

زمانی که از LINQ To Entity استفاده می‌کنیم، با هر بار اجرای یک کوئری، این کوئری به سمت دیتابیس ارسال شده و اطلاعات مورد نظر را بازیابی می‌کند. حال اگر ما موجودیت جدیدی را به Context جاری اضافه کرده ولی آن را ذخیره نکرده باشیم، به علت عدم وجود موجودیت در دیتابیس (در حافظه وجود دارد) کوئری ارسالی ما این موجودیت جدید را شامل نمی‌شود. البته شایان ذکر است زمانیکه از متد Find استفاده می‌کنیم، به صورت پیش فرض ابتدا داخل حافظه کاوش شده و در صورت عدم وجود اطلاعات، کوئری را در دیتابیس اجرا می‌کند.

نکته قابل توجه این است که بعنوان مثال ما نیاز داریم یک لیست از موجودیت‌ها را به اشکال زیر داشته باشیم. به صورت لیست

به صورت لیست sort شده براساس Name

فیلتر بر روی لیستی از فیلدهای موجود

این لیست باید شامل تمامی داده‌های موجود (چه در رم و چه در دیتابیس) باشد.

نکته: توسط متد ToList میتوان به لیستی از موجودیت‌های مورد نظر دست یافت ولی امکان استفاده از تمامی داده‌های موجود (چه در رم و چه در دیتابیس) میسر نمی‌باشد.

کلاس DbSet خاصیتی به نام Local دارد که امکان استفاده از تمامی داده‌های موجود را به ما می‌دهد و شامل هر داده‌ای که از دیتابیس Load شده، هر داده‌ای که اضافه شده، هر داده‌ای که پاک شده (Delete Flag) ولی هنوز ذخیره نشده می‌شود. بنابراین در هنگام استفاده باید توجه داشت به علت اینکه هیچ نوع کوئری به دیتابیس ارسال نشده، قطعه کد زیر دارای مقدار Destinations in memory: 0 خواهد بود

```
private static void GetLocalDestinationCount()
{
    using (var context = new BreakAwayContext())
    {
        var count = context.Destinations.Local.Count;
        Console.WriteLine("Destinations in memory: {0}", count);
    }
}
```

استفاده از متد Load

برای مثال می‌توانیم از Foreach استفاده کنیم و تمام اطلاعات مورد نظر را بدست آوریم

```
private static void GetLocalDestinationCount()
{
    using (var context = new BreakAwayContext())
    {
        foreach (var destination in context.Destinations)
        {
            Console.WriteLine(destination.Name);
        }
        var count = context.Destinations.Local.Count;
        Console.WriteLine("Destinations in memory: {0}", count);
    }
}
```

کد بالا یک حلقه بر روی موجودیت‌های Destinations است که نیازی به توضیح خاصی ندارد. حال با استفاده از متد Load قادر به جمع آوری اطلاعات دیتابیس به داخل رم نیز خواهیم بود و کد بالا تمیزتر خواهد شد.

```
private static void GetLocalDestinationCountWithLoad()
{
    using (var context = new BreakAwayContext())
    {
        context.Destinations.Load();
        var count = context.Destinations.Local.Count;
        Console.WriteLine("Destinations in memory: {0}", count);
    }
}
```

متد Load یک extension method روی $IQueryable<T>$ است که در فضای نام System.Data.Entity موجود است. پس امکان اجرای یک LINQ query و سپس Load کردن آن را در حافظه را خواهیم داشت. به کد زیر توجه کنید:

```
private static void LoadAustralianDestinations()
{
    using (var context = new BreakAwayContext())
    {
        var query = from d in context.Destinations
                     where d.Country == "Australia"
                     select d;
        query.Load();
        var count = context.Destinations.Local.Count;
        Console.WriteLine("Aussie destinations in memory: {0}", count);
    }
}
```

همچنین ما قادر به استفاده از LINQ query روی داده‌های Local که این متد در حافظه جمع آوری کرده، نیز خواهیم بود.

