// not valid without active transaction： 没有开启事务

/\* UsersTest.java \*/

@Test

public void test05() {

List<Users> user = queryUser();

for (Users u : user) {

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

}

}

/\*\*

\* 动态查询

\* @return

\*/

public List<Users> queryUser(){

Users u = new Users();

// 条件

u.setName("admin");

u.setTelephone("18538320326");

// hashmap 和 hashtable 的区别

// 存放参数的map

Map<String, String> map = new HashMap<>();

StringBuffer hql = new StringBuffer("from Users where 1=1");

// 根据条件，设置参数

if(!StringUtils.isNullOrEmpty(u.getName())) {

hql.append(" and name = :name");

map.put("name", u.getName());

}

if(!StringUtils.isNullOrEmpty(u.getTelephone())) {

hql.append(" and telephone = :telephone");

map.put("telephone", u.getTelephone());

}

System.out.println(hql.toString());

session = sf.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql.toString());

// 遍历map

for (Entry<String, String> entry : map.entrySet()) {

// System.out.println("键："+entry.getKey());

// System.out.println("值："+entry.getValue());

q.setParameter(entry.getKey(), entry.getValue());

}

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

return list;

}

/\*\*

\* 求出总条数

\*/

@Test

public void testCount() {

// hql 支持聚合函数 count sum max min avg

String hql = "select count(1) from Users";

session = sf.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

// 默认查总条数，唯一结果集返回是long类型

long count = (long) q.uniqueResult();

System.out.println(count);

// 大的数值都可以采用BigDecimal

BigDecimal a = new BigDecimal(10);

BigDecimal b = new BigDecimal(3.0);

System.out.println(a.divide(b, 2, RoundingMode.HALF\_UP));

System.out.println(10 / 3.0);

}

/\*\*

\* content内容 context上下文

\*

\* 求一页显示的内容

\*/

@Test

public void testContent() {

String hql = "from Users where password = '123456'";

session = sf.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

// 分页方法 一页显示2条 求第二页

// 开始条数：(pageNo-1)\*pageSize 同 limit 计算一样

q.setFirstResult(2);

q.setMaxResults(2);

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

for (Users u : user) {

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

}

}

/\*\*

\* 为什么要用投影查询：

\* 因为我们查询部分属性，并且结果要封装到实体类中

\*/

/\*\*

\* 查询返回对象

\*/

@Test

public void list1() {

String hql = "select new Users(id, name) from Users";

session = sf.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

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

for (Users u : user) {

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

}

}

/\*\*

\* 直接查询

\*/

@Test

public void list2() {

String hql = "select id, name from Users";

session = sf.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Object[]> list = q.list();

for (Object[] o : list) {

System.out.println(o[0]+"---"+o[1]);

}

}

/\*\*

\* 查询返回Map键值对

\*/

@Test

public void list3() {

String hql = "select new Map(u.id as id, u.name as name) from Users u";

session = sf.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

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

for (Map m : list) {

System.out.println(m.get("id")+"----"+m.get("name"));

}

}

/\* TestCriteriaUsers.java \*/

public class TestCriteriaUsers {

Session session = null;

Transaction tx = null;

/\*\*

\* Criteria 查询的条件查询

\*/

@Test

public void testQueryAllUser() {

session = HB.getCurrentSession();

tx = session.beginTransaction();

Criteria ct = session.createCriteria(Users.class);

ct.add(Restrictions.eq("name", "admin")).add(Restrictions.eq("password", "admin"));

// ct.add(Restrictions.eq("name", "admin"));

// ct.add(Restrictions.eq("password", "admin"));

List<Users> user = ct.list();

for (Users u : user) {

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

}

}

/\*\*

\* Criteria between 查询

\*/

@Test

public void testQueryHouse() {

session = HB.getCurrentSession();

tx = session.beginTransaction();

Criteria ct = session.createCriteria(House.class);

String startTime = "2018-01-01";

String endTime = "2018-07-31";

//string >> date

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

// 检查时异常

// start 和 end为局部变量，局部变量在使用的时候，一定要赋初始值，为null

Date start =null;

Date end=null;

try {

start = sdf.parse(startTime);

end = sdf.parse(endTime);

} catch (Exception e) {

e.printStackTrace();

}

ct.add(Restrictions.between("pubdate", start, end));

List<House> hs = ct.list();

for (House h : hs) {

System.out.println(h.getId()+"---"+h.getTitle());

}

}

/\*\*

\* 排序

\*/

@Test

public void testOrderBy() {

session = HB.getCurrentSession();

tx = session.beginTransaction();

Criteria ct = session.createCriteria(House.class);

ct.addOrder(Order.desc("id"));

List<House> hs = ct.list();

for (House h : hs) {

System.out.println(h.getId()+"---"+h.getTitle());

}

}

/\*\*

\* 分页查询

\*/

@Test

public void testPage() {

session = HB.getCurrentSession();

tx = session.beginTransaction();

// 一页显示2条 第二页

Criteria ct = session.createCriteria(House.class);

ct.addOrder(Order.desc("id"));

ct.setFirstResult(2);

ct.setMaxResults(2);

List<House> hs = ct.list();

for (House h : hs) {

System.out.println(h.getId()+"---"+h.getTitle());

}

}

/\*\*

\* 根据 title price floorage(面积) 动态查询

\*/

@Test

public void testQuery() {

// 条件

House h = new House();

h.setTitle("别墅");

h.setPrice(10000d);

h.setFloorage(900);

session = HB.getCurrentSession();

tx = session.beginTransaction();

Criteria ct = session.createCriteria(House.class);

if (!StringUtils.isNullOrEmpty(h.getTitle())) {

ct.add(Restrictions.like("title", "%"+h.getTitle()+"%"));

}

if (h.getPrice()!=null) {

ct.add(Restrictions.le("price", h.getPrice()));

}

if (h.getFloorage()!=null) {

ct.add(Restrictions.le("floorage", h.getFloorage()));

}

List<House> hs = ct.list();

for (House h1 : hs) {

System.out.println(h1.getId()+"---"+h1.getTitle());

}

}

@Test

public void testMonth() {

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

Date d = new Date();

System.out.println("当前日期："+sdf.format(d));

Calendar cl = Calendar.getInstance();

cl.setTime(d);

cl.add(Calendar.MONTH, -1);

Date lastMoth = cl.getTime();

System.out.println("上一个月日期："+sdf.format(lastMoth));

}

/\* HB.java \*/

package com.lanou.util;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.Configuration;

import org.hibernate.service.ServiceRegistry;

public class HB {

public static SessionFactory sf;

static {

try {

Configuration cfg = new Configuration().configure();

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

sf = cfg.buildSessionFactory(serviceRegistry);

} catch (HibernateException e) {

e.printStackTrace();

}

}

/\*\*

\* 获取sessoin

\* @return

\*/

public static Session getCurrentSession() {

return sf.getCurrentSession();

}

}