عنوان: **آشنایی با TransactionScope**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۳/۰۹ ۰:۱۵
آدرس: www.dotnettips.info
گروه‌ها: msdtc, transactionScope

TransactionScope روشی برای پیاده سازی تراکنش در .Net است که برای اولین بار در دات نت 2 معرفی شده است. روش پیاده سازی آن بسیار ساده است و همین سادگی و راحتی کار با اون باعث شده است که خیلی از برنامه نویس‌ها رو متمایل به خودش کنه. در ادامه به روش استفاده و مزایا و معایب این روش برای پیاده سازی تراکنش‌ها می‌پردازیم.
این روش دارای تمام خواص یک تراکنش است(اصطلاحا به این خواص ACID Properties گفته میشود)
1- **Atomic** : به این معناست که تمام دستورات بین بلاک (دستورات SQL و سایر عملیات) باید به صورت عملیات اتمی کار کنند. یعنی یا تمام عملیات موفقیت آمیز است یا همه با شکست روبرو می‌شوند.
2 - **Consistent** : به این معناست که اگر تراکنش موفقیت آمیز بود پایگاه داده باید در شروع تراکنش بعدی تغییرات لازم رو انجام داده باشد و در غیر این صورت پایگاه داده باید به حالت قبل از شروع تراکنش برگردد.
3- **Isolated** : اگر چند تا تراکنش هم زمان شروع شوند اجرای هیچ کدوم از اون‌ها نباید بر اجرای بقیه تاثیر بزاره.
4- **Durable** : یعنی تغییرات حاصل شده بعد از اتمام تراکنش باید دائمی باشند.

روش کار به این صورت است تمام کارهایی که قصد داریم در طی یک تراکنش انجام شوند باید در یک بلاک قرار بگیرند و تا زمانی که متد Complete فراخوانی شود. در این بلاک شما هر عملیاتی رو که به عنوان جزئی از تراکنش می‌دونید قرار بدید. در صورتی که کنترل اجرا به فراخوانی دستور Complete برسه تمام موارد قبل از این دستور Commit می‌شوند در غیر این صورت RollBack.
به مثال زیر دقت کنید.
ابتدا به پروژه مربوطه باید اسمبلی System.Transaction رو اضافه کنید.

```
using ( TransactionScope scope = new TransactionScope() )
  {
       //Statement1
       //Statement2
       //Statement3
         scope.Complete();
    }
```

تمام دستوراتی که در این بلاک نوشته شوند بعد از فراخوانی دستور scope.Complete اصطلاحا Commit می‌شوند. اگر به هر دلیلی فراخوانی دستورات به scope.Complete نرسد عمل RollBack انجام می‌شود. در نتیجه برای این که عمل RollBack رو انجام دهید بهتره که قبل از دستور Complete یک Exception رو پرتاب کنید که باعث فراخوانی Dispose می‌شود. کد زیر

```
using ( System.Transactions.TransactionScope scope = new System.Transactions.TransactionScope() )
            {
                if ( result == 0 )
                {
                    throw new ApplicationException();
                }
                scope.Complete();
            }
```

نکته حائز اهمیت این است که اگر در هنگام اجرای برنامه به این روش به خطای

```
MSDTC on server {} is unavailable
```

برخوردید باید سرویس MSDTC رو Start کنید.برای این کار باید سرویس **Distributed Transaction Coordinator** رو از لیست سرویس‌های ویندوز پیدا کنید و بر روی اون راست کلیک کرده و دکمه Start رو بزنید.
نکته 1 : میزان Timeout در این تراکنش‌ها چه قدر است؟
برای بدست آوردن مقدار Timeout در این گونه تراکنش‌ها می‌توانید از کلاس TransactionManager استفاده کنید. به صورت زیر :

```
var defaultTimeout = TransactionManager.DefaultTimeout
var maxTimeout = TransactionManager.MaximumTimeout
```

