

## 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; }

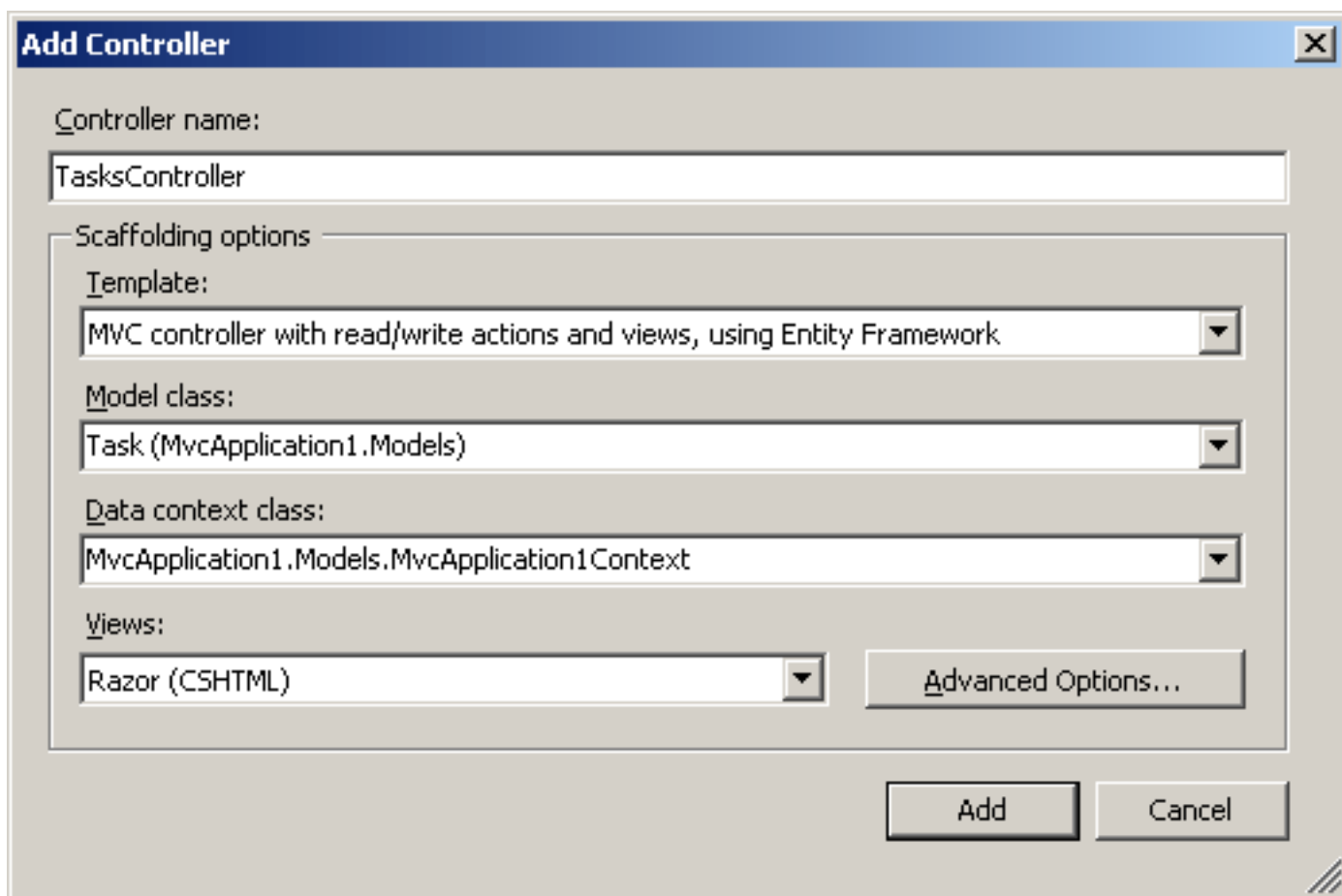
        [Required]
        public string Name { set; get; }

        [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:

A screenshot of the 'Add Controller' dialog box in Visual Studio. The dialog has a title bar with 'Add Controller' and a close button. It contains several input fields and dropdown menus. The 'Controller name' field is filled with 'TasksController'. The 'Scaffolding options' section is expanded, showing 'Template' as 'MVC controller with read/write actions and views, using Entity Framework', 'Model class' as 'Task (MvcApplication1.Models)', and 'Data context class' as 'MvcApplication1.Models.MvcApplication1Context'. The 'Views' dropdown is set to 'Razor (CSHTML)'. There is an 'Advanced Options...' button next to the Views dropdown. At the bottom right, there are 'Add' and 'Cancel' buttons.

