Calling Oracle Stored Procedure From Hibernate

In this blog, I show two ways to call a stored procedure of Oracle from Hibernate. I assume the reader knows how to use hibernate and how to create stored procedure in Oracle. The example stored procedure takes two parameters and doesn't return anything. It raises an Oracle error if the given employee id is not found.

Create Stored Procedure

Create a stored procedure to change salary of the given employee, by enter the following code at SQL prompt in Sql*Plus. You can use any of the other client tools provided by Oracle such as Application Express and SQL Developer.

```
create or replace procedure changesalary(p_employeeid number, p_newsalary number)  is
begin
  update employees set salary= p_newsalary
  where employee_id = p_employeeid;

  if  sql%notfound then
      raise_application_error(-20100,'Invalid Employee Id');
  end if;

end;
/
```

Hibernate Configuration File (hibernate.cfg.xml)

The following details are to be provided in hibernate.cfg.xml. I am using Oracle Thin driver to connect to HR account. My Oracle is running on the same system as Hibernate application. According to your environment, change the setting as necessary.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-cor
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.Oracle9Dialect</property>
    <property name="hibernate.connection.driver_class">oracle.jdbc.driver.OracleDriver</property>
    <property name="hibernate.connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
    <property name="hibernate.connection.username">hr</property>
    <property name="hibernate.connection.password">hr</property>
    <property name="hibernate.hbm2ddl.auto">update</property>
    <property name="hibernate.show_sql">true</property>
  </session-factory>
</hibernate-configuration>
```

Hibernate Application

You can call stored procedure using any of the following two techniques:

Using JDBC Connection

In this technique, we obtain Connection object from Hibernate Session object and then use CallableStatement to call stored procedure. The following code shows this technique:

```
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class CallSP {

    public static void main(String[] args) throws Exception  {

        Configuration c = new Configuration().configure();
        SessionFactory sf = c.buildSessionFactory();
        Session session = sf.openSession();
        session.beginTransaction();

        Connection con = session.connection();  // obtain JDBC connection from Session object
        CallableStatement cs = con.prepareCall("{ call changesalary(?,?) }");
        cs.setInt(1,100);  // first parameter index start with 1
        cs.setInt(2,6000); // second parameter
        cs.execute();  // call stored procedure

        session.getTransaction().commit();
        session.close();
        sf.close();
    }
}
```

Using Native SQL

The second technique (and recommended one) is to call a stored procedure using native query with standard stored procedure calling syntax of JDBC. The following code demonstrates it.

```
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
```

```
import org.hibernate.cfg.Configuration;

public class CallSP {

    public static void main(String[] args) throws Exception  {

        Configuration c = new Configuration().configure();
        SessionFactory sf = c.buildSessionFactory();
        Session session = sf.openSession();
        session.beginTransaction();

        Query q = session.createSQLQuery(" { call changesalary(?,?) }");
        q.setInteger(0,100);  // first parameter, index starts with 0
        q.setInteger(1,4000); // secon parameter
        q.executeUpdate();
        session.getTransaction().commit();
        session.close();
        sf.close();
    }
}
```


Post Your Comment

Enter the code given in the above image :

Your Name :

Your Email Address :

Comment :

Post Comment

Comments

Posted By **srinivasu namburi** On 10-May-10 04:20:53 PM

Sir, Its really good example ..... If possible could u pls provide any example on hibernate caching ...first level and second level of caching techniques...

Posted By **Prabhas Raju** On 10-May-10 05:30:08 PM

Thank you very much sir.

iam searching for this from a week.

in interview of L&T Infotech they asked me the same question.

Thanks & Regards
Raju

Posted By **Puru** On 26-May-10 09:52:42 PM

I used the native sql approach and I am getting :"could not execute native bulk manipulation query error".
Below is my code:
Query q = session.createSQLQuery(" { call SP_LIB_DTL_INSERT(?,?,?,?) }");
q.setLong(0, lib.getBranch_Code());
q.setLong(1, lib.getAuth_Code());
q.setString(2, lib.getBookName());
q.setLong(3, lib.getISBN());
//@SuppressWarnings("unchecked")
System.out.println("before getting the list");
int retval=q.executeUpdate();

Appreciate your help

Posted By **sivagopal** On 18-Jun-10 01:58:06 AM

This is not really calling stored proc using hibernate, just obtained connection thru hibernate and used the same old approach of using prepare call method...I still believe there is no approach of calling stored proc, by using a hibernate config file...I am being stupid to raise this concern, your explanation is much appreciated...