Michael

永久域名 http://sjsky.iteye.com

sjsky

浏览: 62049 次

性别:

来自: 上海

我现在离线

详细资料 留言簿

搜索本博客

搜索

最近访客 >>更多访客


freeza


squll369


jzinfo

malinfei

博客分类

- 全部博客 (86)
- Java (15)
- 企业应用 (5)
- SNMP (4)
- 企业架构 (4)
- 监控&调优 (2)
- Javascript (3)
- IO (3)
- web (1)
- Chart (2)
- webservice (4)
- oracle (5)
- MySQL (3)
- Android (2)

2011-07-01

unauthenticated user 3顶 2踩

分布式事务JTA之实践:Spring+ATOMIKOS

博客分类: Spring

框架 Spring 企业应用 OpenSource JTA

在2011.7.1号这个特殊的日子，发一blog，名为:分布式事务JTA之实践:Spring+ATOMIKOS 来庆祝党的90岁生日和iteye博客产品成功改版上线。

本文的目录结构如下:

一、概述:

本文主要讲述如何基于Atomikos 和 spring 在项目中实现分布式事务管理

二、应用场景:

如果项目中的数据源来自多个数据库，同时又需要在多数据源中保证事务，此时就需要用到分布式事务处理了。

三、实验模拟需求 :

比如有两个对象：用户信息、用户存款，用户信息存在数据库A、存款信息存在数据库B，若客户甲向乙转账，需要在数据库B中对甲、乙的存款信息修改，同时在数据库A中把甲、乙的备注信息最新为最近一次的操作时间。

四 、实例测试环境 :

- spring、hibernate3.2
- mysql5.1.51（需要版本5.0+）
- AtomikosTransactionsEssentials-3.7.0 （详细可参加它的官网：http://www.atomikos.com ）

说明:

1. 测试的数据库需要支持分布式事务，同时JDBC要支持XA 连接驱动 。本次测试用的mysql5.1是支持事务的，JDBC驱动版本：mysql-connector-java-5.1.7-bin.jar，包含对 XA连接的支持：com.mysql.jdbc.jdbc2.optional.MysqlXAConnection。
2. 附件提供AtomikosTransactionsEssentials 3.7.0 lib包下载 ： AtomikosTransactionsEssentials-3.7.0-lib.zip
   官方下载地址：http://www.atomikos.com/Main/TransactionsEssentialsDownloadForm ，需要先注册才能下载。同时这里也提供目前3.7.0的下载链接：http://www.atomikos.com/downloads/transactions-essentials/com/atomikos/AtomikosTransactionsEssentials/3.7.0/AtomikosTransactionsEssentials-3.7.0-bin.zip

五、代码及配置介绍:

源代码下载 ： 分布式事务实例演示源代码

- opensource (0)
- Linux (6)
- NoSQL (1)
- Struts (1)
- Spring (5)
- Hibernate (2)
- 爬虫 (2)
- 技术小贴士 (8)
- Exception (8)
- 杂谈 (0)

我的留言簿 >>更多留言

- 楼主，您好，请问有没有在web上实现实时跟踪曲线的例子。万分感激。小弟在此先谢过。 ...
  -- by 243644128
- 请教问题
  -- by 吴小懒
- 谢了
  -- by wsxahn

其他分类

- 我的收藏 (0)
- 我的代码 (0)
- 我的论坛主题帖 (8)
- 我的所有论坛帖 (30)
- 我的精华良好帖 (0)

最近加入群组

- JAVA 3T
- JBPM @net

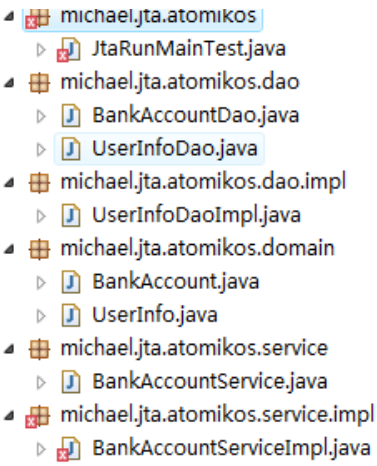
存档

- 2011-07 (1)
- 2011-06 (12)
- 2011-05 (11)
- 更多存档...

