عنوان: **استفاده از خواص راهبری در EF Code first جهت ساده سازی کوئری‌ها**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۰۴ ۲۳:۴۷
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

گاهی از اوقات یافتن معادل LINQ کوئری‌های SQLایی که پیشتر به سادگی و بر اساس ممارست، در کسری از دقیقه نوشته می‌شدند، آنچنان ساده نیست. برای مثال فرض کنید یک سری پروژه وجود دارند که به ازای هر پروژه، تعدادی بازخورد ثبت شده است. هر بازخورد نیز دارای وضعیت‌هایی مانند «در حال انجام» و «انجام شد» است. می‌خواهیم کوئری LINQ سازگار با EF ایی را تهیه کنیم که تعداد موارد «در حال انجام» را نمایش دهد.
بر این اساس، کلاس‌های مدل دومین مساله به صورت زیر خواهند بود:

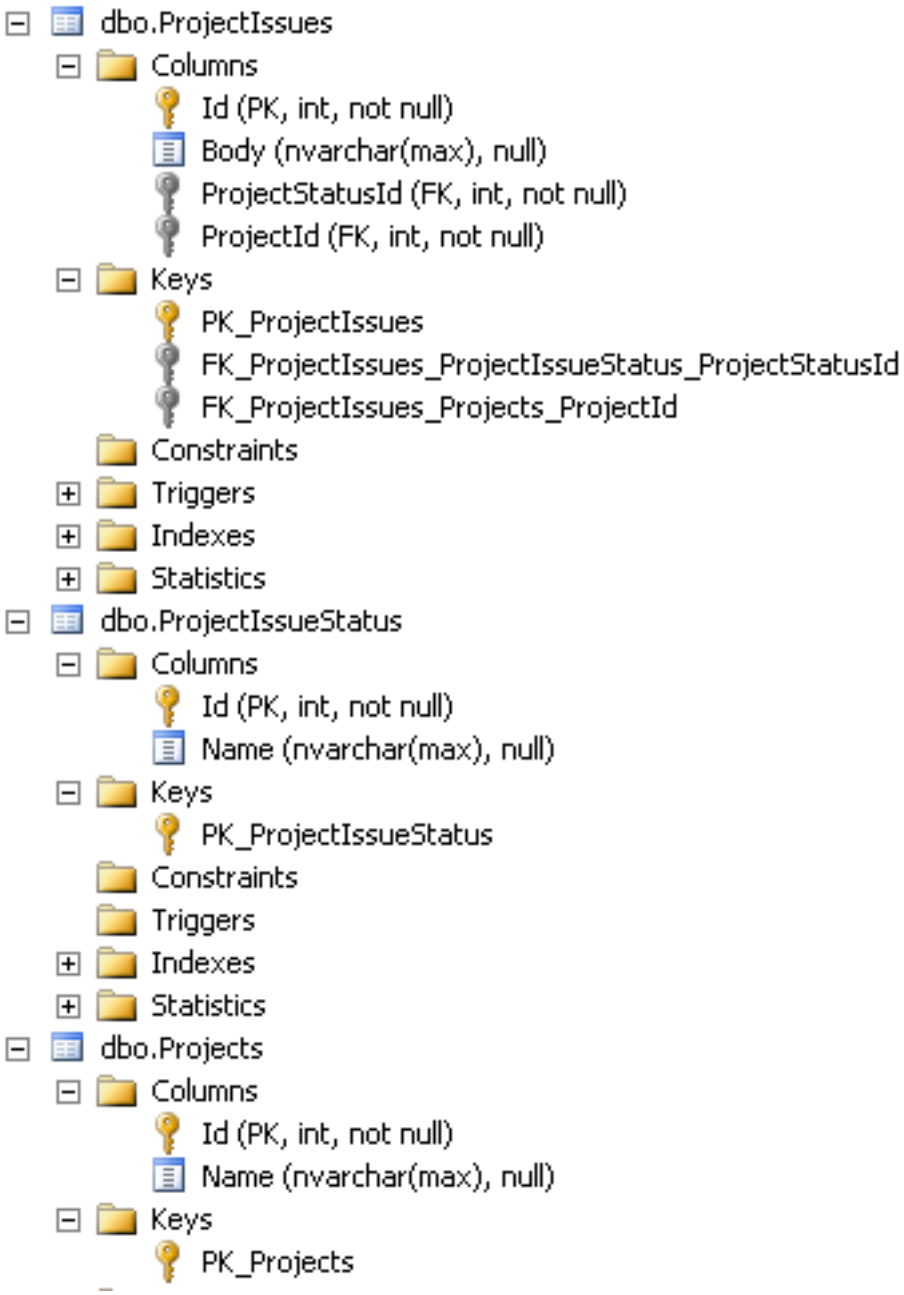
```
public class Project
    {
        public int Id { set; get; }
        public string Name { set; get; }

        public virtual ICollection<ProjectIssue> ProjectIssues { set; get; }
    }

    public class ProjectIssue
    {
        public int Id { set; get; }
        public string Body { set; get; }

        [ForeignKey("ProjectStatusId")]
        public virtual ProjectIssueStatus ProjectIssueStatus { set; get; }
        public int ProjectStatusId { set; get; }

        [ForeignKey("ProjectId")]
        public virtual Project Project { set; get; }
        public int ProjectId { set; get; }
    }

    public class ProjectIssueStatus
    {
        public int Id { set; get; }
        public string Name { set; get; }

        public virtual ICollection<ProjectIssue> ProjectIssues { set; get; }
    }
```