به کد زیر توجه کنید.

```
private static void LocalLinqQueries()
{
    using (var context = new BreakAwayContext())
    {
        context.Destinations.Load();
        var sortedDestinations = from d in context.Destinations.Local
                                  orderby d.Name
                                  select d;
        Console.WriteLine("All Destinations:");
        foreach (var destination in sortedDestinations)
        {
            Console.WriteLine(destination.Name);
        }
        var aussieDestinations = from d in context.Destinations.Local
                                  where d.Country == "Australia"
                                  select d;
        Console.WriteLine();
        Console.WriteLine("Australian Destinations:");
        foreach (var destination in aussieDestinations)
        {
            Console.WriteLine(destination.Name);
        }
    }
}
```

ابتدا کلیه داده‌ها Load می‌شود سپس به وسیله Foreach نام‌ها استخراج می‌شوند. سپس از همان داده‌ها جهت اعمال فیلتر، استفاده می‌شود.

تفاوت بین Linq provider های مختلف:

عموماً دیتابیس‌ها حساس به حروف کوچک و بزرگ نیستند؛ به عنوان مثال اگر Great و great در دیتابیس وجود داشته باشند اگر به دیتابیس کوئری ارسال شود و درخواست great داشته باشد هر دو را شامل می‌شود. حال اگر از Local استفاده شود به جهت اینکه در واقع از Linq to Object استفاده می‌کند فقط great را شامل خواهد شد. تفاوت دیگر این است که Linq to Object از متد Last پشتیبانی می‌کند ولی Linq to Entities خیر. در پست بعدی قصد دارم در مورد ObservableCollection توضیحاتی کلی بدهم.

در مبحث [Entity Framework در Local خاصیت](#) ملاحظه نمودید که خاصیت Local به راحتی می‌تواند از رفت و آمدهای بی جهت به دیتابیس جلوگیری کند. حال قصد معرفی یک collection را به نام ObservableCollection داریم. همانطور که از نامش پیداست برای مشاهده و تحت نظر قرار دادن داده‌های اضافه شده یا پاک شده کاربرد دارد. به کد زیر دقت کنید.

```
private static void ListenToLocalChanges()
{
    using (var context = new BreakAwayContext())
    {
        context.Destinations.Local.CollectionChanged += (sender, args) =>
        {
            if (args.NewItems != null)
            {
                foreach (Destination item in args.NewItems)
                {
                    Console.WriteLine("Added: " + item.Name);
                }
            }
            if (args.OldItems != null)
            {
                foreach (Destination item in args.OldItems)
                {
                    Console.WriteLine("Removed: " + item.Name);
                }
            }
        };
        context.Destinations.Load();
    }
}
```

در بالا به وسیله یک event handler جدید به collection محلی ما (Local) نظر می‌اندازد و در صورت اضافه شدن یا حذف موجودیتی، آن را به ما نشان می‌دهد. فقط توجه کنید که اگر نیاز دارید در صفحه‌ای این تغییرات را مشاهده کنید باید عمل Refresh کردن صفحه را چه به صورت دستی یا با نوشتن کد خودتان مدیریت کنید. البته با استفاده از WPF میتوان (استفاده از کنترل‌های مانند ListBox) این کار را به صورت خودکار انجام داد.

نظرات خوانندگان

نویسنده: محمد زارع
تاریخ: ۱۴۰۲/۰۵/۱۳ ۱۴:۰۰

سلام آقای جعفری. ممنون از مطلب مفیدتون. شاید سوالم خیلی ربطی به مطلب شما نداشته باشه ولی میخوام ازتون بپرسم که ما وقتی توی پروژه WPF از Entity Framework Code First استفاده میکنیم باید اینترفیس INotifyPropertyChanged رو برای Entity پیاده سازی کنیم یا نه؟! هر

نویسنده: وحید نصیری
تاریخ: ۱۴۰۲/۰۵/۱۳ ۸:۱۵

- بله. باید اینکار را انجام دهید.
- استفاده مستقیم از مدل‌های EF در WPF یا MVC یا هر جای دیگری کار توصیه شده‌ای نیست.
View شما باید Model مخصوص به خود را داشته باشد و این مدل الزاماً با موجودیت‌های بانک اطلاعاتی شما یکی نیست. برای نگاشت اطلاعات مدل یک View به مدل داده‌ای می‌شود از کتابخانه‌هایی مانند Auto-Mapper استفاده کرد.