评论排行榜

- JVM内存监控:visualVM jconsole jstatd jm ...
- 分布式事务JTA之实践:Spring+ATOMIKOS
- WebService Exception: Unable to create J ...

1.代码的目录结构图如下:



转账测试的的代码片段:

Java代码

```java
/**
     * 转账测试
     * @param srcId
     * @param destId
     * @param money
     * @return boolean
     */
    public boolean doTestTransfer(String srcId, String destId, float money) {

        BankAccount srcAccount = bankAccountDao.getByUserName(srcId);
        BankAccount destAccount = bankAccountDao.getByUserName(destId);
        if (srcAccount.getDeposit() < money) {
            System.out.println("warn :" + srcAccount.getUserName()
                    + " has not enough money to transfer");
            return false;
        }
        srcAccount.setDeposit(srcAccount.getDeposit() - money);
        destAccount.setDeposit(destAccount.getDeposit() + money);
        // 把更新存款信息置于异常发生之前
        bankAccountDao.update(srcAccount);
        bankAccountDao.update(destAccount);

        Date curTime = new Date();
        UserInfo srcUser = userInfoDao.getById(srcId);
        UserInfo destUser = userInfoDao.getById(destId);

        destUser.setRemark1(curTime + "");
        destUser.setRemark2(curTime + "");
        // 把更新基本信息置于异常发生之前
        userInfoDao.update(destUser);
        srcUser.setRemark1(curTime + "");
        if (srcAccount.getDeposit() < 18000) {
            throw new RuntimeException("michael test exception for JTA  ");
        }
        srcUser.setRemark2(curTime + "");

        userInfoDao.update(srcUser);
        System.out.println("success done:" + srcAccount.getUserName()
                + " transfer ￥" + money + " to " + destAccount.getUserName());

```

```
41.         return true;
42.     }
```

2. 配置文件详细介绍:

jta.jdbc.properties

Java代码

```
1.  #see http://sjsky.iteye.com
2.  # eg. for mysql
3.  jdbc.SDS.class=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource
4.  jdbc.SDS.properties=URL=jdbc:mysql://192.168.8.253:3306/demota;user=root;password=111111
5.
6.  jdbc.SDS2.class=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource
7.  jdbc.SDS2.properties=URL=jdbc:mysql://192.168.8.150:3306/demota;user=root;password=111111
```

jta.properties

Java代码

```
1.  #see http://sjsky.iteye.com
2.  com.atomikos.icatch.service=com.atomikos.icatch.standalone.UserTransactionServiceFactory
3.  com.atomikos.icatch.console_file_name = tm.out
4.  com.atomikos.icatch.log_base_name = tmlog
5.  com.atomikos.icatch.tm_unique_name = com.atomikos.spring.jdbc.tm
6.  com.atomikos.icatch.console_log_level = INFO
```

jta1.hibernate.cfg.xml

Xml代码

```
1.  <!DOCTYPE hibernate-configuration
2.      PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
3.      "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
4.
5.  <hibernate-configuration>
6.          <session-factory>
7.                  <property name="dialect">
8.                          org.hibernate.dialect.MySQL5Dialect
9.                  </property>
10.                 <property name="hbm2ddl.auto">update</property>
11.                 <mapping class="michael.jta.atomikos.domain.UserInfo" />
12.         </session-factory>
13.
14. </hibernate-configuration>
```

jta2.hibernate.cfg.xml

Xml代码

```
1.  <!DOCTYPE hibernate-configuration
2.      PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
3.      "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
4.
5.  <hibernate-configuration>
6.
7.          <session-factory>
8.                  <property name="show_sql">true</property>
9.                  <property name="dialect">
10.                         org.hibernate.dialect.MySQL5Dialect
11.                 </property>
12.                 <property name="hbm2ddl.auto">update</property>
13.                 <mapping class="michael.jta.atomikos.domain.BankAccount" />
14.         </session-factory>
15.
16. </hibernate-configuration>
```

jta.spring.xml

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
	<!-- Configurer that replaces ${...} placeholders with values from properties files -->

	<bean id="propertyConfigurer"
		class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

		<property name="locations">
			<list>
				<value>classpath:jta.jdbc.properties</value>
			</list>
		</property>
	</bean>

