Spring事务管理只对出现运行期异常进行回滚

一、结论

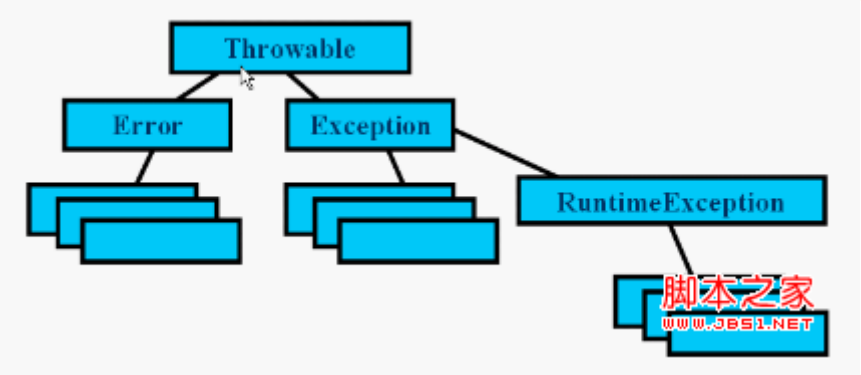
Spring的事务管理默认只对出现运行期异常(java.lang.RuntimeException及其子类)进行回滚。

如果一个方法抛出Exception或者Checked异常，Spring事务管理默认不进行回滚。

关于异常的分类一下详细介绍：

1、基本概念

看java的异常结构图



Throwable是所有异常的根，java.lang.Throwable

Error是错误，java.lang.Error

Exception是异常，java.lang.Exception

2、Exception

一般分为Checked异常和Runtime异常，所有RuntimeException类及其子类的实例被称为Runtime异常，不属于该范畴的异常则被称为CheckedException。

①Checked异常

只有java语言提供了Checked异常，Java认为Checked异常都是可以被处理的异常，所以Java程序必须显示处理Checked异常。如果程序没有处理Checked异常，该程序在编译时就会发生错误无法编译。这体现了Java的设计哲学：没有完善错误处理的代码根本没有机会被执行。对Checked异常处理方法有两种

(1) 当前方法知道如何处理该异常，则用try...catch块来处理该异常。

(2) 当前方法不知道如何处理，则在定义该方法是声明抛出该异常。

复制代码 代码如下:

```
package cn.xy.test;
import java.io.IOException;
/**
* Checked异常测试方法
* @author xy
*
*/
public class CheckedExceptionMethods
{
// 总异常类，既有checkedException又有RuntimeException，所以其中的checkedException必须处理
```

```
public void method1() throws Exception
{
System.out.println("我是抛出异常总类的方法");
}
// 捕获并处理这个异常
public void testMethod1_01()
{
try
{
method1();
}
catch (Exception e)
{
e.printStackTrace();
}
}
// 把异常传递下去
public void testMethod1_02() throws Exception
{
method1();
}
public void testMethod1_03() throws Exception
{
throw new Exception();
}
public void testMethod1_04()
{
try
{
throw new Exception();
}
catch (Exception e)
{
e.printStackTrace();
}
}
// checkedException典型代表IOException
public void method2() throws IOException
{
System.out.println("我是抛出IO异常的方法");
}
public void testMethod2_01()
{
try
```

```
{
method2();
}
catch (Exception e)
{
e.printStackTrace();
}
}
public void testMethod2_02() throws Exception
{
method2();
}
}
```

我们比较熟悉的Checked异常有
Java.lang.ClassNotFoundException
Java.lang.NoSuchMetodException
java.io.IOException
②RuntimeException
Runtime如除数是0和数组下标越界等，其产生频繁，处理麻烦，若显示申明或者捕获将会对程序的可读性和运行效率影响很大。所以由系统自动检测并将它们交给缺省的异常处理程序。当然如果你有处理要求也可以显示捕获它们。
复制代码 代码如下:

```
package cn.xy.test;
/**
* 运行时异常测试方法
* @author xy
*
*/
public class RuntimeExcetionMethods
{
public void method3() throws RuntimeException
{
System.out.println("我是抛出运行时异常的方法");
}
public void testMethod3_01()
{
method3();
}
public void testMethod1_02()
{
throw new RuntimeException();
}
```

```
}
```

我们比较熟悉的RumtimeException类的子类有
Java.lang.ArithmeticException
Java.lang.ArrayStoreExcetpion
Java.lang.ClassCastException
Java.lang.IndexOutOfBoundsException
Java.lang.NullPointerException

3、Error

当程序发生不可控的错误时，通常做法是通知用户并中止程序的执行。与异常不同的是Error及其子类的对象不应被抛出。

Error是throwable的子类，代表编译时间和系统错误，用于指示合理的应用程序不应该试图捕获的严重问题。

Error由Java虚拟机生成并抛出，包括动态链接失败，虚拟机错误等。程序对其不做处理。

二、改变默认方式

在@Transaction注解中定义noRollbackFor和RollbackFor指定某种异常是否回滚。

@Transaction(noRollbackFor=RuntimeException.class)
@Transaction(RollbackFor=Exception.class)
这样就改变了默认的事务处理方式。

三、启示

这就要求我们在自定义异常的时候，让自定义的异常继承自RuntimeException，这样抛出的时候才会被Spring默认的事务处理准确处理。