نویسنده: ایلیا
تاریخ: ۱۴۰۲/۰۶/۱۳ ۱۴:۵۹

با سلام.
در پروژه WPF در لایه سرویس یکبار Local را بر میگردانم مانند زیر :

```
public override IList<City> GetAll()
{
    var query = from item in _tEntities
                select item;
    query.Load();
    return _tEntities.Local;
}
```

همه چیز درست است ولی وقتی برای جستجو متد زیر را اجرا می‌کنم باز Local شامل همان داده‌های قبلی است:

```
public override IList<City> GetAll(Func<City, bool> predicate)
{
    var query = from item in _tEntities
                select item;
    query.Where<City>(predicate);
    query.Load();
    return _tEntities.Local;
}
```

لطفاً راهنمایی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۴۰۲/۰۶/۱۳ ۱۶:۱۴

[در مثال رسمی](#) EF Code first و WPF یک سری Refresh هم هست که باید وقت بگذارید و مطالعه کنید.

نویسنده: kianush
تاریخ: ۱۴۰۳/۰۸/۱۳ ۱۳:۲۹

من در winform یک BindingSource دارم و این رو به گرید می‌دم. گرید به یک BindingSource بایند شده، اگه مقادیری رو تغییر بدم یا اضافه کنم در گرید (در واقع به BindingSource پشت صحنه) ، ChangeTracker تشخیص می‌ده و می‌تونم کار رو هندل کنم

اما اگر سطری رو از BindingSource (یا همون گرید) حذف کنم ChangeTracker متوجه نمی‌شه و وضعیتشون Unchanged می‌مونه! (Local هم فایده نداره) و نمی‌تونم کل تغییرات رو ذخیره کنم با صدا زدن SaveChanges در ChangeTracker ایتم‌هایی با State‌های Add, Modified رو می‌تونم ببینم ولی اگر ایتمی رو Delete کنم نمی‌تونم ببینمش و کاری انجام بدم! مشکل از کجاست؟

اینجا مربوط به Context هست برای بررسی وضعیت تغییرات:

```
public bool HasChanges()
{
    return this.ChangeTracker.Entries().Any(e => e.State != EntityState.Unchanged);
}

public void RejectChanges()
{
    foreach (var entry in this.ChangeTracker.Entries())
    {
        switch (entry.State)
        {
            case EntityState.Modified:
                entry.State = EntityState.Unchanged;
                break;

            case EntityState.Added:
                entry.State = EntityState.Detached;
                break;
        }
    }
}
```

توضیحات بیشتر:

چرا مثل Add, Modify عمل Delete در Entity ها و ChangeTracker بصورت خودکار شنیده نمی‌شه؟! من یک کتابخانه‌ی مستقل تهیه کردم که گرید، خودش یک کنترل BindingSource داره و اصلاً نباید به EntityFramework و سرویس‌هایی که با Entity ها کار می‌کنن در ارتباط باشه، BindingSource باید بتونه CurrentItem اش رو Remove کنه در حافظه مثل کاری که برای Add, Update انجام می‌ده و در انتها من (بعنوان کاربر نهایی) در لایه نمایش تصمیم بگیرم که تمام تغییرات انجام شده در گرید (Add, Modify, Delete) رو ذخیره یا لغو کنم.

***** شاید بشه با کمک رویداد BindingSource.ListChanged به نحوی حل کرد ولی اصلاً جالب نمی‌شه چون BindingSource من یک POCO Entity هست و فقط و فقط چنتا property داره و اصلاً از وضعیت خودش خبر نداره و قرار هم نیست بیشتر از این باشه و من نمی‌تونم وضعیتش رو تغییر بدم در این رویداد چون همچین چیزی نداره اصلاً!

نویسنده: وحید نصیری

تاریخ: ۱۴:۰۱/۰۸/۱۳۹۱

- در مورد Tracking API [یک مطلب جداگانه](#) در سایت هست. Tracking API همان ObservableCollection نیست. Tracking API در سایر ORM ها نامی به شکل سطح اول کش دارد (first level caching).
- با توجه به اینکه برای بررسی کارهای شخصی و کتابخانه‌های مستقل، نیاز به کد کامل هست، بهتر است به مقاله زیر مراجعه کرده و جزئیات کار خودتان را با آن مقایسه کنید:

« [Implementing Undo/Redo feature for DbContext of Entity Framework](#) »

نویسنده: kianush

تاریخ: ۱۴:۰۳/۰۸/۱۳۹۱

ممنون.. بررسی می‌کنم ببینم می‌تونم اصولی حل کنم این مسئله رو یا خیر :-
ولی یه نکته، اینکه گفتین "نیاز به کد کامل هست.." اصلاً تصور کنین کتابخانه‌ی مستقلی نیست. مثلاً یک Form, BindingSource, DataGridView رو داشته باشین و روال بالا که توضیحشو دادم. انگار یک Bug هست! یجورایی که وضعیت "حذف" رو مثل "افزوده شدن" و "تغییر کرده" نمی‌تونه اعلام کنه به دیتاسورس پشت سرش bindingsource

نویسنده: وحید نصیری
تاریخ: ۱۴:۲۳ ۱۳۹۱/۰۸/۰۳

- Tracking API فقط داخل یک Context مفهوم پیدا می‌کند نه مجزای آن.
- همچنین این API دارای متد [DetectChanges](#) هم هست که می‌شود به صورت دستی جهت اطمینان بیشتر آن‌را در هر زمانی (مثلا داخل بررسی HasChanges) فراخوانی کرد.