یک پروژه می‌تواند تعدادی Issue ثبت شده داشته باشد. هر Issue نیز دارای وضعیتی مشخص است.
اگر EF Code first را وادار به تهیه جداول و روابط معادل کلاس‌های فوق کنیم:

```
public class MyContext : DbContext
    {
        public DbSet<ProjectIssueStatus> ProjectStatus { get; set; }
        public DbSet<ProjectIssue> ProjectIssues { get; set; }
        public DbSet<Project> Projects { get; set; }
    }

    public class Configuration : DbMigrationsConfiguration<MyContext>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = true;
            AutomaticMigrationDataLossAllowed = true;
        }

        protected override void Seed(MyContext context)
        {
            var project1 = new Project { Name = "پروژه جدید" };
            context.Projects.Add(project1);

            var stat1 = new ProjectIssueStatus { Name = "درحال انجام" };
            var stat2 = new ProjectIssueStatus { Name = "انجام شد" };
            context.ProjectStatus.Add(stat1);
            context.ProjectStatus.Add(stat2);

            var issue1 = new ProjectIssue
            {
```

```
                Body = "تغییر قلم گزارش",
                ProjectIssueStatus = stat1,
                Project = project1
            };
            var issue2 = new ProjectIssue
            {
                Body = "تغییر لوگوی گزارش",
                ProjectIssueStatus = stat1,
                Project = project1
            };
            context.ProjectIssues.Add(issue1);
            context.ProjectIssues.Add(issue2);

            base.Seed(context);
        }
    }
```

به شکل زیر خواهیم رسید:



سابقا برای یافتن تعداد متناظر با هر IssueStatus خیلی سریع می‌شد چنین کوئری را نوشت:

```
SELECT [Id],
       [Name],
       InUseCount = (
           SELECT COUNT(*)
           FROM   ProjectIssues
           WHERE  ProjectStatusId = [ProjectIssueStatus].id
       )
FROM   [ProjectIssueStatus]
```

Results | Messages

	Id	Name	InUseCount
1	1	درحال انجام	2
2	2	انجام شد	0

اما اکنون معادل آن با EF Code first چیست؟

```
public static class Test
    {
        public static void RunTests()
        {
            Database.SetInitializer(new MigrateDatabaseToLatestVersion<MyContext, Configuration>());

            using (var ctx = new MyContext())
            {
                var projectId = 1;
                var list = ctx.ProjectStatus.Select(x => new
                                                    {
                                                        Id = x.Id,
                                                        Name = x.Name,
                                                        Count = x.ProjectIssues.Count(p => p.ProjectId
== projectId)
                                                    }).ToList();
                foreach (var item in list)
                    Console.WriteLine("{0}:{1}",item.Name, item.Count);
            }
        }
    }
```