**Add Controller**

Controller name:  
TasksController

Scaffolding options

Template:  
MVC controller with read/write actions and views, using Entity Framework

Model class:  
Task (MvcApplication1.Models)

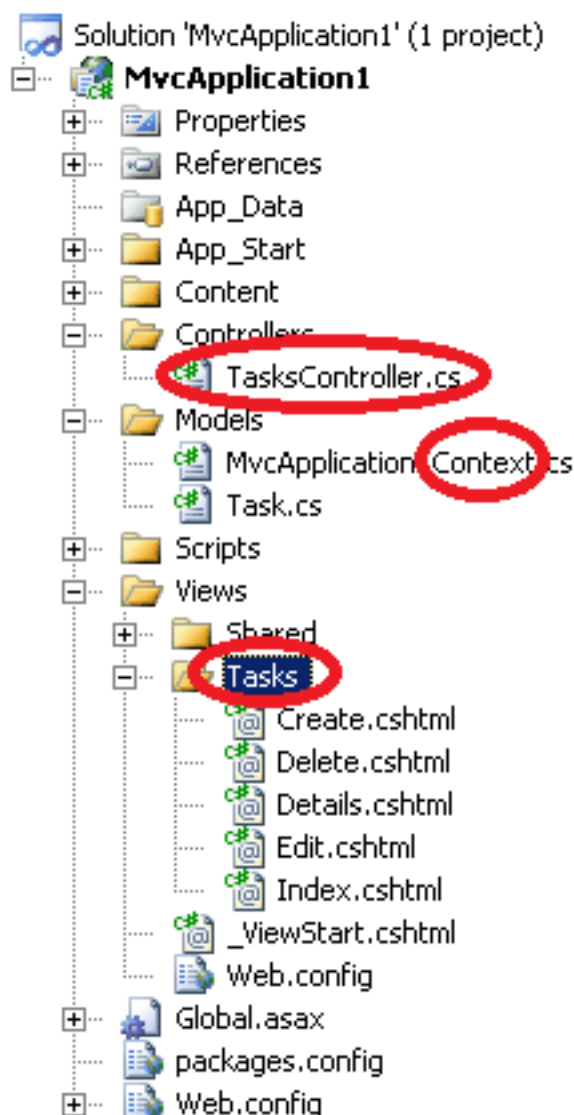
Data context class:  
MvcApplication1.Models.MvcApplication1Context

Views:  
Razor (CSHTML)

Advanced Options...

Add Cancel

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 آن را 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 MvcApplicationContext.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 بیشتر 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)

#### مغایب:

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 تنذیه 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 آن را مستقیماً در معرض دید یک View قرار داد (خصوصاً 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 نهایتاً آنچنان تفاوت عمده‌ای هم با کدهای تولیدی در اینجا نخواهند داشت. حداکثر فرم‌های آن‌را بخواهید با Query 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 مجدداً بفشارید. منویی ظاهر خواهد شد که بیانگر سوئیچ‌های قابل استفاده است.

```
PM> Install-Package MvcS
Attempting to resolve de
Attempting to resolve de
Attempting to resolve de
Successfully installed '
Successfully installed '
Successfully installed '
Successfully added 'T4Sc
Successfully added 'T4Sc
Successfully added 'MvcS

-OutVariable
-OutBuffer
-ControllerName
-ModelType
-CodeLanguage
-DbContextType
-Area
-ViewScaffolder
-Layout

plication1.
tion1.
ation1.
```

PM> Scaffold Controller -

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 مجدداً 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 بی‌کیفیت 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: ۱۳:۴۹ ۱۳۹۱/۱۱/۰۲

فوق العاده بود. اصلاً نمیدونستم که Scaffolding به همین قابلیت هایی هم داره. ممنون.

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

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

نویسنده: سعید  
تاریخ: ۱۶:۲۰ ۱۳۹۱/۱۱/۰۲

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

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

دست مریزاد. خیلی مفید بود. ممنون

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

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: ۱۷:۲۷ ۱۳۹۲/۱۲/۰۴

از EF 6 استفاده کردید؟ [بله](#) . فقط برای MVC 5 ابزار Scaffolding را جهت کار با EF 6 [به روز کرده اند](#) .