نویسنده: kianush
تاریخ: ۱۶:۶ ۱۳۹۱/۰۸/۰۳

برای این مسئله‌ی من راه حل اصولی ای پیدا نکردم. یه راهی که الان پیاده کردم و جواب گرفتم ولی جالب نیست: در BaseEntity پراپرتی IsDeleted رو کار گذاشتم مثلا یه همچین چیزی فک کنین:

```
public abstract class BaseEntity
{
    [ColumnInfo("کد",pWidth:70)]
    public int Id { get; set; }

    [ColumnInfo("",pIsVisible:false,pIsEditable:false)]
    [NotMapped]
    public bool IsDeleted { get; set; }
}
```

و جایی که BindingSource CurrentItem پاک می‌شه , BindingSource.Current.IsDeleted=true (بصورت dynamic) گذاشتم و در 2 3 , Context خط دیگه اضافه کردم که این رو هندل کنم برای تمام موجودیت ها.. کار می‌کنه ولی بدیش اینه که یک پراپرتی بی ربط (شاید به نوعی) رو در BaseEntity و در واقع در تمام موجودیت‌هام تعریف کردم (که البته NotMapped هست) و "رفتار" رو با "خاصیت" قاطی کردم و الان هم عذاب وجدان دارم :دی.پ.ن: کماکان دنبال راهی می‌گردم با خوندن مقالات

همان طور که می‌دانید نسخه 5 (نهایی) از EF به همراه Visual Studio 2012 منتشر خواهد شد ([...](#)) و قابلیت‌های کلیدی افزوده شده به آن عبارتند از:

پشتیبانی از Enum در هر سه حالت (Database First, Code First, Model First)
 پشتیبانی از Tabel-valued Function در حالت Database First
 پشتیبانی از داده‌های جغرافیایی در هر سه حالت (Database First, Code First, Model First)
 افزایش کارایی قابل توجه در LINQ To Entites و Entity SQL ([...](#))

قابلیت داشتن چند دیاگرام برای یک مدل
 قابلیت ایمپورت دسته ای Stored Procedure ها
 شاید این بهبودها کم به نظر برسند ولی اتفاق مهم دیگری که رخ داده متن باز شدن کامل EF است (قبلا در 4.1 متن باز شده بود)
 که در این آدرس نه تنها می‌توانید ([...](#)) به سورس کدها دسترسی پیدا کنید بلکه می‌توانید در تکمیل پروژه و رفع نواقص آن نیز شرکت کنید. ([...](#))
 بنابراین روند توسعه EF از این پس کاملاً قابل پیگیری (و شاید قابل تغییر) است. ([...](#))

قابلیت‌های جدیدی که برای EF نسخه 6 در نظر گرفته شده اند عبارتند از:

بهره گیری از قابلیت async در دات نت 4.5 و معرفی Async Query & Update

```

public async Task<Store> FindClosestStore(DbGeography location)
{
    using (var context = new StoreContext())
    {
        return await (from s in context.Stores
                      orderby s.Location.Distance(location)
                      select s).FirstAsync();
    }
}
    
```

پشتیبانی از نگاشت Stored Procedure و Function در حالت Code First
 پشتیبانی از Code First conventions سفارشی (یک کاربرد آن برای جلوگیری از حجم زیاد کد نویسی در هنگام تولید مدل
 (...)) (OnModelCreating)

نظرات خوانندگان

نویسنده: رضا.ب

تاریخ: ۱۳۹۱/۰۵/۰۲ ۲:۲۶

اگه میشد مطالب مرجع سایت که غالبا مهندس نصیری نوشتند رو همزمان با این تغییرات نهایی تغییر داد خیلی کار جالب توجهی میشد.

