عنوان: Soft Delete در Entity Framework 6
نویسنده: م کریمی
تاریخ: ۱۳۹۳/۰۲/۳۱ ۱۰:۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

برای حذف نمودن یک رکورد از دیتابیس 2 راه وجود دارد : 1- حذف به صورت فیزیکی 2- حذف به صورت منطقی (مورد بحث این مطلب)
در حذف رکورد به صورت منطقی، طراحان دیتابیس، فیلدی را با نام‌های متفاوتی همچون IsActive , IsDeleted , Flag , و غیره، در جداول ایجاد می‌نمایند. خوب، این روش مزایا و معایب خاص خودش را دارد. مثلا شما در هر پرس و جویی که ایجاد می‌نمایید، بایستی این مورد را چک نموده و رکوردهایی را فراخوانی نمایید که فیلد IsDeleted آن برابر با false باشد. و همچنین در زمان حذف رکورد، برنامه نویس بایستی از متد Update به جای حذف فیزیکی استفاده نماید که تمام این موارد حاکی از مشکلات خاص این روش است.
در این مقاله سعی داریم که مشکلات ذکر شده در بالا را با ایجاد SoftDelete در EF 6 برطرف نماییم . *یکی از پیش نیازهای این پست مطالعه (سری آموزشی EF CodeFirst) در سایت جاری می‌باشد.
برای شروع، ما نیاز به داشتن یک Attribute برای مشخص ساختن موجودیت هایی داریم که بایستی بر روی آنها SoftDelete فعال گردد. پس برای اینکار کلاسی را به شکل زیر طراحی مینماییم:

```
using System.Data.Entity.Core.Metadata.Edm;
```

```
public class SoftDeleteAttribute : Attribute
    {
        public string ColumnName { get; set; }
        public SoftDeleteAttribute(string column)
        {
            ColumnName = column;
        }
        public static string GetSoftDeleteColumnName(EdmType type)
        {
            MetadataProperty column = type.MetadataProperties.Where(x =>
x.Name.EndsWith("customannotation:SoftDeleteColumnName")).SingleOrDefault();
            return column == null ? null : (string)column.Value;
        }
    }
```

توضیحات کد بالا: در متد سازنده، نام فیلدی را که قرار است بر روی آن SoftDelete به صورت اتوماتیک ایجاد شود، دریافت می‌نماییم و متد GetSoftDeleteColumnName در واقع با استفاده از متادیتاهایی که بر روی فیلدها وجود دارد، فیلدی که انتهای نام آن متادیتای "customannotation:SoftDeleteColumnName" را دارد، انتخاب نموده و برگشت می‌دهد.
سؤال: متادیتای "customannotation:SoftDeleteColumnName" از کجا آمد؟ برای پاسخ به این سوال کافیست ادامه‌ی مطلب را کامل مطالعه نمایید.
حال این Attribute برای استفاده در موجودیت‌های ما آمده است. برای استفاده کافیست به روش زیر عمل نمایید .

```
[SoftDelete("IsDeleted")]
    public class TblUser
    {
        [Key]
        public int TblUserID { get; set; }

        [MaxLength(30)]
        public string Name { get; set; }

        public bool IsDeleted { get; set; }
    }
```

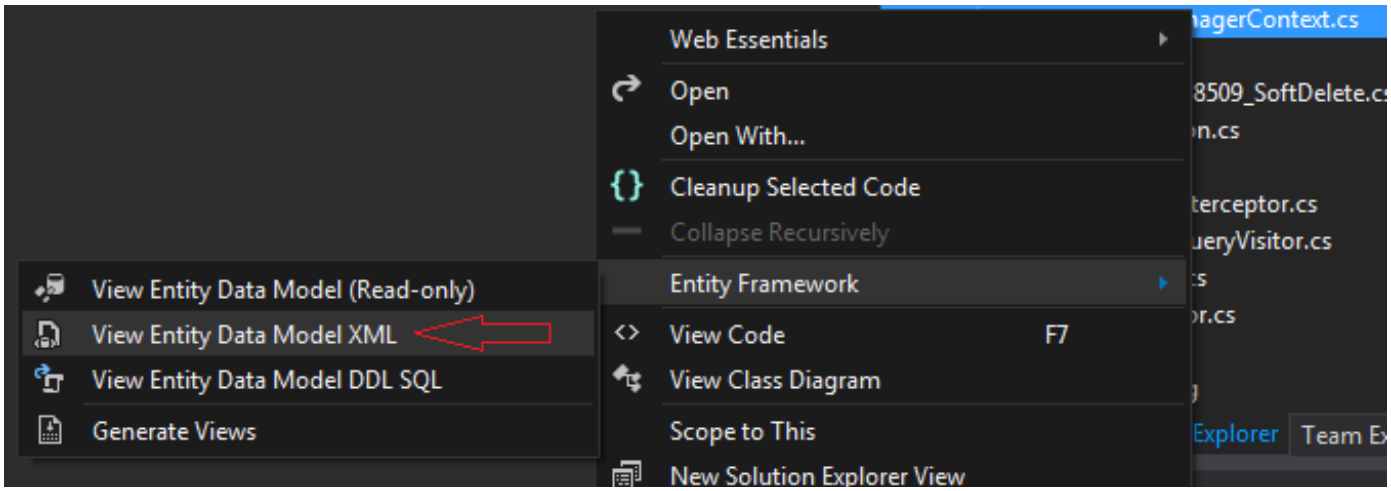
برای معرفی این قابلیت جدید به EF 6 کافیست در DbContext برنامه در متد OnModelCreating به نحو زیر عمل نماییم.