مقدار پیش فرض برای DefaultTimeout یک دقیقه است و برای MaximumTimeout ده دقیقه است. البته خاصیت‌های بالا به صورت فقط خواندنی هستند و نمی‌تونید از این راه مقدار Timeout هر تراکنش را افزایش یا کاهش دهیم. برای این کار بهتره از روش زیر استفاده کنیم.

```
TransactionOptions option = new TransactionOptions();
 option.Timeout = TimeSpan.MaxValue;



  using ( System.Transactions.TransactionScope scope = new
System.Transactions.TransactionScope(TransactionScopeOption.Required ,option) )
  {
      scope.Complete();
  }
```

توضیح درباره انواع TransactionScopeOption

1 - Required : یعنی نیاز به تراکنش وجود دارد. در صورتی که تراکنش در یک تراکنش دیگر شروع شود نیاز به ساختن تراکنش جدید نیست و از همان تراکنش قبلی برای این کار استفاده می‌شود.

2 - RequiresNew: در هر صورت برای محدوده یک تراکنش تولید می‌شود.

3- Suppress : به عنوان محدوه تراکنش در نظر گرفته نمی‌شود.

```
using(TransactionScope scope1 = new TransactionScope())
{
     try
     {
          using(TransactionScope scope2 = new  TransactionScope(TransactionScopeOption.Suppress))
          {
               //این محدوده خارج از تراکنش محسوب می‌شود Suppress به دلیل استفاده از
          }
          //شروع محدوده تراکنش
   }
     catch
     {}
   //Rest of scope1
}
```

مزایا استفاده از این روش

1-این روش از تراکنش‌های توزیع شده پشتیبانی می‌کند . یعنی می‌تونید از چند تا منبع داده استفاده کنید یا می‌تونید از یک تراکنش چند تا Connection به یک منبع داده باز کنید.(استفاده از چند تاconnection در طی یک تراکنش)

```
using (TransactionScope scope = new TransactionScope(TransactionScopeOption.Required))
{
    string strCmd = "SQL to Execute";
    conn = new SqlClient.SqlConnection("Connection to DB1");
    conn.Open()
    objCmd = new SqlClient.SqlCommand(strCmd, conn);
    objCmd.ExecuteNonQuery();
    string strCmd2 = "SQL to Execute";
    conn2 = new SqlClient.SqlConnection("Connection to DB2");
    conn2.Open()
    objCmd2 = new SqlClient.SqlCommand(strCmd2, conn2);
    objCmd2.ExecuteNonQuery();
}
```

2- پیاده سازی این روش واقعا راحت است .

3- با DataProviderهای متفاوت نظیر Oracle و OleDb و ODBC سازگار است.

4- از تراکنش‌های تو در تو به خوبی پشتیبانی میکنه(Nested Transaction)

```
using(TransactionScope scope1 = new TransactionScope())
{
     using(TransactionScope scope2 = new  TransactionScope(TransactionScopeOption.Required))
     {
     ...
     }

     using(TransactionScope scope3 = new TransactionScope(TransactionScopeOption.RequiresNew))
     {
     ...
     }

     using(TransactionScope scope4 = new   TransactionScope(TransactionScopeOption.Suppress))
    {
     ...
    }
}
```

-5 به خوبی توسط سرویس‌های WCF پشتیبانی می‌شود و برای سیستم‌های SOA مبتنی بر WCF مناسب است. **معایب :**
*استفاده از این روش در سیستم هایی که تعداد کاربران آنلاین آن زیاد است و هم چنین تعداد تراکنش‌های موجود نیز در سطح سیستم **خیلی زیاد** باشه مناسب نیست.
*تراکنش‌های استفاده شده از این روش کند هستند.(مخصوصا که تراکنش در سطح دیتابیس با تعداد و حجم داده زیاد باشه)
امکان تغییر IsolationLevel در طی انجام یک تراکنش امکان پذیر نیست.
(به شخصه مواد * رو در سطح یک پروژه با شرایط کاربران و حجم داده زیاد تست کردم و نتیجه مطلوب حاصل نشد)

موفق باشید.