بله. همانطور که ملاحظه می‌کنید در اینجا به کوئری بسیار ساده و واضحی با کمک استفاده از navigation properties (خواص راهبری مانند ProjectIssues) تعریف شده رسیده‌ایم. خروجی SQL تولید شده توسط EF نیز به صورت زیر است:

```
SELECT [Project1].[Id] AS [Id],
       [Project1].[Name] AS [Name],
       [Project1].[C1] AS [C1]
FROM   (
           SELECT [Extent1].[Id] AS [Id],
                  [Extent1].[Name] AS [Name],
                  (
                      SELECT COUNT(1) AS [A1]
                      FROM   [dbo].[ProjectIssues] AS [Extent2]
                      WHERE  ([Extent1].[Id] = [Extent2].[ProjectStatusId])
```

```
                            AND ([Extent2].[ProjectId] = 1 /*@p__linq__0*/)
                ) AS [C1]
            FROM   [dbo].[ProjectIssueStatus] AS [Extent1]
        ) AS [Project1]
```

نظرات خوانندگان

نویسنده: رضا
تاریخ: ۱۳۹۱/۰۷/۰۵ ۷:۲۱

وای، منظورتون Navigation Property بود؟ نیم ساعته دارم فکر می‌کنم خواص راهبری دیگه چیه؟

نویسنده: حسین مرادی نیا
تاریخ: ۱۳۹۱/۰۷/۰۵ ۱۶:۸

در بسیاری از مثالهای این سایت از IList استفاده کردید و در این مثال از ICollection . فرق اینها دقیقا در چیست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۰۵ ۱۷:۴۳

IEnumerable فقط خواندنی است.
ICollection یک IEnumerable است که قابلیت Add و Remove به آن اضافه شده.
IList یک ICollection است که به اعضای آن از طریق ایندکس‌ها می‌توان دسترسی اتفاقی داشت.
در تمام مثال‌های EF Code first این سایت از ICollection برای معرفی خواص راهبری استفاده شده چون EF برای انجام اعمال داخلی خودش به مجموعه‌هایی که قابلیت افزودن یا حذف عناصر را داشته باشند، نیاز دارد. به علاوه اگر به سورس EF هم مراجعه کنید برای تشخیص روابط بین کلاس‌ها به دنبال ICollection می‌گردد.
همچنین رسم است حین انتخاب اینترفیس‌هایی از این دست که از هم مشتق می‌شوند، روال انتخاب «the least specific abstraction» رعایت شود. یعنی انتخاب کوچکترین پیاده سازی با حداقل نیازهایی که کاربرد مورد نظر را برآورده می‌کند.

نویسنده: حسین مرادی نیا
تاریخ: ۱۳۹۱/۰۷/۰۶ ۰:۱۴

مرسی
خیلی کامل بود.

نویسنده: Alex
تاریخ: ۱۳۹۱/۰۷/۰۶ ۰:۲۳

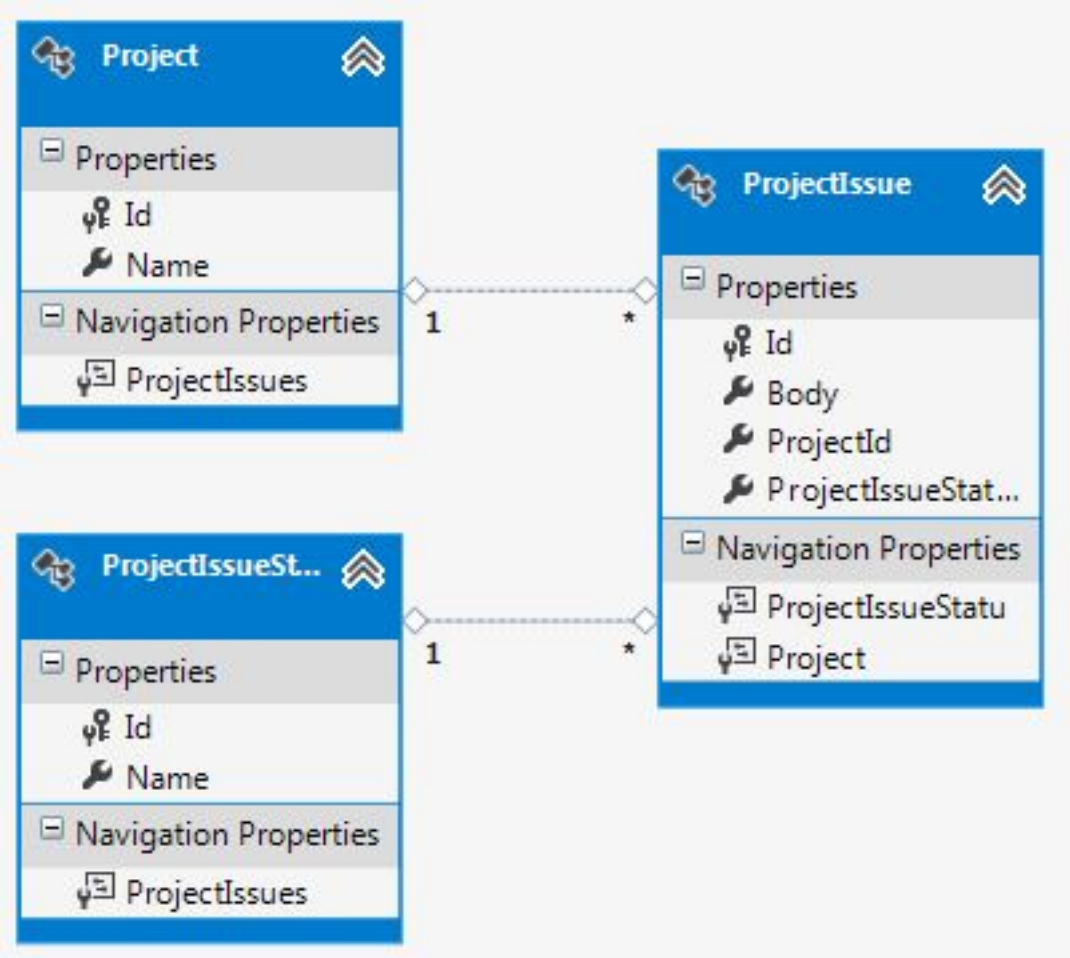
میتونم بپرسم چرا از toList استفاده کردید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۰۶ ۰:۴۵