	<!-- 数据源配置 http://sjsky.iteye.com-->
	<bean id="SDS" class="com.atomikos.jdbc.SimpleDataSourceBean"
		init-method="init" destroy-method="close">
		<property name="uniqueResourceName">
			<value>SDS</value>
		</property>
		<property name="xaDataSourceClassName">
			<value>${jdbc.SDS.class}</value>
		</property>
		<property name="xaDataSourceProperties">
			<value>${jdbc.SDS.properties}</value>
		</property>
		<property name="exclusiveConnectionMode">
			<value>true</value>
		</property>
		<property name="connectionPoolSize">
			<value>3</value>
		</property>
		<property name="validatingQuery">
			<value>SELECT 1</value>
		</property>
	</bean>

	<bean id="SDS2" class="com.atomikos.jdbc.SimpleDataSourceBean"
		init-method="init" destroy-method="close">
		<property name="uniqueResourceName">
			<value>SDS2</value>
		</property>
		<property name="xaDataSourceClassName">
			<value>${jdbc.SDS2.class}</value>
		</property>
		<property name="xaDataSourceProperties">
			<value>${jdbc.SDS2.properties}</value>
		</property>
		<property name="exclusiveConnectionMode">
			<value>true</value>
		</property>
		<property name="connectionPoolSize">
			<value>3</value>
		</property>
		<property name="validatingQuery">
			<value>SELECT 1</value>
		</property>
	</bean>
```

```
58.
59.         <!-- sessionFactory http://sjsky.iteye.com-->
60.         <bean id="sessionFactory1"
61.                 class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBea
    n">
62.                 <property name="dataSource" ref="SDS" />
63.                 <property name="configLocation"
64.                         value="classpath:jta1.hibernate.cfg.xml" />
65.         </bean>
66.
67.         <bean id="sessionFactory2"
68.                 class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBea
    n">
69.                 <property name="dataSource" ref="SDS2" />
70.                 <property name="configLocation"
71.                         value="classpath:jta2.hibernate.cfg.xml" />
72.         </bean>
73.
74.         <!-- TransactionManager http://sjsky.iteye.com-->
75.         <!-- Construct Atomikos UserTransactionManager, needed to configure Spring -->
76.         <bean id="atomikosTransactionManager"
77.                 class="com.atomikos.icatch.jta.UserTransactionManager"
78.                 init-method="init" destroy-method="close">
79.                 <!--  when close is called, should we force transactions to terminate or not?
    -->
80.                 <property name="forceShutdown">
81.                         <value>true</value>
82.                 </property>
83.         </bean>
84.
85.         <!-- Also use Atomikos UserTransactionImp, needed to configure Spring  -->
86.         <bean id="atomikosUserTransaction"
87.                 class="com.atomikos.icatch.jta.UserTransactionImp">
88.                 <property name="transactionTimeout">
89.                         <value>300</value>
90.                 </property>
91.         </bean>
92.
93.         <!-- Configure the Spring framework to use JTA transactions from Atomikos -->
94.         <bean id="springJTATransactionManager"
95.                 class="org.springframework.transaction.jta.JtaTransactionManager">
96.
97.                 <property name="transactionManager">
98.                         <ref bean="atomikosTransactionManager" />
99.                 </property>
100.                <property name="userTransaction">
101.                        <ref bean="atomikosUserTransaction" />
102.                </property>
103.        </bean>
104.
105.        <!-- Configure DAO http://sjsky.iteye.com-->
106.        <bean id="userInfoDao"
107.                class="michael.jta.atomikos.dao.impl.UserInfoDaoImpl">
108.                <property name="sessionFactory" ref="sessionFactory1" />
109.        </bean>
110.        <bean id="bankAccountDao"
111.                class="michael.jta.atomikos.dao.BankAccountDao">
112.                <property name="sessionFactory" ref="sessionFactory2" />
113.        </bean>
114.
115.        <bean id="bankAccountService"
116.                class="michael.jta.atomikos.service.impl.BankAccountServiceImpl">
117.                <property name="userInfoDao" ref="userInfoDao" />
118.                <property name="bankAccountDao" ref="bankAccountDao" />
119.        </bean>
```