نظرات خوانندگان

نویسنده: محمد صافدل
تاریخ: ۱۳۹۲/۰۳/۰۹ ۱۰:۱۱

من از TransactionScope در چند پروژه استفاده کردم. مزایا و معایبی که شما فرمودید را منم بهش رسیدم اما دو مشکل اساسی داشت: اول اینکه با کدهایی تو در تو که بیش از چند لایه میشدن (فکر می‌کنم بیش از 3 فراخوانی تو در تو) مشکل داشت و تولید Exception میکرد متاسفانه الان حضور ذهن ندارم که Exception ایجاد شده چی بود. مشکل دوم این بود که در صورت استفاده در پروژه‌های وب نیاز به دسترسی هایی داشت که هاست‌ها همچین مجوزی را در اختیار برنامه‌ها قرار نمیدن.

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۲/۰۳/۰۹ ۱۱:۵۸

برای قسمت اول مطالب شما باید عنوان کنم که برای Nested Transaction محدودیت خاصی تعیین نشده است. تا زمانی که Timeout رخ ندهد و همه تراکنش‌ها به درستی Commit شوند مشکلی پیش نخواهد آمد. ولی باید به این نکته دقت داشت که اگر هنگام استفاده از تراکنش‌ها تو درتو یکی از تراکنش‌ها به هر دلیلی موفقیت آمیز نباشد یا زمان آن سپری شود یا زمان انجام کل تراکنش به اتمام برسد Exceptionتولید خواهد شد و تمام عملیات RollBack می‌شوند.البته می‌توان برای مدیریت این تراکنش‌ها از TransactionScopeOption که در مطلب عنوان کردم استفاده کنید. اما در توضیح قسمت دوم مطالب شما باید عنوان کنم که برای اجرای تراکنش به این روش نیاز به مجوز اجرای تراکنش از راه دور است. این کار هم از طریق تنظیمات Remote Access MSDTC میسر می‌شود. برای انجام این کار از قسمت زیر استفاده کنید.

```
ComponentServieces/Computers/MyComputer/Distributed Transaction Coodrinator/Local DTC
```

بر روی Local DTC کلیک راست کنید و گزینه‌ی Properties رو بزنید. از برگه Security تیک قسمت‌های زیر رو بزنید.

1-Notwork DTC Access

2- Allow Remote Client

3- Allow Remote Administration

4-Allow Inbound

5-Allow OutBound

6- No Authentication Required.

خود من به شخصه از این روش برای اجرای سرویس‌های WCF به صورت تراکنش استفاده می‌کنم.

نویسنده: امیر بختیاری
تاریخ: ۱۳۹۲/۰۳/۰۹ ۱۶:۴۴

با سلام و عرض خسته نباشید
مطلب خوبی بود ولی هیچ راهی نداره تو سیستمی که دیتا زیاد داره و کاربر آنلاین هم زیاد با یک مدلی تراکنش را پیاده سازی کرد
من خودم هم تو سیستمی که کار می‌کنم به این مشکل برخورد کردم و کلاً بیخیال تراکنش شدم ولی 1 درصد از عملیات سیستمم معلوم نیست چی میشن
یعنی 1 در صد از عملیات هام نصفه و نیمه باقی موندن . اگه روشی برای پیاده سازی این مدل دارید خواهشن عنوان کنید- به صورت عملی هم توضیح بدید خیلی بهتره
با تشکر