مثلا یا به حالت ویکی که بشه نظارت کرد رو ورژن‌های مختلفی که به‌روزرسانی شدند. یا در همچین‌جور پست‌هایی با اشاره به قابلیت جدید و یا منسوخ شده‌ی مطالب مرجع.
یه سوال؛ آیا انتظار درستی که مرز حاضر بین ORM و رابط‌های اشیاء دیتابیس‌های NoSQL رو حذف کرد و به یه اتحاد واحد رسید. که مثلا از EF به عنوان یه روش کلی برای ارتباط با "منبع داده‌ای" یاد شود؟ چراکه همکنون ORM نقشی در NoSQL‌ها ندارند.

نویسنده: سیروان عفیفی

تاریخ: ۱۳۹۱/۰۵/۰۲ ۹:۴۳

دیگه باید شاهد رشد سریع EF باشیم.

نویسنده: علیرضا صالحی

تاریخ: ۱۳۹۱/۰۵/۰۲ ۹:۴۵

در مورد داشتن یک ORM که هم با NoSQL‌ها کار کند و هم با RDBMS‌ها چند نکته وجود دارد، اول این که خود NoSQL‌ها خیلی با هم سازگاری ندارند، روش ذخیره سازی، مدل ذخیره سازی و ...
دوم اینکه به طور کلی طرز تفکر و مورد استفاده و شکل Query‌هایی که همه ما در RDBMS‌ها به آن عادت داریم در NoSQL جاری نیست. بنابراین داشتن ORM ی که هر دو را پوشش دهد شاید منطقی به نظر نرسد.
در اینجا بحث خوبی در این زمینه انجام شده

نویسنده: مهدی پایروند

تاریخ: ۱۳۹۱/۰۵/۰۲ ۱۲:۵۲

هرجا صحبت از تفاوت و مزایای این دو ORM یعنی NH و EF پیش میاد، اولین تفاوت پشتیبانی NHibernate از کش لایه دوم هست.

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۰۵/۰۲ ۱۳:۲۰

یک EFCachingProvider رو می‌تونید اینجا ملاحظه کنید: ([^](#))
همچنین من [از این روش](#) راضی هستم.

نویسنده: مرتضی

تاریخ: ۱۳۹۱/۰۵/۲۰ ۳:۳۱

سلام

EF نسخه 6 از Net 4.0 با وجود Async پشتیبانی می‌کنه؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۰۵/۲۰ ۹:۳۰

5 EF به بعد [بر مبنای](#) دات نت 4 و نیم است.
ویندوز 8 دات نت 4 و نیم سر خود است.
از دیدگاه تیم BCL، دات نت 4 و نیم یک [به روز رسانی درجای](#) دات نت 4 است و صد در صد با آن سازگاری دارد.

دات نت 4 و نیم فقط بر روی ویندوزهای ویستا سرویس پک 2 به بعد [قابل نصب است](#) (روی XP یا ویندوز سرور 2003 نصب نمی‌شود).

نویسنده: ایمان محمدی
تاریخ: ۱۱:۴۹ ۱۳۹۱/۰۵/۲۰

این که روی xp نصب نشه خیلی ناجوره عملا تو بخش application تا مدت‌ها بی استفاده می‌مونه ، بنظرتون این یک اهرم فشار برای حذف xp و سرور 2003 هست یا از لحاظ فنی جوابگو نبودن؟

نویسنده: علیرضا صالحی
تاریخ: ۱۵:۱۸ ۱۳۹۱/۰۵/۲۰

ویندوز ایکس پی در حال حذف شدن، هر چند خیلی‌ها هنوز در حال استفاده از اون هستند، ولی سرعت آپگرید کردن به 7 در حال زیاد شدن.

البته در ایران که هنوز سازمانهایی مانند تامین اجتماعی با فلاپی درایو سر و کار دارند، یک مقداری این مسئله مشکل ساز میشه.

نویسنده: ایمان محمدی
تاریخ: ۱۶:۴۶ ۱۳۹۱/۰۵/۲۰

دو گل سرسبد ایران یکی آموزش و پرورش یکی تامین اجتماعی که علاقتشون به foxpro و فلاپی تموم نمیشه ،ولی در نظر بگیرید به مشتری تون بگید (سازمانی یا عمومی) نرم افزار روی ویندوز xp نصب نمیشه! خودمم باشم قبول نمی‌کنم. با اینکه تمام سیستم‌ها رو معمولا به سون ارتقا میدیم ولی بعضی وقت‌ها سیستم قدیمیه و نمی‌کشه روش سون نصب کرد. نکته ای دیگه ای که وجود داره همه دنیا مثل ما پیشرفته و پول دار نیستند که روی همه کامپیوتر هاشون ویندوز سون ultimate نصب کنند.
پ.ن : [Make .NET 4.5 work on any OS that supports 4.0](#)

نویسنده: وحید نصیری
تاریخ: ۰:۲۴ ۱۳۹۱/۰۵/۲۶

EF 5 امروز [منتشر شد](#) و نکته مهم آن این است که با دات نت 4 هم سازگاری دارد. در دو نسخه دات نت 4 و دات نت 4 و نیم تهیه شده است.