```
120.
121.        <!-- 定义事务规则的拦截器 http://sjsky.iteye.com-->
122.        <bean id="transactionInterceptor"
123.                class="org.springframework.transaction.interceptor.TransactionInterceptor">
124.                <property name="transactionManager"
125.                        ref="springJTATransactionManager" />
126.                <property name="transactionAttributes">
127.                        <props>
128.                                <prop key="*">PROPAGATION_REQUIRED</prop>
129.                        </props>
130.                </property>
131.        </bean>
132.
133.        <!-- 声明式事务边界配置 所有的bean公用一个代理bean http://sjsky.iteye.com-->
134.        <bean id="baseTransactionProxy"
135.                class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean"

136.                abstract="true">
137.                <property name="transactionManager"
138.                        ref="springJTATransactionManager" />
139.                <property name="transactionAttributes">
140.                        <props>
141.                                <!-- 可以根据实际情况细化配置提高性能 -->
142.                                <prop key="*">PROPAGATION_REQUIRED</prop>
143.                        </props>
144.                </property>
145.        </bean>
146.
147.        <bean id="bankAccountServiceProxy" parent="baseTransactionProxy">
148.                <property name="target">
149.                        <ref bean="bankAccountService" />
150.                </property>
151.        </bean>
152.
153. </beans>
```

六、测试验证

1. 初始化数据:

因为mysql数据库表的类型有事务和非事务之分，建表时一定要注意确保表的类型是事务控制的：InnoDB

数据库A（192.168.8.253）：

Sql代码

```
1.  DROP DATABASE IF EXISTS demota;
2.  CREATE DATABASE demota;
3.  USE demota;
4.  DROP TABLE  IF EXISTS tb_user_info;
5.
6.  CREATE TABLE tb_user_info (
7.          user_name varchar(20),
8.          real_name varchar(10),
9.          remark1 varchar(50),
10.         remark2 varchar(50)
11.         ) ENGINE = InnoDB;
12.
13. INSERT INTO tb_user_info (user_name,real_name,remark1,remark2) VALUES ('husband','husband','','');
14. INSERT INTO tb_user_info (user_name,real_name,remark1,remark2) VALUES ('wife','wife','','');
```

数据库B（192.168.8.150）：

Sql代码

```sql
DROP DATABASE IF EXISTS demota;
CREATE DATABASE demota;
USE demota;
DROP TABLE  IF EXISTS tb_account;
CREATE TABLE tb_account (
 id int AUTO_INCREMENT,
 user_name varchar(20),
 deposit float(10,2),
 PRIMARY KEY(id)
 ) ENGINE = InnoDB;

INSERT INTO tb_account (user_name,deposit) VALUES ('husband',20000.00);
INSERT INTO tb_account (user_name,deposit) VALUES ('wife',10000.00);
```

2. 测试过程:

ps: 代码中模拟了异常出现的条件：如果账户金额<18000会抛出异常

JtaRunMainTest.java

Java代码

```java
package michael.jta.atomikos;

import michael.jta.atomikos.service.BankAccountService;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

/**
 * @author michael
 *
 */
public class JtaRunMainTest {

    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("------------start");
        ApplicationContext appCt = new ClassPathXmlApplicationContext(
                "jta.spring.xml");
        System.out.println("------------finished init xml");

        Object bean = appCt.getBean("bankAccountServiceProxy");
        System.out.println(bean.getClass());
        BankAccountService service = (BankAccountService) bean;
        service.doTestTransfer("husband", "wife", 2000);
    }
}
```

运行第一次结果:

写道

```
------------start
------------finished init xml
class $Proxy11
Hibernate: select bankaccoun0_.id as id2_, bankaccoun0_.deposit as deposit2_, bankaccoun0_.user_name
```

