Hibernate3.x调用存储过程大揭密

2007-05-15 来源:IT168 作者:IT168 Amigo

关键词: Hibernate 存储 AT 存储过程 调用

【IT168 专稿】本文以详尽的实例展示了hibernate3.x中调用存储过程各步骤，从建立测试表、存储过程的建立、工程的建立以及类的编写和测试一步一步引导用户学习hibernate3.x中调用存储过程的方法。

如果底层数据库（eg. Oracle、mysql、sqlserver）等支持存储过程，可通过存储过程执行批量删除、更新等操作。本文以实例说明在hibernate3.x中如何调用存储过程。

说明：本例hibernate所用版本为3.0，mysql所用版本为5.0，所用数据库驱动为mysql-connector-java-5.0.4-bin.jar。

一. 建表与初始化数据

在mysql的test数据库中建立一张新表：tbl_user,建表语句如下：

```
DROP TABLE IF EXISTS `user`;
CREATE TABLE `tbl_user` (
`userid` varchar(50) NOT NULL,
`name` varchar(50) default '',
```

热点时讯

国内微博竞争形式分析
计算机人性化界面会展
“一句话酷哥”乔布斯酷劲依旧
中国移动支付行业发展状况与商业模式
甲骨文大手笔6.9亿美元收购医药软件公司

```
`blog` varchar(50) default '',
PRIMARY KEY (`userid`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312;
```

建表成功后，在该表中任意插入几条数据。

二． 建立存储过程

为测试hibernate3.x中存储过程的调用，我们在user表中建立getUserList、createUser、updateUser和deleteUser这四个存储过程，在mysql中建立存储过程的语句如下：

1. 获得用户信息列表的存储过程--getUserList

```
DROP PROCEDURE IF EXISTS `getUserList`;
CREATE PROCEDURE `getUserList`()
begin
select * from tbl_user;
end;
```

2. 通过传入的参数创建用户的存储过程--createUser

```
DROP PROCEDURE IF EXISTS `createUser`;
CREATE PROCEDURE `createUser`(IN userid varchar(50), IN name varchar(50), IN blog va
rchar(50))
begin
insert into tbl_user values(userid, name, blog);
end;
```

3. 通过传入的参数更新用户信息的存储过程--updateUser

```
DROP PROCEDURE IF EXISTS `updateUser`;
CREATE PROCEDURE `updateUser`(IN nameValue varchar(50), IN blogValue varchar(50), I
N useidValue varchar(50))
begin
update tbl_user set name = nameValue, blog = blogValue where userid = useridValue;
end;
```

4. 删除用户信息的存储过程--deleteUser

```
DROP PROCEDURE IF EXISTS `deleteUser`;
CREATE PROCEDURE `deleteUser`(IN useridValue int(11))
begin
```

```
delete from tbl_user where userid = useridValue;
end;
```

三． 编码与测试

在准备工作完成后，进入编码与测试阶段，本例演示了在hibernate3.0中调用mysql的存储过程的方法。

1、hibernate的配置文件

在hibernate的配置文件中包含数据库的连接信息，以及加入OR mapping的xml格式的映射文件，该文件如下（部分内容略）：

```
.....
  <property name="connection.url">jdbc:mysql://localhost:3306/test</property>
  <property name="connection.username">root</property>
  <property name="connection.password">root</property>
  <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
  <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
  <property name="show_sql">true</property>
  <mapping resource="com/amigo/proc/model/User.hbm.xml"/>
 ......
```

2、OR Mapping文件

产生的OR Mapping文件有User.java以及其对应的hibernate映射文件User.hbm.xml。其中User.java的内容如下:

```
public class User implements java.io.Serializable {
private static final long serialVersionUID = 1L;
```

```
/** 用户id*/
private String userid;
/** 用户姓名*/
private String name;
/** 用户blog*/
private String blog;
//省略get/set方法
}
```

User.hbm.xml文件的内容如下：

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mappin
g DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.amigo.proc.model">
    <class name="User" table="tbl_user">
        <id name="userid" column="userid">
            <generator class="assigned"/>
        </id>
        <property name="name" column="name" type="string" />
        <property name="blog" column="blog" type="string" />
    </class>

