عنوان: **استفاده از SQL-CE به کمک NHibernate**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۲/۲۵ ۲۳:۴۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: SQL Server CE

خلاصه‌ای را در مورد SQL Server CE قبلا در این سایت مطالعه کرده‌اید . در ادامه خلاصه‌ای کاربردی را از تنظیمات و نکات مرتبط به کار با SQL-CE به کمک NHibernate ملاحظه خواهید نمود:

1) دریافت SQL-CE 4.0

Microsoft SQL Server Compact 4.0

همین مقدار برای استفاده از SQL-CE 4.0 به کمک NHibernate کفایت می‌کند و حتی نیازی به نصب سرویس پک یک VS 2010 هم نیست.

2) ابزار سازی جهت ایجاد یک بانک اطلاعاتی خالی SQL-CE

```
using System;
using System.IO;

namespace NHibernate.Helper.DbSpecific
{
    public class SqlCEDbHelper
    {
        const string engineTypeName = "System.Data.SqlServerCe.SqlCeEngine, System.Data.SqlServerCe";

        /// <summary>
        /// note: this method will delete existing db and then creates a new one.
        /// </summary>
        /// <param name="filename"></param>
        /// <param name="password"></param>
        public static void CreateEmptyDatabaseFile(string filename, string password = "")
        {
            if (File.Exists(filename))
                File.Delete(filename);

            var type = System.Type.GetType(engineTypeName);
            var localConnectionString = type.GetProperty("LocalConnectionString");
            var createDatabase = type.GetMethod("CreateDatabase");

            var engine = Activator.CreateInstance(type);

            string connectionStr = string.Format("Data Source='{0}';Password={1};Encrypt
Database=True", filename, password);
            if (string.IsNullOrWhiteSpace(password))
                connectionStr = string.Format("Data Source='{0}'", filename);

            localConnectionString.SetValue(
                obj: engine,
                value: connectionStr,
                index: null);
            createDatabase.Invoke(engine, new object[0]);
        }

        /// <summary>
        /// use this method to compact or encrypt existing db or decrypt it to a new db with all
records
        /// </summary>
        /// <param name="sourceConnection"></param>
        /// <param name="destConnection"></param>
        public static void CompactDatabase(string sourceConnection, string destConnection)
        {
            var type = System.Type.GetType(engineTypeName);
            var engine = Activator.CreateInstance(type);
```

```
            var localConnectionString = type.GetProperty("LocalConnectionString");
            localConnectionString.SetValue(
                obj: engine,
                value: sourceConnection,
                index: null);

            var compactDatabase = type.GetMethod("Compact");
            compactDatabase.Invoke(engine, new object[] { destConnection });
        }
    }
}
```

کلاس فوق، یک کلاس عمومی است و مرتبط به NHibernate نیست و در همه جا قابل استفاده است.
متد CreateEmptyDatabaseFile یک فایل بانک اطلاعاتی خالی با فرمت مخصوص SQL-CE را برای شما تولید خواهد کرد. به این ترتیب می‌توان بدون نیاز به ابزار خاصی، سریعا یک بانک خالی را تولید و شروع به کار کرد. در این متد اگر کلمه عبوری را وارد نکنید، بانک اطلاعاتی رمزنگاری شده نخواهد بود و اگر کلمه عبور را وارد کنید، دیتابیس اولیه به همراه کلیه اعمال انجام شده بر روی آن در طول زمان، با کمک الگوریتم AES به صورت خودکار رمزنگاری خواهند شد. کل کاری را هم که باید انجام دهید ذکر این کلمه عبور در کانکشن استرینگ است.
متد CompactDatabase، یک متد چند منظوره است. اگر بانک اطلاعاتی SQL-CE رمزنگاری نشده‌ای دارید و می‌خواهید کل آن‌را به همراه تمام اطلاعات درون آن رمزنگاری کنید، می‌توانید جهت سهولت کار از این متد استفاده نمائید. آرگومان اول آن به کانکشن استرینگ بانکی موجود و آرگومان دوم به کانکشن استرینگ بانک جدیدی که تولید خواهد شد، اشاره می‌کند.
همچنین اگر یک بانک اطلاعاتی SQL-CE رمزنگاری شده دارید و می‌خواهید آن‌را به صورت یک بانک اطلاعاتی جدید به همراه تمام رکوردهای آن رمزگشایی کنید، باز هم می‌توان از این متد استفاده کرد. البته بدیهی است که کلمه عبور را باید داشته باشید و این کلمه عبور جایی درون فایل بانک اطلاعاتی ذخیره نمی‌شود. در این حالت در کانکشن استرینگ اول باید کلمه عبور ذکر شود و کانکشن استرینگ دوم نیازی به کلمه عبور نخواهد داشت.

