[Spring transaction事务之roll back回滚](http://blog.csdn.net/lovejavaydj/article/details/7635848)



**试验方法：**

         写一个单元测试，调用一个service层方法（发生对数据库进行写操作的方法--insert、update、delete）即可.

**试验过程：**

         定义一个service方法如下：

**public** SMSTiming **createSMSTiming**(SMSTiming smsTiming){

                   SMSTiming s= **this**.**getSmsTimingDAO**().**createSMSTiming**(smsTiming);

**return** s;

         }

         定义二个异常(先默认配置TestException为Spring事务回滚异常)：

**publicclass** MyTestException **extends** Exception

**publicclass** TestException **extends** Exception

**注意看下：每次这个方法的不同处（抛出的异常不同）。**

测试1：

**public** SMSTiming **createSMSTiming**(SMSTiming smsTiming){

       SMSTiming s= **this**.**getSmsTimingDAO**().**createSMSTiming**(smsTiming);

**int** i = 4/0; //人为产生异常（实际这里抛出了ArithmeticException运行异常）

**return** s;

    }

测试1结果：会事务回滚----数据库中未插入新数据。

测试2：

**public** SMSTiming **createSMSTiming**(SMSTiming smsTiming) **throws** Exception{//受检异常（非运行异常）必须抛出

       SMSTiming s= **this**.**getSmsTimingDAO**().**createSMSTiming**(smsTiming);

**try**{

**int** i = 4/0; //人为产生异常

       }**catch**(Exception e){

**thrownew** Exception ("");//抛出Exception异常

       }

**return** s;

    }

测试2结果：不会事务回滚----数据库中插入新数据。

测试3：

**public** SMSTiming **createSMSTiming**(SMSTiming smsTiming) **throws** RuntimeException{//运行异常（非受检异常）可以不抛出

       SMSTiming s= **this**.**getSmsTimingDAO**().**createSMSTiming**(smsTiming);

**try**{

**int** i = 4/0; //人为产生异常

       }**catch**(Exception e){

**thrownewRuntimeException**("");//抛出RuntimeException异常

       }

**return** s;

    }

测试3结果：会事务回滚----数据库中未插入新数据

测试4：

**public** SMSTiming **createSMSTiming**(SMSTiming smsTiming) **throws** TestException{//受检异常（非运行异常）必须抛出

       SMSTiming s= **this**.**getSmsTimingDAO**().**createSMSTiming**(smsTiming);

**try**{

**int** i = 4/0; //人为产生异常

       }**catch**(Exception e){

**thrownewTestException**("");//抛出TestException异常

       }

**return** s;

    }

测试4结果：会事务回滚----数据库中未插入新数据。

测试5：

**public** SMSTiming **createSMSTiming**(SMSTiming smsTiming) **throws** MyTestException{//受检异常（非运行异常）必须抛出

       SMSTiming s= **this**.**getSmsTimingDAO**().**createSMSTiming**(smsTiming);

**try**{

**int** i = 4/0; //人为产生异常

       }**catch**(Exception e){

**thrownewMyTestException**("");//抛出MyTestException异常

       }

**return** s;

    }

 测试5结果：不会事务回滚----数据库中插入新数据。

测试6：

**public** SMSTiming **createSMSTiming**(SMSTiming smsTiming) **throws** MyTestException{//受检异常（非运行异常）必须抛出 (注意：此时spring指定配置此异常回滚)

       SMSTiming s= **this**.**getSmsTimingDAO**().**createSMSTiming**(smsTiming);

**try**{

**int** i = 4/0; //人为产生异常

       }**catch**(Exception e){

**thrownewMyTestException**("");//抛出MyTestException异常

       }

**return** s;

    }

 测试6结果：会事务回滚----数据库中未插入新数据。

**试验总结：**

测试1、测试3、测试4、测试6会进行事务回滚；测试2、测试5不会进行事务回滚。

**为什么会这样？因为是异常的类型（受检异常、运行时异常）不同或使用了Spring的rollback-for配置。**

测试1和测试3是因为抛出了运行时异常，会事务回滚。

测试4和测试5、测试6分别抛出受检异常TestException、MyTestException，那为什么测试4和测试6会事务回滚呢？

因为是我们在Spring事务配置中指定了此异常（指定rollback-for）。

**Spring框架的事务基础架构代码将默认地只在抛出运行时和unchecked exceptions时才标识事务回滚。 也就是说，当抛出一个RuntimeException 或其子类例的实例时。（Errors 也一样 - 默认地 - 标识事务回滚。）从事务方法中抛出的Checked exceptions将不被标识进行事务回滚**

现在如果我们需要手动捕获异常，并且也希望抛异常的时候能回滚肿么办呢？  
下面这样写就好了，手动回滚事务：

1. @Transactional(rollbackFor = { Exception.**class** })
2. **public** **void** test() {
3. **try** {
4. doDbStuff1();
5. doDbStuff2();
6. } **catch** (Exception e) {
7. e.printStackTrace();
8. **TransactionAspectSupport.currentTransactionStatus().setRollbackOnly();** //加上这句，如果doDbStuff2()抛了异常, doDbStuff1()是会回滚的
9. }
10. }