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            var Conv = new AttributeToTableAnnotationConvention<SoftDeleteAttribute, string>(
                "SoftDeleteColumnName",
```

```
                (type, attribute) => attribute.Single().ColumnName);
            modelBuilder.Conventions.Add(Conv);

        }
```

در واقع ما در اینجا به Ef می‌گوییم که یک Annotation جدید، با نام SoftDeleteColumnName به Entity که توسط این Attribute مزین شده است، اضافه نماید و همچنین مقدار این Annotation را نام فیلدی که در متد سازنده SoftDeleteAttribute معرفی گردیده است قرار دهد.
برای اطمینان حاصل کردن از اینکه آیا Annotation جدید به مدل برنامه اضافه شده است یا نه کافیست بر روی فایل cs کانتکست DbContext، کلیک راست نموده و در منوی نمایش داده شده گزینه‌ی EntityFramework و سپس گزینه View Entity Data Model را انتخاب نمایید . مانند تصویر زیر:



در پنجره باز شده به قسمت سوم یعنی <StorageModels> مراجعه نمایید و بایستی گزینه زیر را مشاهده نمایید .

```
<EntityType Name="TblUser" customannotation:SoftDeleteColumnName="IsDeleted">
```

تا اینجای کار ما توانستیم یک Annotation جدید را به Ef اضافه نماییم .

در مرحله بعد بایستی به Ef دستور دهیم که در تولید Query بر روی این Entity، این مورد را نیز لحاظ کند.

برای این کار کلاسی را ایجاد می‌نماییم که از اینترفیس IDbCommandTreeInterceptor ارث بری می‌نماید. مانند کد زیر :

```
public class SoftDeleteInterceptor : IDbCommandTreeInterceptor
    {
        public void TreeCreated(DbCommandTreeInterceptionContext interceptionContext)
        {
            if (interceptionContext.OriginalResult.DataSpace ==
System.Data.Entity.Core.Metadata.Edm.DataSpace.SSpace)
            {
                var QueryCommand = interceptionContext.Result as DbQueryCommandTree;
                if (QueryCommand != null)
                {
                    var newQuery = QueryCommand.Query.Accept(new SoftDeleteQueryVisitor());
                    interceptionContext.Result = new DbQueryCommandTree(QueryCommand.MetadataWorkspace,
QueryCommand.DataSpace, newQuery);
                }
            }
        }
}
```

در ابتدا تشخیص داده می‌شود که نوع خروجی Query آیا از نوع Storage Model است . (برای توضیحات بیشتر) سپس پرس و جوی تولید شده را با استفاده از الگوی visitor تغییر داده و Query جدید را تولید نموده و در انتها Query جدیدی را به جای Query قبلی جایگزین می‌نماییم.

در اینجا ما نیاز به داشتن کلاس SoftDeleteQueryVisitor برای تغییر دادن Query و اضافه نمودن IsDeleted <>1 به Query می‌باشیم.

یک کلاس دیگری با نام SoftDeleteQueryVisitor به شکل زیر به برنامه اضافه می‌نماییم.

```
public class SoftDeleteQueryVisitor : DefaultExpressionVisitor
    {
        public override DbExpression Visit(DbScanExpression expression)
        {
            var column = SoftDeleteAttribute.GetSoftDeleteColumnName(expression.Target.ElementType);
            if (column!=null)
            {
                var Binding = DbExpressionBuilder.Bind(expression);
                return DbExpressionBuilder.Filter(Binding,
DbExpressionBuilder.NotEqual(DbExpressionBuilder.Property(DbExpressionBuilder.Variable(Binding.Variable
Type, Binding.VariableName), column), DbExpression.FromBoolean(true)));
            }
            else
            {
                return base.Visit(expression);
            }
        }
    }
```

در متد Visit تشخیص داده می‌شود که آیا Query ساخته شده دارای customannotation:SoftDeleteColumnName است؟
چنانچه این Annotation را دارا باشد، نام فیلدی را که بالای Entity ذکر شده است، بازگشت می‌دهد و در خط بعدی، نام این فیلد را با مقدار مخالف True به Query تولید شده اضافه می‌نماید.
در نهایت برای اینکه EF تشخیص دهد که یک‌چنین Interceptor ایی وجود دارد، بایستی در کلاس DbContextConfig، کلاس SoftDeleteInterceptor را اضافه نماییم؛ همانند کد زیر:

```
public class DbContextConfig : DbConfiguration
    {
        public DbContextConfig()
        {
             AddInterceptor(new SoftDeleteInterceptor());
        }
    }
```

تا اینجا در تمام Queryهای تولید شده بر روی Entity که با خاصیت SoftDelete مزین شده است، مقدار IsDeleted <> 1 را به صورت اتوماتیک اعمال می‌نماید. حتی به صورت هوشمند چنانچه این موجودیت در یک Join استفاده شده باشد این شرط را قبل از Join به Query تولید شده اضافه می‌نماید.

در مقاله بعدی در مورد تغییر کد Remove به کد Update توضیح داده خواهد شد.

برای مطالعه بیشتر

Entity Framework: Building Applications with Entity Framework 6

نظرات خوانندگان

نویسنده: MD
تاریخ: ۱۳۹۳/۰۳/۰۴ ۰:۶

با سلام و تشکر

کسانی که از روش DB First استفاده می‌کنند می‌توانند در پنجره Mapping Details از گزینه "Add a Condition" برای پیاده سازی این گونه شرط‌ها استفاده کنند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۱۰ ۰:۲۸

راه حلی که در اینجا ارائه شد، تبدیل به کتابخانه‌ای شده به نام EntityFramework.Filters .