عنوان: **پیاده سازی UnitOfWork برای BrightStarDb**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۲/۲۶ ۲۰:۰
آدرس: www.dotnettips.info
گروه‌ها: Design patterns, NoSQL, UnitOfWork, BrightstarDB

در این پست با BrightStarDb و مفاهیم اولیه آن آشنا شدید. همان طور که پیش‌تر ذکر شد BrightStarDb از تراکنش‌ها جهت ذخیره اطلاعات پشتیبانی می‌کند. قصد داریم روش شرح داده شده در اینجا را بر روی BrightStarDb فعال کنیم. ابتدا بهتر است با روش ساخت مدل در B*Db آشنا شویم.
*یکی از پیش نیازهای این پست مطالعه این دو مطلب (^) و (^) می‌باشد.
فرض می‌کنیم در دیتابیس مورد نظر یک Store به همراه یک جدول به صورت زیر داریم:

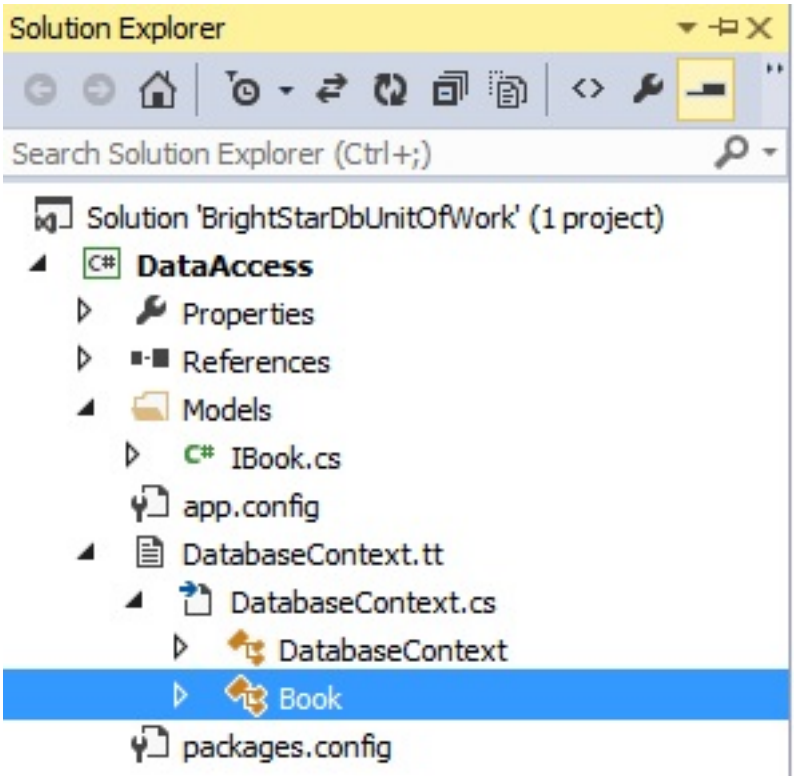
```
[Entity]
    public interface IBook
    {
        [Identifier]
        string Id { get; }

        string Title { get; set; }

        string Isbn { get; set; }
    }
```

بر روی پروژه مورد نظر کلیک راست کرده و گزینه Add new Item را انتخاب نمایید. از برگه Data گزینه BrightStar Entity Context را انتخاب کنید

بعد از انتخاب گزینه بالا یک فایل با پسوند tt به پروژه اضافه خواهد شد که وظیفه آن جستجو در اسمبلی مورد نظر و پیدا کردن تمام اینترفیس هایی که دارای EntityAttribute هستند و همچنین ایجاد کلاس‌های متناظر جهت پیاده سازی اینترفیس‌های بالا است. در نتیجه ساختار پروژه تا این جا به صورت زیر خواهد شد.



واضح است که فایلی به نام Book به عنوان پیاده سازی مدل IBook به عنوان زیر مجموعه فایل DatabaseContext.tt به پروژه اضافه شده است.

تا اینجا برای استفاده از Context مورد نظر باید به صورت زیر عمل نمود:

```
DatabaseContext context = new DatabaseContext();
  context.Books.Add(new Book());
```

Context پیش فرض ساخته شده توسط B*Db از Generic DbSetهای معادل EF پشتیبانی نمی‌کند و از طرفی IUnitOfWork مورد نظر به صورت زیر است

```
public interface IUnitOfWork
    {
        BrightstarEntitySet<T> Set<T>() where TEntity : class;
        void DeleteObject(object obj);
         void SaveChanges();
    }
```

در اینجا فقط به جای IDbSet از BrightStarDbSet استفاده شده است. همان طور که در این مقاله توضیح داده شده است، برای پیاده سازی مفهوم UnitOfWork؛ نیاز است تا کلاس DatabaseContext که نماینده BrightStarDbContext پروژه است، از اینترفیس IUnitOfWork طراحی شده ارث بری کند. جهت انجام این مهم و همچنین جهت اضافه کردن قابلیت ایجاد Generic DbSetها نیز باید کمی در فایل Template Generator تغییر ایجاد نماییم. این تغییرات را قبلا در طی یک پروژه ایجاد کرده‌ام و شما می‌توانید آن را از اینجا دریافت کنید. بعد از دانلود کافیست فایل DatabaseContext.tt مورد نظر را در پروژه خود کپی کرده و گزینه Run Custom Tools را فراخوانی نمایید.

نکته: برای حذف یک آبجکت از Store، باید از متد DeleteObject تعبیه شده در Context استفاده نماییم. در نتیجه متد مورد نظر نیز در اینترفیس بالا در نظر گرفته شده است.

استفاده از IOC Container جهت رجیستر کردن IUnitOfWrok

در این قدم باید IUnitOfWork را در یک IOC container رجیستر کرده تا در جای مناسب عملیات وهله سازی از آن میسر باشد.
من در اینجا از Castle Windsor Container استفاده کردم. کلاس زیر این کار را برای ما انجام خواهد داد:

```
public class DependencyResolver
    {
        public static void Resolve(IWindsorContainer container)
        {
            var context = new
DatabaseContext("type=embedded;storesdirectory=c:\brightstar;storename=test ");
            container.Register(Component.For<IUnitOfWork>().Instance(context).LifestyleTransient());
        }
    }
```

حال کافیست در کلاس‌های سرویس برنامه UnitOfWork رجیستر شده را به سازنده آن‌ها تزریق نماییم.

```
public class BookService
    {
        public BookService(IUnitOfWork unitOfWork)
        {
            UnitOfWork = unitOfWork;
        }

        public IUnitOfWork UnitOfWork
        {
            get;
            private set;
        }

        public IList<IBook> GetAll()
        {
            return UnitOfWork.Set<IBook>().ToList();
        }

        public void Add()
        {
            UnitOfWork.Set<IBook>().Add(new Book());
        }

        public void Remove(IBook entity)
        {
            UnitOfWork.DeleteObject(entity);
        }
    }
```

سایر موارد دقیقا معادل مدل EF آن است.
نکته: در حال حاضر امکان جداسازی مدل‌های برنامه (تعاریف اینترفیس) در قالب یک پروژه دیگر(نظیر مدل CodeFirst در EF) در B*Db امکان پذیر نیست.
نکته : برای اضافه کردن آیتم جدید به Store نیاز به وهله سازی از اینترفیس IBook داریم. کلاس Book ساخته شده توسط DatabaseContext.tt در عملیات Insert و update کاربرد خواهد داشت.