عنوان: **ایجاد سرویس چندلایه‌ی WCF با Entity Framework در قالب پروژه - 4**
نویسنده: حامد قنادی
تاریخ: ۱۳۹۲/۱۰/۲۷ ۱۱:۴۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, LINQ, WCF

برای ادامه‌ی کار به لایه‌ی Interface بازمی‌گردیم. کلیه‌ی متدهایی که به آن نیاز داریم، نخست در این لایه تعریف می‌شود. در این‌جا نیز از قراردادهایی برای تعریف کلاس و روال‌های آن بهره می‌بریم که در ادامه به آن می‌پردازیم. پیش از آن باید بررسی کنیم، برای استفاده از این دو موجودیت، به چه متدهایی نیاز داریم. من گمان می‌کنم موارد زیر برای کار ما کافی باشد:

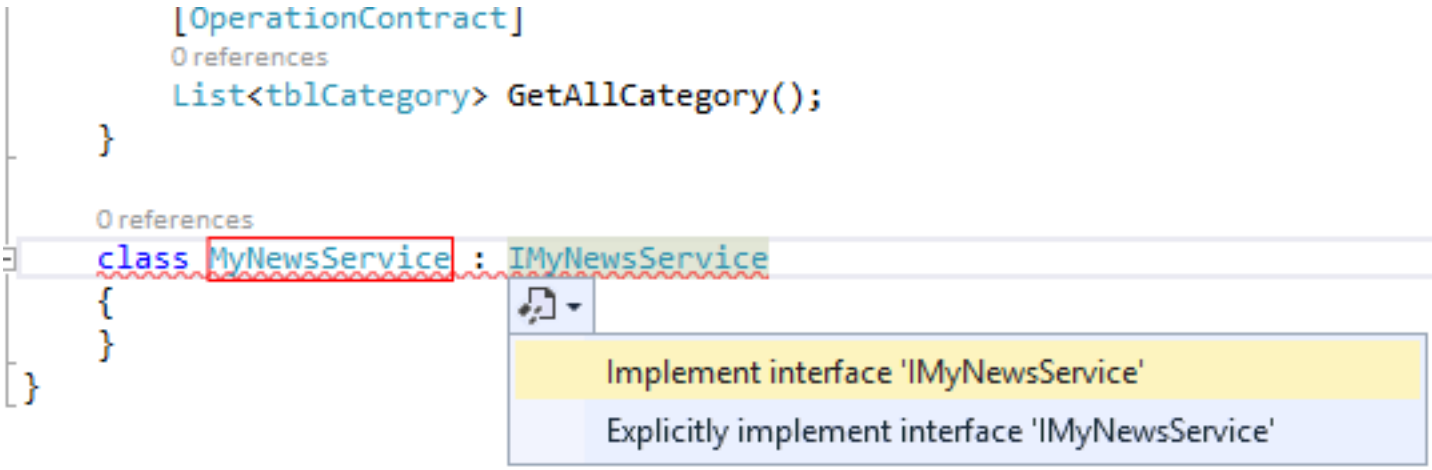
1- نمایش کلیه‌ی رکوردهای جدول خبر
2- انتخاب رکوردی از جدول خبر با پارامتر ورودی شناسه‌ی جدول خبر
3- درج یک رکورد جدید در جدول خبر
4- ویرایش یک رکورد از جدول خبر
5- حذف یک رکورد از جدول خبر
6- افزودن یک دسته
7- حذف یک دسته
8- نمایش دسته‌ها

هم‌اکنون به صورت زیر آن‌ها را تعریف کنید:

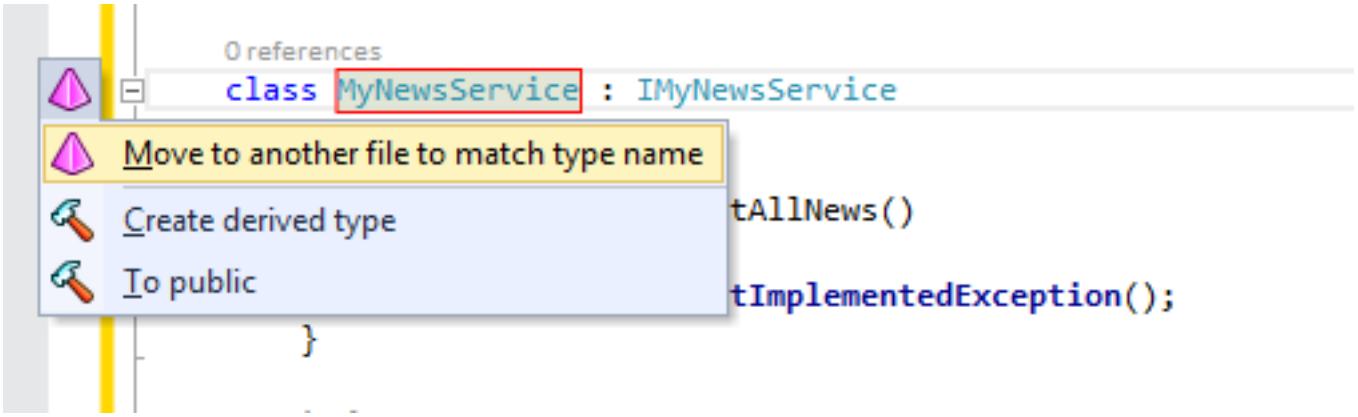
```
using System.Collections.Generic;
using System.Linq;
using System.ServiceModel;
using System.Text;
using System.Threading.Tasks;

namespace MyNewsWCFLibrary
{
    [ServiceContract]
    interface IMyNewsService
    {
        [OperationContract]
        List<tblNews> GetAllNews();

        [OperationContract]
        tblNews GetNews(int tblNewsId);

        [OperationContract]
        int AddNews(tblNews News);

        [OperationContract]
        bool EditNews(tblNews News);

        [OperationContract]
        bool DeleteNews(int tblNewsId);

        [OperationContract]
        int AddCategory(tblCategory News);

        [OperationContract]
        bool DeleteCategory(int tblCategoryId);

        [OperationContract]
        List<tblCategory> GetAllCategory();
    }
}
```

همان‌گونه که مشاهده می‌کنید از دو قرارداد جدید ServiceContract و OperationContract در فضای نام System.ServiceModel بهره برده ایم. ServiceContract صفتی است که بر روی Interface اعمال می‌شود و تعیین می‌کند که مشتری چه فعالیت‌هایی را روی سرویس می‌تواند انجام دهد و OperationContract تعیین می‌کند، چه متدهایی در اختیار قرار خواهند گرفت. برای ادامه‌ی کار نیاز است تا کلاس اجرا را ایجاد کنیم. برای این‌کار از ابزار Resharper بهره خواهم برد: روی نام interface همانند شکل کلیک کنید و سپس برابر با شکل عمل کنید:

کلاسی به نام MyNewsService با ارث‌بری از IMyNewsService ایجاد می‌شود. زیر حرف I از IMyNewsService یک خط دیده می‌شود که با کلیک روی آن برابر با شکل زیر عمل کنید:



ملاحظه خواهید کرد که کلیه‌ی متدها برابر با Interface ساخته خواهد شد. اکنون همانند شکل روی نشان هرم شکلی که هنگامی که روی نام کلاس کلیک می‌کنید، در سمت چپ نشان داده می‌شود کلیک کنید و گزینه Move to another file to match type name را انتخاب کنید:



به صورت خودکار محتوای این کلاس به یک فایل دیگر انتقال می‌یابد. اکنون هر کدام از متدها را به شکل دلخواه ویرایش می‌کنیم. من کد کلاس را این‌گونه تغییر دادم:

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;

namespace MyNewsWCFLibrary
{
    class MyNewsService : IMyNewsService
    {
        private dbMyNewsEntities dbMyNews = new dbMyNewsEntities();
        public List<tblNews> GetAllNews()
        {
            return dbMyNews.tblNews.Where(p => p.IsDeleted == false).ToList();
        }

        public tblNews GetNews(int tblNewsId)
        {
            return dbMyNews.tblNews.FirstOrDefault(p => p.tblNewsId == tblNewsId);
        }

        public int AddNews(tblNews News)
        {
            dbMyNews.tblNews.Add(News);
            dbMyNews.SaveChanges();
            return News.tblNewsId;
        }

        public bool EditNews(tblNews News)
        {
            try
            {
                dbMyNews.Entry(News).State = EntityState.Modified;
                dbMyNews.SaveChanges();
                return true;
            }
            catch (Exception exp)
            {
                return false;
            }
        }

        public bool DeleteNews(int tblNewsId)
        {
            try
            {
                tblNews News = dbMyNews.tblNews.FirstOrDefault(p => p.tblNewsId == tblNewsId);
                News.IsDeleted = true;
                dbMyNews.SaveChanges();
            return true;
            }
            catch (Exception exp)
            {
                return false;
            }
        }
```

```
        public int AddCategory(tblCategory Category)
        {
            dbMyNews.tblCategory.Add(Category);
            dbMyNews.SaveChanges();
            return Category.tblCategoryId;
        }

        public bool DeleteCategory(int tblCategoryId)
        {
            try
            {
                tblCategory Category = dbMyNews.tblCategory.FirstOrDefault(p => p.tblCategoryId ==
tblCategoryId);
                Category.IsDeleted = true;
                dbMyNews.SaveChanges();
                return true;
            }
            catch (Exception exp)
            {
                return false;
            }
        }

        public List<tblCategory> GetAllCategory()
        {
            return dbMyNews.tblCategory.Where(p => p.IsDeleted == false).ToList();
        }
    }
}
```

ولی شما ممکن است درباره‌ی حذف، دوست داشته باشید رکوردها از پایگاه داده حذف شوند و نه این‌که با یک فیلد بولی آن‌ها را مدیریت کنید. در این صورت کد شما می‌تواند این‌گونه نوشته شود:

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;

namespace MyNewsWCFLibrary
{
    class MyNewsService : IMyNewsService
    {
        private dbMyNewsEntities dbMyNews = new dbMyNewsEntities();
        public List<tblNews> GetAllNews()
        {
            return dbMyNews.tblNews.ToList();
        }

        public tblNews GetNews(int tblNewsId)
        {
            return dbMyNews.tblNews.FirstOrDefault(p => p.tblNewsId == tblNewsId);
        }

        public int AddNews(tblNews News)
        {
            dbMyNews.tblNews.Add(News);
            dbMyNews.SaveChanges();
            return News.tblNewsId;
        }

        public bool EditNews(tblNews News)
        {
            try
            {
                dbMyNews.Entry(News).State = EntityState.Modified;
                dbMyNews.SaveChanges();
                return true;
            }
            catch (Exception exp)
            {
                return false;
            }
        }

        public bool DeleteNews(tblNews News)
        {
```

```
            try
            {
                dbMyNews.tblNews.Remove(News);
                dbMyNews.SaveChanges();
            return true;
            }
            catch (Exception exp)
            {
                return false;
            }
        }

        public int AddCategory(tblCategory Category)
        {
            dbMyNews.tblCategory.Add(Category);
            dbMyNews.SaveChanges();
            return Category.tblCategoryId;
        }

        public bool DeleteCategory(tblCategory Category)
        {
            try
            {
                dbMyNews.tblCategory.Remove(Category);
                dbMyNews.SaveChanges();
                return true;
            }
            catch (Exception exp)
            {
                return false;
            }
        }

        public List<tblCategory> GetAllCategory()
        {
            return dbMyNews.tblCategory.ToList();
        }
    }
}
```

البته باید در نظر داشته باشید که در صورت هر گونه تغییر در پارامترهای ورودی، لایه‌ی Interface نیز باید تغییر کند. گونه‌ی دیگر نوشتن متد حذف خبر می‌تواند به صورت زیر باشد:

```
public bool DeleteNews(int tblNewsId)
        {
            try
            {
                tblNews News = dbMyNews.tblNews.FirstOrDefault(p => p.tblNewsId == tblNewsId);
                dbMyNews.tblNews.Remove(News);
                dbMyNews.SaveChanges();
            return true;
            }
            catch (Exception exp)
            {
                return false;
            }
        }
```

در بخش 5 درباره‌ی تغییرات App.Config خواهم نوشت.