فرمت کلی کانکشن استرینگ SQL-CE هم به شکل زیر است:

```
Data Source=c:\path\db.sdf;Password=1234;Encrypt Database=True
```

البته این برای حالتی است که قصد داشته باشید بانک اطلاعاتی مورد استفاده را رمزنگاری کنید یا از یک بانک اطلاعاتی رمزنگاری شده استفاده نمائید. اگر بانک اطلاعاتی شما کلمه عبوری ندارد، ذکر Data Source=c:\path\db.sdf کفایت می‌کند.

این کلاس هم از این جهت مطرح شد که NHibernate می‌تواند ساختار بانک اطلاعاتی را بر اساس تعاریف نگاشت‌ها به صورت خودکار تولید و اعمال کند، «اما» بر روی یک بانک اطلاعاتی خالی SQL-CE از قبل تهیه شده (در غیراینصورت خطای The database file cannot be found. Check the path to the database را دریافت خواهید کرد).

نکته:
اگر دقت کرده باشید در این کلاس engineTypeName به صورت رشته ذکر شده است. چرا؟
علت این است که با ذکر engineTypeName به صورت رشته، می‌توان از این کلاس در یک کتابخانه عمومی هم استفاده کرد، بدون اینکه مصرف کننده نیازی داشته باشد تا ارجاع مستقیمی را به اسمبلی SQL-CE به برنامه خود اضافه کند. اگر این ارجاع وجود داشت، متدهای یاد شده کار می‌کنند، در غیراینصورت در گوشه‌ای ساکت و بدون دردسر و بدون نیاز به اسمبلی خاصی برای روز مبادا قرار خواهند گرفت.

3) ابزار مرور اطلاعات بانک اطلاعاتی SQL-CE

با استفاده از management studio خود SQL Server هم می‌شود با بانک‌های اطلاعاتی SQL-CE کار کرد، اما ... اینبار برخلاف نگارش کامل اس کیوال سرور، با یک نسخه‌ی بسیار بدوی، که حتی امکان rename فیلدها را هم ندارد مواجه خواهید شد. به همین جهت به شخصه برنامه SqlCe40Toolbox را ترجیح می‌دهم و اطمینان داشته باشید که امکانات آن برای کار با SQL-CE از امکانات ارائه شده توسط management studio مایکروسافت، بیشتر و پیشرفته‌تر است!

4) تنظیمات NHibernate جهت کار با SQL-CE

الف) پس از نصب SQL-CE ، فایل‌های آن‌را در مسیر C:\Program Files\Microsoft SQL Server Compact Edition\v4.0 می‌توان یافت. درایور ADO.NET آن هم در مسیر C:\Program Files\Microsoft SQL Server Compact Edition\v4.0\Desktop قرار دارد. بنابراین در ابتدا نیاز است تا ارجاعی را به اسمبلی System.Data.SqlServerCe.dll به برنامه خود اضافه کنید (نام پوشه desktop آن هم غلط انداز است. از این جهت که نگارش 4 آن، به راحتی در برنامه‌های ذاتا چند ریسمانی ASP.Net بدون مشکل قابل استفاده است).
نکته مهم: در این حالت NHibernate قادر به یافتن فایل درایور یاد شده نخواهد بود و پیغام خطای «Could not create the driver from NHibernate.Driver.SqlServerCeDriver» را دریافت خواهید کرد. برای رفع آن، اسمبلی System.Data.SqlServerCe.dll را در لیست ارجاعات برنامه یافته و در برگه خواص آن، خاصیت «Copy Local» را true کنید. به این معنا که NHibernate این اسمبلی را در کنار فایل اجرایی برنامه شما جستجو خواهد کرد.

