/\* hibernate.cfg.xml \*/

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- 主要配置数据库的连接信息 -->

<property name="connection.username">root</property>

<property name="connection.password">123456</property>

<property name="connection.url">jdbc:mysql://localhost:3306/house</property>

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<!-- 方言 -->

<property name="dialect">org.hibernate.dialect.MySQLDialect</property>

<!-- 显示sql -->

<property name="show\_sql">true</property>

<!-- 格式化sql -->

<property name="format\_sql">true</property>

<!-- 创建数据库连接（session）的方式

每次打开数据库连接的时候，都会从线程池中查看有木有已经开打的数据库连接，

如果有的话，就用现成的数据库连接。如果没有的话，就新创建一个新的连接，然后放入线程池中

用这种方法打开的数据库连接，不需要.close关闭，只需要.commit()就可以关闭数据库连接

-->

<property name="current\_session\_context\_class">thread</property>

<!-- 导入映射文件

package 注解形式导入

-->

<mapping resource="com/lanou/entity/Type.hbm.xml"/>

<mapping resource="com/lanou/entity/District.hbm.xml"/>

<mapping resource="com/lanou/entity/House.hbm.xml"/>

<mapping resource="com/lanou/entity/Street.hbm.xml"/>

<mapping resource="com/lanou/entity/Users.hbm.xml"/>

</session-factory>

</hibernate-configuration>

/\* UsersTest.java \*/

package com.lanou.test;

import java.util.Iterator;

import java.util.List;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.Configuration;

import org.hibernate.service.ServiceRegistry;

import org.junit.BeforeClass;

import org.junit.Test;

import com.lanou.entity.Users;

public class UsersTest {

static Configuration cfg = null;

static SessionFactory sf = null;

Session session = null;

Transaction tx = null;

@BeforeClass

public static void init() {

// 读取hibernate.cfg.xml

cfg = new Configuration().configure();

ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder().applySettings(cfg.getProperties()).build();

// 创建session 工厂

sf = cfg.buildSessionFactory(serviceRegistry);

}

@Test

public void testQueryUser() {

// 获取session

session = sf.getCurrentSession();

// 事务开启

tx = session.beginTransaction();

// 编写hql

String hql = "from Users";

// 创建query

Query q = session.createQuery(hql);

// 获取所有用户信息

List<Users> list = q.list();

for (Users u : list) {

System.out.println(u.getId()+"-----"+u.getName());

}

tx.commit();

}

/\*\*

\* query.list() 一次性查出所有的信息

\* query.iterate() 先查出所有的id，当调用next(),

\* 根据id 查询这条行记录（n+1）sql 语句

\*/

@Test

public void test02() {

session = sf.getCurrentSession();

tx = session.beginTransaction();

String hql = "from Users";

Query q = session.createQuery(hql);

Iterator<Users> it = q.iterate();

while(it.hasNext()) {

Users u = it.next();

System.out.println(u.getId()+"----"+u.getName());

}

}

/\*\*

\* hql 参数

\*/

@Test

public void test03() {

session = sf.getCurrentSession();

tx = session.beginTransaction();

String hql = "from Users where id = ? and name = ?";

Query q = session.createQuery(hql);

//填充参数 ? 下标从0开始

q.setParameter(0, 1000);

q.setParameter(1, "admin");

// uniqueResult 获取的唯一结果集

Users u = (Users) q.uniqueResult();

System.out.println(u.getId()+"----"+u.getName());

}

@Test

public void test04() {

session = sf.getCurrentSession();

tx = session.beginTransaction();

String hql = "from Users where name = :name and password = :pwd";

Query q = session.createQuery(hql);

q.setParameter("name", "admin");

q.setParameter("pwd", "admin");

Users u = (Users) q.uniqueResult();

System.out.println(u.getId()+"----"+u.getName());

}

}