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۲/۰۳/۱۰ ۲۱:۱۵

برای اجرای تراکنش در سیستم‌های با کاربر و حجم داده زیاد بهتر ه از امکانات تراکنش موجود در ORMها استفاده کنید. برای مثال در Entity Framework می‌تونید از DBTransactionها استفاده کنید یا در NHibernate از تراکنش موجود در Session استفاده کنید. برای مثال در CodeFirst

```
public void Save( TEntity entity )
        {
            DbTransaction transaction = null;
            try
            {
                transaction = this.Database.Connection.BeginTransaction();
                //عملیات مورد نظر
                transaction.Commit();
            }
            catch
            {
                transaction.Rollback();
            }
            finally
            {
                transaction.Dispose();
            }
        }
```

البته در زمان مناسب در صورت نیاز یک پست رو به این مورد اختصاص خواهم داد.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۳/۱۰ ۲۲:۳۰

```
SaveChanges operates within a transaction. SaveChanges will roll back that transaction and
throw an exception if any of the dirty ObjectStateEntry objects cannot be persisted.
```

خود متد SaveChanges از تراکنش استفاده می‌کنه و کل تغییرات DbContext رو در طی یک تراکنش اعمال می‌کنه.

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۲/۰۳/۱۱ ۱:۶

درسته. فراخوانی دستور SaveChanged به صورت تراکنش internal عمل میکنه اما اگر در طی عملیات ذخیره سازی مجبور باشیم که بیش از یک بار SaveChanged رو فراخوانی کنیم اون موقع مجبور به استفاده از DbTransactionها هستیم.برای مثال

```
context.Connection.Open();
        DEPT department = context.DEPT.Where(d => d.DEPTNO == 10).First();
        department.LOC = "TEST";
        using (System.Data.Common.DbTransaction transaction = context.Connection.BeginTransaction())
        {
           context.SaveChanges();
           department.DNAME = "TEST";
           context.SaveChanges();
           if(flag)
           {
              transaction.Commit();
           }
           else
           {
              transaction.Rollback();
           }
        }
```

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۲/۰۳/۱۱ ۱:۲۳

یا به صورت

```
DEPT department = context.DEPT.Where(d => d.DEPTNO == 25).First();
        department.DNAME = "xxx";
        using (TransactionScope tscope = new TransactionScope(TransactionScopeOption.RequiresNew))
        {
           context.SaveChanges();
           department.DNAME = "TEST";
           context.SaveChanges();
           if(flag)
                 tscope.Complete();
        }
```

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۳/۱۱ ۱۰:۹

من با استفاده از SQL Server Profiler این قضیه رو بررسی کردم. به ازای هربار فراخوانی SaveChanges، یک تراکنش جدید درست میشه از نوع set transaction isolation level read committed . در EF6 قرار هست به READ_COMMITTED_SNAPSHOT تغییر کنه ولی اصلش فرقی نکرده.

نویسنده: علی اکبری
تاریخ: ۱۳۹۲/۰۶/۱۸ ۱۶:۲

سلام
دستورات Alter روی دیتابیس در EF6 اشکال گرفته و با نوشتن کدهای زیر پیغام خطا می‌دهد :

```
context.Database.UseTransaction(null);
context.Database.ExecuteSqlCommand("ALTER DATABASE FDb_20120  COLLATE Persian_100_CS_AI");
Error Message :   ALTER DATABASE statement not allowed within multi-statement transaction.
```

جهت رفع این اشکال دوستان چه توصیه ای دارند؟ البته سرعت عملیات و تراکنش هم برام مهمه چون تعداد Alterهایی که قرار است در اولین اجرای نرم افزار روی سیستم جهت تنظیمات مورد نظر اجرا شود زیاد است.

نویسنده: علی اکبری
تاریخ: ۱۳۹۲/۰۶/۱۸ ۱۶:۵

از چه روشی میشه دستور Create Database رو در Context مربوط به EF6 بدون خطا اجرا کرد و نتیجه مثبت ازش گرفت ؟
هرچند در EF5 هیچ مشکلی با این دستور و Alter هایی که روی دیتابیس زده میشد نداشتیم ولی اینجا پیغام multi transaction رو میده .....

