**[在LINQ to SQL中使用分布式事务处理](http://www.cnblogs.com/sunnycoder/archive/2009/12/24/1631561.html)**

一、**现决条件**

当决定使用SQL Server分布式事务处理模式时，必须启动名称为Distributed Transaction Coordinator的Windows服务项，你可以在服务管理器中启动它（开始菜单-运行- services.msc），也可以在命令提示符中直接启动它（net start msdtc）。

二、**使用TransactionScope对象，使代码块成为事务性代码**

**具体代码示例如下：**

//创建两个DataContext对象

TestDataContext db1 = new Test1DataContext(); //数据库1，有一个USER表

TestDataContext db2 = new Test2DataContext(); //数据库2，有一个USER表

//创建两个Users对象

Users u1 = new Users()

{

UserName = "张三",

Password = "123"

};

Users u2 = new Users()

{

UserName = "张三",

Password = "456"

};

//将两个Users对象分别加入到不同的DataContext对象中

db1.Users.InsertOnSubmit(u1);

db2.Users.InsertOnSubmit(u2);

// 使用TransactionScope对象，使代码块成为事务性代码。

using (TransactionScope rs = new TransactionScope())

{

try

{

// 尝试提交事务

db1.SubmitChanges();

db2.SubmitChanges();

rs.Complete(); //所有的记录已经提交至数据库

}

catch (Exception ex)

{

Console.WriteLine("引发的异常类型为： {0}", ex.GetType());

Console.WriteLine("异常信息： {0}", ex.Message);

}

备注：

1、包含在TransactionScope对象代码块中的两个DataContext对象，其执行SubmitChanges方法时将处于事务之中，如果此时出现了异常，那么之前的所有操作将会回滚。也就是说如果在事务范围中（即从初始化 TransactionScope 对象到调用其 Dispose 方法之间）发生异常了异常，事务都将结束，并回滚至初始状态。这意味着，即使执行了Complete方法，在这之后发生的异常也将导致事务失败。所以，尽量不要在执行Complete方法与Dispose方法之间，加入可能会导致事务失败的代码。

2、如果TransactionScope对象成功执行了Complete方法，那么所有数据将提交至数据库中。需要注意的是，如果在TransactionScope对象代码块中未执行TransactionScope对象的Complete方法，那么事务也将终止并回滚至初始状态。

3、同样的，TransactionScope对象代码块也适用于其他代码级数据库访问，例如将多个SqlConnection对象的提交过程放在同一个TransactionScope对象代码块中，就可以方便地控制事务的执行方式。