wumingabc的公告

Read a bit and take it out , then come back read some more

文章分类

- drollery
- English
- gossip
- java

hibernate调用存储过程 收藏

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 '',
  `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 varchar(50))
begin
   insert into tbl_user values(userid, name, blog);
end;
```

javascript
java相关工具
my question
others
safe
spring
SQL Sever
web
系统小点滴
正则表达式

收藏

at sixes and sevens

存档

2009年07月(3)
2009年06月(1)
2009年05月(3)
2009年02月(4)
2008年11月(1)
2008年10月(1)
2008年08月(3)
2008年01月(1)
2007年11月(2)
2007年10月(1)
2007年09月(2)
2007年07月(2)
2007年06月(2)
2007年05月(1)
2007年04月(1)
2007年03月(5)
2006年12月(15)

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), IN useridValue 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;
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 Mapping 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。

发表于 @ 2009年05月19日 18:54:00 | 评论(0) | 举报 | 收藏

旧一篇:Spring中事务的传播属性详解 | 新一篇:常用正则表达式收集

给wumingabc的留言 □□□□□□□□□□□□□□ □□ □□