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۶/۱۸ ۱۶:۵۹

```
The ALTER DATABASE statement must run in autocommit mode
(the default transaction management mode)
and is not allowed in an explicit or implicit transaction.
```

یعنی نمیشه ALTER **DATABASE** رو داخل یک تراکنش دیگر اجرا کرد. خود اس کیوال سرور اجازه نمی‌ده.

نویسنده: فریدون
تاریخ: ۱۳۹۲/۰۷/۰۴ ۲۳:۳۰

با سلام
من روی پروژه‌ای کار می‌کنم که در اون سرور داده‌ها از سرور وب جداست. روی هردو سرور فایروال خاموش بوده و تنظیمات مربوط به Network DTC نیز روی هردو سرور انجام شده است(منظور تنظیمات زیر است) 1- Notwork DTC Access
2- Allow Remote Client

Allow Remote Administration 3-

4-Allow Inbound

5-Allow OutBound

.No Authentication Required- 6

اما با وجود همه این تنظیمات در هنگام کارکردن با دیتابیس با خطای زیر مواجه میشم

System.Transactions.TransactionManagerCommunicationException: Communication with the underlying transaction manager has failed

توی اینترنت خیلی سرچ کردم ولی همه معتقد هستند که مشکل فایروال هست در حالی که فایروال سرورها کلا خاموشه. نکته اینکه روی هردو سرور ویندوز سرور 2003 نصب هست. آیا کسی هست که بتونه در این مورد منو راهنمایی کنه. ممنون

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۷/۰۵ ۹:۳۷

سرورهای متصل شده‌ی SQL Server و مبحث تراکنش‌ها

نویسنده: محمد سلامی
تاریخ: ۱۳۹۳/۰۶/۲۶ ۹:۵۸

سلام؛ اگر از الگوی Unit of work استفاده نماییم آنگاه برای پیاده سازی .Database.BeginTransaction به چه شکل باید عمل کنیم ؟

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۶/۲۶ ۱۱:۲۸

می توان یک خاصیت به صورت زیر به اینترفیس IUnitOfWork اضافه کرد. این خاصیت از نوع کلاس Database است که در فضای نام System.Data.Entity قرار دارد.

```
public interface IUnitOfWork
    {
        void Dispose();


        DbSet<T> Set<T>() where T : class;
        int SaveChanges();
        Database Database { get; }
    }
```

کلاس DbContext خود دارای خاصیتی به نام Database هست. در نتیجه نیاز به پیاه سازی مجدد ندارد.

نویسنده: جمشیدی فر
تاریخ: ۱۳۹۳/۰۶/۲۶ ۱۸:۷

اگر کد بالا بصورت زیر باشد، unitOfWork باهاش چطور برخورد میکنه؟ به عنوان مثال اگر SaveChange اولی با شکست مواجه شد، آیا تغییرات آن از context پاک میشه یا باعث خواهند شد که SaveChange دوم هم با شکست مواجه شود؟

```
context.Connection.Open();
DEPT department = context.DEPT.Where(d => d.DEPTNO == 10).First();
department.LOC = "TEST";
using (System.Data.Common.DbTransaction transaction =
context.SaveChanges();
department.DNAME = "TEST";
context.SaveChanges();
}
```

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۶/۲۷ ۱۱:۵۷

زمانی که از تراکنش‌های مربوط به DbConnection استفاده می‌کنید اتمام آن توسط متد Commit موجود در این کلاس مشخص خواهد شد نه متد SaveChanges . در نتیجه اگر تمامی SaveChangesها فراخوانی شده به درستی انجام شوند کل عملیات Commit در غیر این صورت عملیات RollBack می‌شود. در مورد TransactionScopeها نیز به همین صورت است. اگر قبل از رسیدن اجرای برنامه به دستور transcation.Complete هر گونه خطایی صادر شود عملیات RollBack خواهد شد.