این یک عادت خوب در EF است. زمانیکه خروجی کار شما IEnumerable باشد، هر بار دسترسی به نتیجه آن، یکبار رفت و برگشت به بانک اطلاعاتی را سبب خواهد شد. برای نمونه در مثال زیر دوبار رفت و برگشت به بانک اطلاعاتی خواهیم داشت (یکبار در حلقه اول و یکبار در حلقه دوم). اما با استفاده از ToList فقط یکبار رفت و برگشت صورت گرفته و اطلاعات اصطلاحا materialized خواهند شد.

```
var list = ctx.ProjectStatus.Select(...);

foreach (var item in list)
{...}

foreach (var item in list)
{...}
```

نویسنده: رضا بزرگی
تاریخ: ۱۳۹۱/۰۷/۰۶ ۱۸:۲۳



شمای edmxی برای درک بهتر.

نویسنده: alireza
تاریخ: ۱۳۹۱/۰۸/۰۲ ۰:۲۸

لطفا راهنمایی کنید برای اینکه ببینیم ef برای یک کوئری linq چه عبارت sql ایی تولید میکند و آن را اجرا میکند، چه کاری باید انجام داد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۰۲ ۰:۵۱

نحوه‌ی مشاهده‌ی خروجی SQL تولید شده توسط WCF RIA Services

نویسنده: ایمان باقری
تاریخ: ۱۳۹۳/۰۱/۲۷ ۱۰:۴۴

من وقتی از IList استفاده میکنم برای تعریف خواص راهبری زمانی که اون رو به یک گرید بایند میکنم AddNewRow گرید کار

نمیکنه(devExpress)با جستجو و تعریف bindingList به جای لیست مشکل حل شد.
میخواستم ببینم تعریف خواص راهبری از نوع bindingList مشکلی ندارد؟
فقط binginList متد AddRange ندارد و برای اضافه کردن چند لیست به آن باید از foreach استفاده کرد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۷ ۱۱:۱۱

این‌ها بیشتر مباحث binding دو طرفه است. تیم EF هم یک ObservableListSource را برای برنامه‌های WinForm تدارک دیده:
اینجا . برای WPF هم اینجا

نویسنده: آحمد
تاریخ: ۱۳۹۳/۰۶/۳۱ ۱۹:۴۳

با سلام
من یه سوال داشتم، این سوالم مربوط میشه به جدول هایی که توی اینجا ساختید
میخواستم بدونم بهینه‌ترین نوع دستور توی EF برای گرفتن خروجی زیر چی میتونه باشه :
گرفتن FirstName و LastName کاستومر هایی که در نقش با Name ادمین هستند!
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۳۱ ۲۰:۴۵

در انتهای مطلب « بررسی تفصیلی رابطه Many-to-Many در EF Code first » در این مورد بحث شده.