Tests است.
نکته مهم آن، عدم حذف یا بازنویسی کامل کنترلر یاد شده است. کاری هم که در تولید متد آزمون واحد متناظر انجام می‌شود، تولید بدنه متد آزمون واحد به همراه تولید کدهای اولیه الگوی Arrange/Act/Assert است. پر کردن جزئیات بیشتر آن با برنامه نویس است.
و یا به صورت خلاصه‌تر:

```
PM> Scaffold UnitTest Tasks Delete
```

در اینجا متد آزمون واحد کنترلر Tasks و اکشن متد Delete آن، تولید می‌شود.

کار مقدماتی با MVC Scaffolding و امکانات مهیای در آن همینجا به پایان می‌رسد. در قسمت‌های بعد به سفارشی سازی این مجموعه خواهیم پرداخت.

نظرات خوانندگان

نویسنده: سهبلا صالح زاده
تاریخ: ۱۳۹۲/۰۶/۲۳ ۱۶:۰

در بخش EF Code First #11 عنوان کردید که مایکروسافت در تعریف DbContext اعلام می‌کند که DbSetها همان repository هستند و لایه ای دیگری ایجاد نشود، پس چرا در Scaffolding پارامتری برای آن در نظر گرفته است.

ببخشید من در استفاده از scaffolding در پروژه اصلی زمانی که کلاس‌ها را در پروژه دیگری تعریف می‌کنم مشکل دارم . خطا میدهد ولی اگر کلاس‌ها در یک پروژه تعریف شوند مشکلی ندارد.

نویسنده: سهبلا صالح زاده
تاریخ: ۱۳۹۲/۰۶/۲۳ ۱۶:۵

می‌خواستم بدونم در حالت One-to-many امکان استفاده از Html.EditForModel وجود دارد؟ یعنی میتوان بدون استفاده از UiHint ویا امثال اون فرم اتوماتیک ساخته شود و فیلدهای Dropdownlist را ایجاد کند چرا که در حالت عادی View به صورت EditForModel ساخته نشده و عناصر جدول وابسته به صورت لیست به View پاس داده می‌شود.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۲۳ ۱۶:۳۰

- لینک مطلب « پیاده سازی generic repository یک ضد الگو است » را براشون ارسال کنید تا مطالعه کنند.
- در متن عنوان شده « ModelType: برای ذکر صریح کلاس مورد استفاده در تشکیل کنترلر بکار می‌رود. اگر ذکر نشود، از نام کنترلر حدس زده خواهد شد.» ModelType دقیقا مانند نحوه مقدار دهی نوع مدل در صفحه دیالوگ استاندارد اضافه کردن یک View در VS.NET مقدار دهی می‌شود؛ یک fully qualified name است. با این شرط که اسمبلی مربوطه به پروژه اصلی ارجاع دارد و یکبار هم کل پروژه Build شده.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۲۳ ۱۶:۳۵

قسمت سوم این بحث به سفارشی سازی scaffolding پرداخته. اگر از پیش فرض‌های آن راضی نیستید یا هر تغییر خاصی را علاقمند بودید که به کلاس‌ها یا فایل‌های پیش فرض آن اعمال کنید، با سفارشی سازی قابل انجام است.

نویسنده: صالح زاده
تاریخ: ۱۳۹۲/۰۶/۲۴ ۱۶:۸

من خیلی سعی کردم اما نشد؛ مثلا کد زیر در پروژه DataLayer به درستی کار می‌کنه اما در پروژه اصلی با وجود Add شدن Refrence پروژه DataLayer کار نمی‌کنه و خطا میده.

مجبور میشم کدها را در DataLayer بسازم و بعد منتقل کنم به پروژه اصلی !

```
scaffold repository DataLayer.Models.City
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۲۴ ۱۸:۱۶

- سوئیچ ModelType رو ذکر نکردید. مثالش هست در متن (... Task *ModelType* - ...)

- خطاهایی رو هم که دریافت می‌کنید، اینجا به نویسنده اصلی گزارش بدید (به صورت کامل البته؛ نه اینکه صرفا عنوان کنید کار نمی‌کند).

عنوان: **MVC Scaffolding #3**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۴ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, MVC, MVC Scaffolding

شاید کیفیت کدهای تولیدی یا کدهای View حاصل از MVC Scaffolding مورد تائید شما نباشد. در این قسمت به نحوه تغییر و سفارشی سازی این موارد خواهیم پرداخت.

آشنایی با ساختار اصلی MVC Scaffolding

پس از نصب MVC Scaffolding از طریق NuGet به پوشه Packages مراجعه نمائید. در اینجا پوشه‌های MvcScaffolding، T4Scaffolding و T4Scaffolding.Core ساختار اصلی این بسته را تشکیل می‌دهند. برای نمونه اگر پوشه T4Scaffolding\tools را باز کنیم، شاهد تعدادی فایل ps1 خواهیم بود که همان فایل‌های پاورشل هستند. مطابق طراحی NuGet، همواره فایلی با نام init.ps1 در ابتدا اجرا خواهد شد. همچنین در اینجا پوشه‌های T4Scaffolding\tools\EFRepository و T4Scaffolding\tools\EFDbContext نیز قرار دارند که حاوی قالب‌های اولیه کدهای مرتبط با الگوی مخزن و DbContext تولیدی می‌باشند.
در پوشه MvcScaffolding\tools، ساختار قالب‌های پیش فرض تولید Viewها و کنترلرهای تولیدی قرار دارند. در اینجا به ازای هر مورد، دو نگارش vb و cs قابل مشاهده است.

سفارشی سازی قالب‌های پیش فرض Viewهای MVC Scaffolding

برای سفارشی سازی قالب‌های پیش فرض از دستور کلی زیر استفاده می‌شود:

```
Scaffold CustomTemplate Name Template
```

مانند دستور زیر:

```
Scaffold CustomTemplate View Index
```

در اینجا View نام یک Scaffolder است و Index نام قالبی در آن.
اگر دستور فوق را اجرا کنیم، فایل جدیدی به نام CodeTemplates\Scaffolders\MvcScaffolding.RazorView\Index.cs.t4 به پروژه جاری اضافه می‌شود. از این پس کلیه فرامین اجرایی، از نسخه محلی فوق بجای نمونه‌های پیش فرض استفاده خواهند کرد.
در ادامه قصد داریم اندکی این قالب پیش فرض را جهت اعمال ویژگی DisplayName به هدر جدول تولیدی نمایش اطلاعات Tasks تغییر دهیم. در کلاس Task، خاصیت زمان موعود با ویژگی DisplayName مزین شده است. این نام نمایشی حین تولید فرم‌های ثبت و ویرایش اطلاعات بکار گرفته می‌شود، اما در زمان تولید جدول اطلاعات ثبت شده، به هدر جدول اعمال نمی‌گردد.

```
[DisplayName("Due Date")]
public DateTime? DueDate { set; get; }
```

برای تغییر و بهبود این مساله، فایل Index.cs.t4 را که پیشتر به پروژه اضافه کردیم باز کنید. کلاس ModelProperty را یافته و خاصیت جدید DisplayName را به آن اضافه کنید:

```
// Describes the information about a property on the model
class ModelProperty {
    public string Name { get; set; }
    public string DisplayName { get; set; }
    public string ValueExpression { get; set; }
    public EnvDTE.CodeTypeRef Type { get; set; }
    public bool IsPrimaryKey { get; set; }
    public bool IsForeignKey { get; set; }
    public bool IsReadOnly { get; set; }
}
```

در حالت پیش فرض فقط از خاصیت Name برای تولید هدر جدول در ابتدای فایل t4 در حال ویرایش استفاده می‌شود. در پایان فایل t4 جاری، متد زیر را اضافه کنید:

```
static string GetDisplayName(EnvDTE.CodeProperty prop)
{
  var displayAttr = prop.Attributes.OfType<EnvDTE80.CodeAttribute2>().Where(x => x.FullName ==
typeof(System.ComponentModel.DisplayNameAttribute).FullName).FirstOrDefault();
  if(displayAttr == null)
  {
    return prop.Name;
  }
  return displayAttr.Value.Replace("\"","");
}
```

در اینجا بررسی می‌شود که آیا ویژگی DisplayNameAttribute بر روی خاصیت در حال بررسی وجود دارد یا خیر. اگر خیر از نام خاصیت استفاده خواهد شد و اگر بلی، مقدار ویژگی نام نمایشی استخراج شده و بازگشت داده می‌شود.
اکنون برای اعمال متد GetDisplayName، متد GetEligibleProperties را یافته و به نحو زیر تغییر دهید:

```
results.Add(new ModelProperty {
Name = prop.Name,
DisplayName = GetDisplayName(prop),
ValueExpression = "Model." + prop.Name,
Type = prop.Type,
IsPrimaryKey = Model.PrimaryKeyName == prop.Name,
IsForeignKey = ParentRelations.Any(x => x.RelationProperty == prop),
IsReadOnly = !prop.IsWriteable()
});
```

در اینجا خاصیت DisplayName به لیست خروجی اضافه شده است.
اکنون قسمت هدر جدول تولیدی را در ابتدای فایل t4 یافته و به نحو زیر تغییر می‌دهیم تا از DisplayName استفاده کند:

```
<#
List<ModelProperty> properties = GetModelProperties(Model.ViewDataType, true);
foreach (ModelProperty property in properties) {
    if (!property.IsPrimaryKey && !property.IsForeignKey) {
#>
        <th>
            <#= property.DisplayName #>
        </th>
<#
    }
}
#>
```

در ادامه برای آزمایش تغییرات فوق، دستور ذیل را صادر می‌کنیم:

```
PM> Scaffold Controller -ModelType Task -ControllerName TasksController -DbContextType TasksDbContext -
Repository -Force
```

پس از اجرای دستور، به فایل Views\Tasks\Index.cshtml مراجعه نمائید. اینبار هدر خودکار تولیدی از Due Date بجای DueDate استفاده کرده است.

سفارشی سازی قالب‌های پیش فرض کنترلرهای MVC Scaffolding

در ادامه قصد داریم کدهای الگوی مخزن تهیه شده را اندکی تغییر دهیم. برای مثال با توجه به اینکه از تزریق وابستگی‌ها استفاده خواهیم کرد، نیازی به سازنده اولیه پیش فرض کنترلر که در بالای آن ذکر شده «در صورت استفاده از یک DI این مورد را حذف کنید»، نداریم. برای این منظور دستور زیر را اجرا کنید:

```
PM> Scaffold CustomTemplate Controller ControllerWithRepository
```

در اینجا قصد ویرایش قالب پیش فرض کنترلرهای تشکیل شده با استفاده از الگوی مخزن را داریم. نام ControllerWithRepository از فایل ControllerWithRepository.cs.t4 موجود در پوشه packages\MvcScaffolding\tools\Controller گرفته شده است.

به این ترتیب فایل جدید CodeTemplates\Scaffolders\MvcScaffolding.Controller\ControllerWithRepository.cs.t4 به پروژه جاری اضافه خواهد شد. در این فایل چند سطر ذیل را یافته و سپس حذف کنید:

```
// If you are using Dependency Injection, you can delete the following constructor
        public <#= Model.ControllerName #>() : this(<#= String.Join(", ", Repositories.Values.Select(x
=> "new " + x.RepositoryTypeName + "()")) #>)
        {
        }
```

برای آزمایش آن دستور زیر را صادر نمائید:

```
PM> Scaffold Controller -ModelType Task -ControllerName TasksController -DbContextType TasksDbContext -
Repository -Force -ForceMode ControllerOnly
```

چون تنها قصد تغییر کنترلر را داریم از پارامتر ForceMode با مقدار ControllerOnly استفاده شده است.

یا اگر نیاز به تغییر کدهای الگوی مخزن مورد استفاده است می‌توان از دستور ذیل استفاده کرد:

```
Scaffold CustomScaffolder EFRepository
```

به این ترتیب فایل جدید CodeTemplates\Scaffolders\EFRepository\EFRepositoryTemplate.cs.t4 جهت ویرایش به پروژه جاری اضافه خواهد شد.

لیست Scaffolderهای مهیا با دستور Get-Scaffolder قابل مشاهده است.

نظرات خوانندگان

نویسنده: محسن عباس آباد عربی
تاریخ: ۱۳۹۱/۱۱/۰۴ ۹:۵۲

مرسی از مطلب مفیدتون
یا علی.

نویسنده: حسینی
تاریخ: ۱۳۹۱/۱۱/۲۰ ۱:۵

سلام . ممنونم از مطلب مفیدتون...
سوالی که دارم اینه که برای سفارشی کردن **MVC Scaffolding به طوری که همانند این قسمت شامل الگوی واحد باشد باید چگونه عمل کرد ؟**

http://www.dotnettips.info/post/842/ef-code-first-12

نویسنده: سعید
تاریخ: ۱۳۹۱/۱۱/۲۱ ۱۹:۵۴

در چهار سطر آخر این مقاله توضیح دادن. فایل قالب الگوی مخزن رو به پروژه اضافه کنید، بعد اون رو کمی ویرایش کرده و اینترفیس و پیاده سازی لایه سرویس رو اضافه کنید.

نویسنده: ح.م
تاریخ: ۱۳۹۲/۰۳/۰۸ ۱:۲۲

زمانی که از Scaffold CustomTemplate استفاده می‌کنیم، چنانچه در الگوهای جدید، از کلمات فارسی استفاده شود، حتی زمانی که Encoding فایلها یونی کد است(Without signature) عبارات فارسی در خروجی به هم ریخته می‌شود، برای حل مشکل در فایل Web.config تگ زیر را در قسمت system.web درج کنید:

```
<globalization fileEncoding="utf-8" requestEncoding="utf-8" responseEncoding="utf-8"/>
```

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۲/۰۹/۱۵ ۱۴:۱۹

با تشکر از مطالب خوب شما
ممکنه در مورد
سفارشی کردن MVC Scaffolding به طوری که همانند این قسمت شامل الگوی واحد باشد
توضیح بیشتری بدید؟
اینکه چگونه با ویرایش EfRepository ، میشه الگوی واحد کار رو پیاده سازی کرد.

نویسنده: رضا
تاریخ: ۱۳۹۳/۰۱/۲۲ ۲۲:۱۷

من template رو تغییر دادم و DisplayNameها جایگزین PropertyNameها میشه ، ولیکن عبارات ساده مثل "Edit" رو اگر ویرایش کنم و معادل فارسی بزارم هیچ تاثیری نداره. کسی دلیلش رو میدونه ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۲ ۲۲:۲۶

encoding را به این نحو باید تنظیم کرد:

```
<#@ output extension=".cs" encoding="utf-8" #>
```

« نحوه استفاده از Text templateها در دات نت - قسمت سوم »