البته اکثر قابلیت‌های جدید آن مخصوص دات نت 4 و نیم است مانند:

enum support
spatial data types
table-valued functions

نویسنده: محمد رضا کارونی
تاریخ: ۹:۲۵ ۱۳۹۱/۰۵/۲۶

سلام جناب مهندس نصیری،

می‌خواستم بدونم EF5 و MVC4 در نسخه‌های Express و ویژوال استودیو قابل نصب و بکارگیری می‌باشند یا خیر؟
در کل مایکروسافت برای ترویج عموم توسعه دهندگان به نوشتن app بر روی ویندوز 8 تا چه میزان بر روی نسخه‌های Express و ویژوال استودیو سرمایه گذاری و آینده نگری می‌کند؟

نویسنده: وحید نصیری
تاریخ: ۹:۳۵ ۱۳۹۱/۰۵/۲۶

EF5 چندتا DLL بیشتر نیست. این‌ها رو دریافت و به پروژه خودتون اضافه کنید.

- نسخه express مخصوص vs2012 هم موجود است ([^](#)).

نویسنده: اژدری
تاریخ: ۱۳۹۱/۰۶/۱۳ ۱۰:۴۳

بسیار هم عالی

Entity framework 5 نسبت به نسخه‌های پیشین شاهد تغییرات بسیاری بوده است و مانند هر تغییر دیگری اینجا نیز ممکن است تغییرات ؛ باعث بروز مشکلاتی در روند توسعه نرم افزار شوند. EF در نسخه جدید خود در کدهای پشت صحنه Model به جایObjectContext از DbContext که نسخه محدود شدهObjectContext می‌باشد استفاده می‌کند. همین امر به خودی خود باعث محدود شدن متدهای شئی Context شده است. متدها و خواصی که گاهی برای انجام اعمال خاصی به آنها نیاز پیدا می‌کنیم ولی دیگر در دسترس نیستند. برای مثال برای یک برنامه خاص می‌خواستیم مقدار CommandTimeout را به صورت دستی برای شئی Context تنظیم کنیم ؛ ولی کد زیر دیگر قابل استفاده نبود:

```
context.CommandTimeout = 180;
```

همچنین این استفاده از DbContext در هنگام استفاده از WCF Ria در سیلورلایت باعث بروز مشکل شده و کلاس‌های مدل در هنگام تعریف Domain Service Class توسط WCF Ria قابل شناسایی نیستند. یعنی WCF RIA به صورت خودکار قادر به تشخیص کلاس‌های Model نمی‌باشد.

×

Add New Domain Service Class

Domain Service class name:

☒ Enable client access
☐ Expose OData endpoint

Available context classes:

Some Entity Framework context classes may have been excluded.
[More information can be found in the knowledge base.](#)

Entities	Enable editing

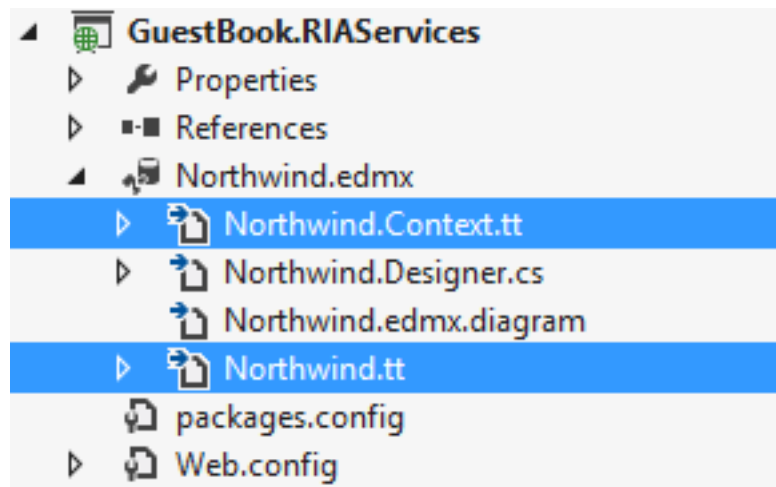
☐ Generate associated classes for metadata