```
as user3_2_ from tb_account bankaccoun0_ where bankaccoun0_.user_name=?
Hibernate: select bankaccoun0_.id as id2_, bankaccoun0_.deposit as deposit2_, bankaccoun0_.user_name
as user3_2_ from tb_account bankaccoun0_ where bankaccoun0_.user_name=?
success done:husband transfer ￥2000.0 to wife
Hibernate: update tb_account set deposit=?, user_name=? where id=?
Hibernate: update tb_account set deposit=?, user_name=? where id=?
```

运行第二次结果:

写道

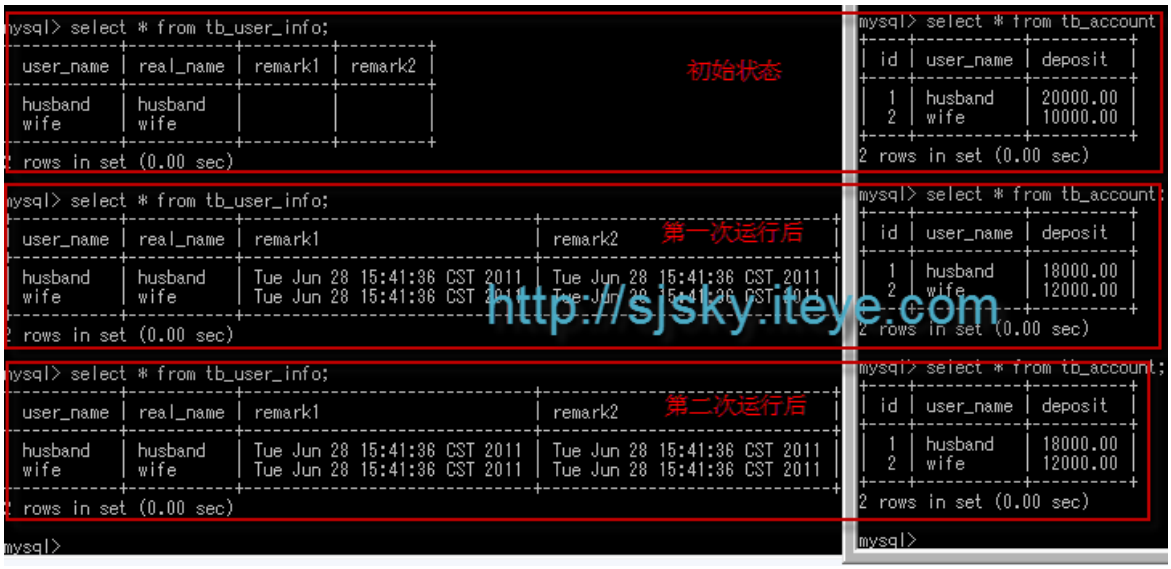
```
----------start
----------finished init xml
class $Proxy11
Hibernate: select bankaccoun0_.id as id2_, bankaccoun0_.deposit as deposit2_, bankaccoun0_.user_name
as user3_2_ from tb_account bankaccoun0_ where bankaccoun0_.user_name=?
Hibernate: select bankaccoun0_.id as id2_, bankaccoun0_.deposit as deposit2_, bankaccoun0_.user_name
as user3_2_ from tb_account bankaccoun0_ where bankaccoun0_.user_name=?
Exception in thread "main" java.lang.RuntimeException: michael test exception for JTA
at
michael.jta.atomikos.service.impl.BankAccountServiceImpl.doTestTransfer(BankAccountServiceImpl.java:51)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:597)
at org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:299)
at
org.springframework.aop.framework.ReflectiveMethodInvocation.invokeJoinpoint(ReflectiveMethodInvocation.java:

at
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:139)

at org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionInterceptor.java:107)
at
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:161)

at org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:202)
at $Proxy11.doTestTransfer(Unknown Source)
at michael.jta.atomikos.JtaRunMainTest.main(JtaRunMainTest.java:26)
```

测试过程中数据库查询的结果截图:



从上面的数据库截图可见，第一正常运行时两个数据库同步更新了，第二次运行发生异常后，两个数据库的数据为发生变化，实现了事务回滚。

转载请注明来自：Michael's blog @ http://sjsky.iteye.com

----------分----------隔----------线----------

评论

3 楼 sjsky 7 小时前 引用

suhuanzheng7784877 写道

兄弟，什么时候将你的blog做成电子书啊~~~支持。。。。

没问题下次一定做的，谢谢支持噢
改版之后好像一个文章属于多个分类，做电子书时好像会重复的

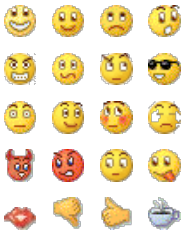
2 楼 suhuanzheng7784877 8 小时前 引用

兄弟，什么时候将你的blog做成电子书啊~~~支持。。。。

1 楼 吴小懒 8 小时前 引用

发表评论

表情图标



B I U Quote Code List Img URL Flash Table

字体颜色: 标准 字体大小: 标准 对齐: 标准

提示：选择您需要装饰的文字，按上列按钮即可添加上相应的标签

您还没有登录，请登录后发表评论(快捷键 Alt+S / Ctrl+Enter) 提交