    <sql-query name="getUserList" callable="true">
  <return alias="user" class="User">
   <return-property name="userid" column="userid"/>
   <return-property name="name" column="name"/>
   <return-property name="blog" column="blog" />
  </return>
  {call getUserList()}
```

```
    </sql-query>
</hibernate-mapping>
```

在该文件中需注意<sql-query…></sql-query>中的这段代码，调用的存储过程在其中定义，并定义了调用存储过程后将记录组装成User对象，同时对记录的字段与对象的属性进行相关映射。

3．hibernate调用存储过程的测试类

本类是该例的核心类，在本类中，以实例清楚地说明了在hibernate中如何调用存储过程，例示了hibernate调用查询、更新、插入和删除这四类存储过程的方法，该类的内容如下：

```
// hibernate调用存储过程
public class ProcTest {

 public static void main(String[] args) throws Exception {
  ProcTest proc = new ProcTest();
  Session session = HibernateSessionFactory.getSession();
  proc.testProcQuery(session);
  proc.testProcUpdate(session);
  System.out.println("update successfully");

  proc.testProcInsert(session);
  System.out.println("insert successfully");

  proc.testProcDelete(session);
  System.out.println("delete successfully");
  session.close();
 }
```

```
// 测试实现查询的存储过程
private void testProcQuery(Session session) throws Exception {
 //查询用户列表
      List list = session.getNamedQuery("getUserList").list();
      for (int i = 0; i < list.size(); i++) {
          User user = (User) list.get(i);
          System.out.print("序号: " + (i+1));
          System.out.print(", userid: " + user.getUserid());
          System.out.print(", name: " + user.getName());
          System.out.println(", blog: " + user.getBlog());
      }
}

/**
 * 测试实现更新的存储过程
 * @throws Exception
 */
private void testProcUpdate(Session session) throws Exception {
 //更新用户信息
 Transaction tx = session.beginTransaction();
      Connection con = session.connection();
      String procedure = "{call updateUser(?, ?, ?)}";
      CallableStatement cstmt = con.prepareCall(procedure);
      cstmt.setString(1, "陈xx");
      cstmt.setString(2, "http://www.blogjava.net/sterningChen");
      cstmt.setString(3, "sterning");
      cstmt.executeUpdate();
      tx.commit();
}

// 测试实现插入的存储过程
private void testProcInsert(Session session) throws Exception {
```

```
    //创建用户信息
    session.beginTransaction();
        PreparedStatement st = session.connection().prepareStatement("{call
createUser(?, ?, ?)}");
        st.setString(1, "amigo");
        st.setString(2, "阿蜜果");
        st.setString(3, "http://www.wblogjava.net/amigoxie");
        st.execute();
        session.getTransaction().commit();
    }

    // 测试实现删除的存储过程
    private void testProcDelete(Session session) throws Exception {
    //删除用户信息
    session.beginTransaction();
    PreparedStatement st = session.connection().prepareStatement("{call
deleteUser
(?)}");
        st.setString(1, "amigo");
        st.execute();
        session.getTransaction().commit();
    }
}
```

在本类中，调用查询类存储过程时，调用session.getNamedQuery("...")方法来获得User.hbm.xml中配置的查询存储过程。在其余的存储过程调用的测试中，首先通过hibernate的session获得connection，然后调用connection对象的相应方法来实现存储过程的调用。

五．总结

本例提出了在hibernate3中调用mysql的存储过程的实现方案，从本例可以看出，hibernate提供了在*.hbm.xml中配置调用存储过程，并通过向用户提供session.getNamedQuery("...")方法来调用配置的调用查询相关的存储过程的方法，另外，hibernate还提供了取得sql的connection的方法，从而能够通过connection中存储过程调用相关的方法来实现存储过程的调用。

上一篇：Spring事务探索 下一篇：GOF设计模式 --- 工厂模...

相关文章

IBM：相变存储计划取得重大突破(图)
日立环球存储科技就硬盘召回事件发表声明
世纪存储--2000年硬盘市场回顾及展望
IBM存储解决方案帮助航运公司降低运营成本
回顾IBM公司在存储器历史上的第一
VERTIAS：2003年适用性存储软件架构全国10城巡展