ب) مطلب بعد، تنظیمات ابتدایی NHibernate است جهت شناساندن SQL-CE . مابقی مسایل (نکات mapping، کوئری‌ها و غیره) هیچ تفاوتی با سایر بانک‌های اطلاعاتی نخواهد داشت و یکی است. به این معنا که اگر برنامه شما از ویژگی‌های خاص بانک‌های اطلاعاتی استفاده نکند (مثلا اگر از رویه‌های ذخیره شده اس کیوال سرور استفاده نکرده باشد)، فقط با تغییر کانکشن استرینگ و معرفی dialect و driver جدید، به سادگی می‌تواند به یک بانک اطلاعاتی دیگر سوئیچ کند؛ بدون اینکه حتی بخواهید یک سطر از کدهای اصلی برنامه خود را تغییر دهید.

دریافت یک مثال کامل NHibernate 3.2 در این زمینه

تنها نکته جدید آن این متد است:

```
private Configuration getConfig()
{
            var configure = new Configuration();
            configure.SessionFactoryName("BuildIt");

            configure.DataBaseIntegration(db =>
            {
                db.ConnectionProvider<DriverConnectionProvider>();
                db.Dialect<MsSql2008Dialect>();
                db.Driver<SqlServerCeDriver>();
                db.KeywordsAutoImport = Hbm2DDLKeyWords.AutoQuote;
                db.IsolationLevel = IsolationLevel.ReadCommitted;
                db.ConnectionString = ConnectionString;
                db.Timeout = 10;

                //for testing ...
                db.LogFormattedSql = true;
                db.LogSqlInConsole = true;
            });

            return configure;
}
```

که در آن نحوه تعریف MsSql2008Dialect و SqlServerCeDriver مشخص شده است.

نکته حاشیه‌ای!

در این مثال primary key از نوع identity تعریف شده و بدون مشکل کار کرد. همین را اگر با EF تست کنید، این خطا را دریافت می‌کنید: «Server-generated keys and server-generated values are not supported by SQL Server Compact». بله، EF نمی‌تواند با primary key از نوع identity حین کار با SQL-CE کار کند. برای رفع آن توصیه شده است که از Guid استفاده کنید!

نکته تکمیلی:

استفاده از Dialect سفارشی در NHibernate

نکته پایانی!

و در پایان باید اشاره کرد که SQL-CE یک بانک اطلاعاتی نوشته شده با دات نت نیست (با CPP نوشته شده است و نصب آن هم نیاز به ران تایم به روز VC را دارد). به این معنا که جهت سیستم‌های 64 بیتی و 32 بیتی باید نسخه مناسب آن‌را توزیع کنید. یا اینکه Target platform پروژه جاری دات نت خود را بر روی X86 قرار دهید (نه بر روی Any CPU پیش فرض) و در این حالت تنها یک نسخه X86 بانک اطلاعاتی SQL-CE و همچنین برنامه خود را برای تمام سیستم‌ها توزیع کنید.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۲/۲۷ ۱۷:۴۶:۴۱

جهت تکمیل این مطلب، MsSqlCe40Dialect پیش فرض تعریف یک سری از توابع SQL-CE را ندارد. این کلاس رو تکمیل کردم که از اینجا می‌تونید دریافت کنید: (^)
استفاده از آن هم بسیار ساده است. در متد getConfig فوق، بجای MsSqlCe40Dialect بنویسید CustomMsSqlCe40Dialect

نویسنده: MehdiPayervand
تاریخ: ۱۳۹۰/۱۲/۲۷ ۱۸:۰۸:۲۷

بابت اشتراک مطالبتون ممنون، مهندس نمیدونم امسال وبلاگ چه هدیه ای برای کاربرا داره ;)
سال جدید رو هم بهتون تبریک میگم و آرزوی سالی پر از موفقیت دارم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۲/۲۷ ۲۰:۲۹:۲۱

سلامت باشید؛ سال نوی شما هم مبارک.