OK

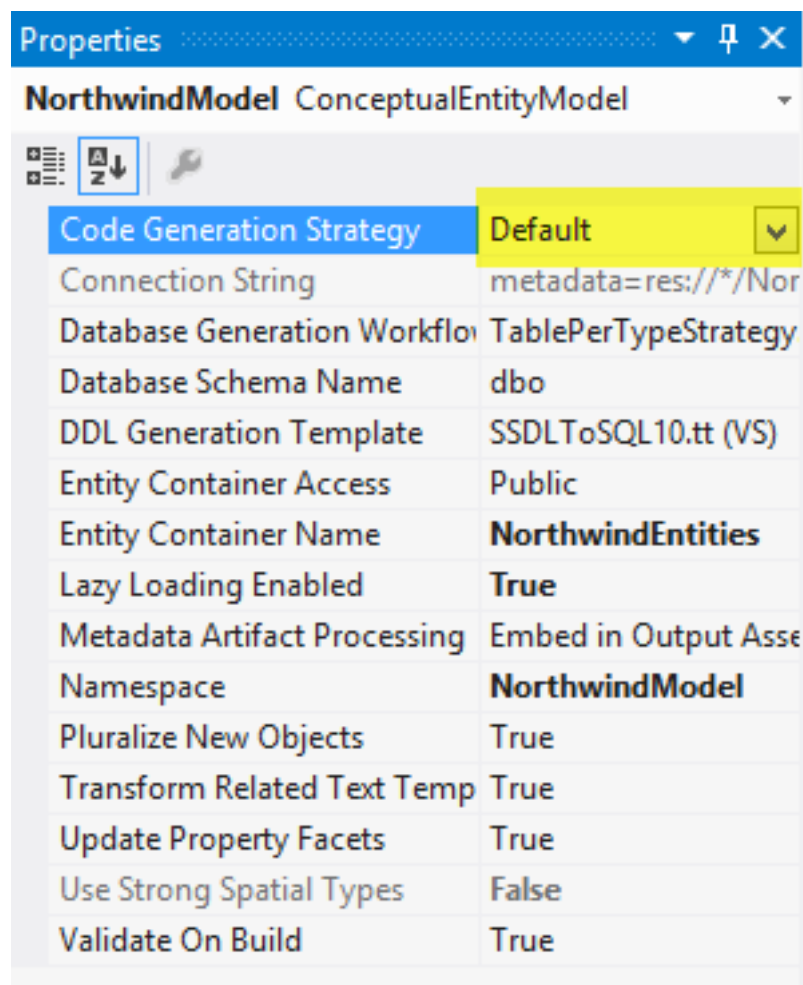
Cancel

برای رفع این مشکل مراحل زیر را انجام دهید:

دو فایل tt موجود در مدل را حذف نمایید.



2. مدل را در حالت Designer باز کنید و در بخش خصوصیات مدل مقدار Code Generation Strategy را از None به Default تغییر دهید.



3. پروژه را Rebuild نمایید. مشکل به همین سادگی رفع می‌شود.
حالا با خیال راحت می‌توانید کلاس‌های مدل را در پنجره Add New Domain Service Class مشاهده نمایید.

Add New Domain Service Class

Domain Service class name:

☒ Enable client access
☒ Expose OData endpoint

Available context classes:

Some Entity Framework context classes may have been excluded.
[More information can be found in the knowledge base.](#)

Entities	Enable editing
<input type="checkbox"/> Category	<input type="checkbox"/>
<input type="checkbox"/> Customer	<input type="checkbox"/>
<input type="checkbox"/> CustomerDemographic	<input type="checkbox"/>
<input type="checkbox"/> Employee	<input type="checkbox"/>
<input type="checkbox"/> Order	<input type="checkbox"/>
<input type="checkbox"/> Order_Detail	<input type="checkbox"/>
<input type="checkbox"/> Product	<input type="checkbox"/>
<input type="checkbox"/> Region	<input type="checkbox"/>
<input type="checkbox"/> Shipper	<input type="checkbox"/>
<input type="checkbox"/> Supplier	<input type="checkbox"/>
<input type="checkbox"/> Territory	<input type="checkbox"/>

☐ Generate associated classes for metadata

نظرات خوانندگان

نویسنده: سعید

تاریخ: ۱۹:۱۵ ۱۳۹۱/۰۹/۳۰

سلام، [از به روز رسانی جدید](#) ria services استفاده می کنید؟

dbcontext در code first هست. شما از db first استفاده کردید؟

این مدل ها قبلا هم ظاهر نمی شدند. یعنی چون بر اساس reflection کار می کند برنامه یکبار باید کامپایل شود.

نویسنده: حسین مرادی نیا

تاریخ: ۱۹:۲۴ ۱۳۹۱/۰۹/۳۰

خیر

از این به روز رسانی استفاده نکردم.

این مورد برای حالت Database First ذکر شده و مشکل به این روش حل می شود.

کنجاکو شدم بدونم که این مشکل در این به روز رسانی حل شده یا نه! اگر کسی استفاده کرده به ما هم بگه.

نویسنده: محسن

تاریخ: ۱۹:۲۴ ۱۳۹۱/۰۹/۳۰

در code first هم میشه به object context دسترسی پیدا کرد:

```
public class YourContext : DbContext
{
    public YourContext()
        : base(".....ConnectionString.....")
    {
        //ObjectContext of DbContext
        var objectContext = (this as IObjectContextAdapter).ObjectContext;
        objectContext.CommandTimeout = 120;
    }
}
```

نویسنده: ناصر فرجی

تاریخ: ۱۷:۵۰ ۱۳۹۲/۰۱/۱۵

این مشکل توی یک پروژه جدید با 2012 vs کلی وقت من رو هدر داد! بالاخره هم به صورت شانس حلش کردم. کاش زودتر این مطلب رو دیده بودم (:

نویسنده: احسان

تاریخ: ۱۰:۲۱ ۱۳۹۲/۰۱/۱۹

راه حل ارائه شده باعث می شه که شما قابلیت های DbContext رو از دست بدهید و با همون اشیاء Object Context کار کنید . در این حالت دیگه فکر نکنم استفاده از EF5 ضرورتی داشته باشه.

راه حل صحیح اینه که RiaServices.EntityFramework را با استفاده از Manage Neget Packages رو نصب بفرمائید تا Ria service بتونه اشیاء DbContext رو هم ساپورت کنه.

شاد و پیروز باشید.

نویسنده: محسن خان

تاریخ: ۱۳:۲ ۱۳۹۲/۰۱/۱۹

DbContext مطابق کدی که چند سطر بالاتر نوشته شده فقط یک Wrapper است برایObjectContext. چیز جدیدی این وسط اختراع نشده. گل همان گل است.

نویسنده: احسان
تاریخ: ۱۳۹۲/۰۱/۱۹ ۱۳:۱۳

منظور از اختراع نمی‌دونم چیه؟ ولی DbContext دارای قابلیت‌های اضافی بسیاری نسبت بهObjectContext هست. مثل تابع Find و
بله حرف شما متین هست و DbContext ازObjectContext ارث می‌بره.
شاد و پیروز باشید

نویسنده: ناصر فرجی
تاریخ: ۱۳۹۲/۰۱/۲۱ ۱۳:۴۹

میشه بگید توی یک برنامه asp.net ساده که توی vs2012 نوشته شده راه حل چیه؟ چون در صورتی که Code Generation Strategy رو تغییر ندیم دیگه به مدل دسترسی نداریم؟ (روش database first)

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۱/۲۱ ۱۳:۵۹

روش database first همان روش استفاده مستقیم ازObjectContext است.

ضمناً چرا در یک برنامه ASP.NET ازRIA Services استفاده کردید؟ RIA Services بهینه شده برای استفاده در Silverlight و فناوری‌های مانند اون.

نویسنده: ناصر فرجی
تاریخ: ۱۳۹۲/۰۱/۲۱ ۱۴:۰۴

بنده از RIA Services استفاده نکردم. در vs2012 هنگام کار با asp.net web forms پیش فرض هنگامی که یک مدل رو به برنامه اضافه می‌کنید به مدل توی کلاس‌ها و فرم‌ها دسترسی ندارید. تنها راه اینه که برید Code Generation Strategy رو از none به default تغییر بدید تا برنامه به درستی کار بکنه و بتونید مدل رو توی namespace‌ها داشته باشید. میخاستم ببینم این کاری که من انجام میدم درسته یا باید راه دیگه‌ای رو انجام بدم؟