<https://www.cnblogs.com/hellokitty1/p/5047796.html>

# [Hibernate 分页 查询](http://www.cnblogs.com/hellokitty1/p/5047796.html)

 昨天的作业  分页：

主要的代码块：（明天实现分页的封装）

[复制代码](javascript:void(0);)

1 package com.cy.beans;

2

3 import java.util.List;

4

5 /\*\*

6 \* 定义一个分页对象

7 \* @author acer

8 \*

9 \*/

10 public class Pager {

11

12 private int page;//当前页码

13 private int pageTotal;//总页码

14 private int rowsTotal;//总条数

15 private int rows;//每页显示条数

16 private String hql;//分页语句

17 private List<?> list;//返回的数据集合

18

19

20 public Pager() {

21 super();

22 }

23

24 public String getHql() {

25 return hql;

26 }

27

28

29

30 public void setHql(String hql) {

31 this.hql = hql;

32 }

33

34

35

36 public int getPage() {

37 return page;

38 }

39 public void setPage(int page) {

40 this.page = page;

41 }

42 public int getPageTotal() {

43 return pageTotal;

44 }

45 public void setPageTotal(int pageTotal) {

46 this.pageTotal = pageTotal;

47 }

48 public int getRowsTotal() {

49 return rowsTotal;

50 }

51 public void setRowsTotal(int rowsTotal) {

52 this.rowsTotal = rowsTotal;

53 pageTotal = rowsTotal%rows==0 ? rowsTotal/rows : rowsTotal/rows+1;

54 }

55 public int getRows() {

56 return rows;

57 }

58 public void setRows(int rows) {

59 this.rows = rows;

60 }

61 public List<?> getList() {

62 return list;

63 }

64 public void setList(List<?> list) {

65 this.list = list;

66 }

67

68

69 @Override

70 public String toString() {

71 return "Pager [list=" + list + ", page=" + page + ", pageTotal="

72 + pageTotal + ", rows=" + rows + ", rowsTotal=" + rowsTotal

73 + "]";

74 }

75

76

77 }

[复制代码](javascript:void(0);)

 UserDaoImpl.java中的分页方法代码：

[复制代码](javascript:void(0);)

1 public Pager pagerff(Pager p, Map<String, Object> pram) {

2 Session session = null;

3 Transaction tx = null;

4 try {

5 session=HibernateUtils.getSessionFactory().openSession();

6 tx=session.beginTransaction();

7 String hql=p.getHql();//获取查询语句

8 Query query= session.createQuery(hql).setCacheable(true);

9 //设置参数

10 query.setProperties(pram);

11 //查询具体数据

12 int count=query.list().size();

13 p.setRowsTotal(count);

14 int nowPage=1;

15 if(p.getPage()>0){

16 nowPage=p.getPage();

17 }

18 //指定从那个对象开始查询，参数的索引位置是从0开始的，

19 query.setFirstResult((p.getPage()-1)\*p.getRows());

20 //分页时，一次最多产寻的对象数

21 query.setMaxResults(p.getRows());

22 List<?> list1=query.list();

23 p.setList(list1);

24 tx.commit();

25

26 } catch (Exception e) {

27 e.printStackTrace();

28 tx.rollback();

29 }finally{

30 session.close();

31 }

32 return p;

33 }

[复制代码](javascript:void(0);)

这个分页里主要使用了两个方法。

setFirstResult（）：设置从哪条数据开始查询的（分页的索引是从0开始的）；

setProperties（）：设置每页显示的条数。  
  
  
**hibernate查询实现**

## ****hibernate中查询对象的方式有：OID查询，HQL查询，标准对象（条件）查询，本地sql查询。****

**使用例子UserBean.java**

[复制代码](javascript:void(0);)

1 package com.cy.beans;

2

3 import java.io.Serializable;

4

5 public class UserBean implements Serializable {

6

7 private static final long serialVersionUID = 1L;

8

9 private Integer id;

10 private String userName;

11 private String password;

12 public UserBean() {

13 super();

14 }

15

16 public Integer getId() {

17 return id;

18 }

19

20 public void setId(Integer id) {

21 this.id = id;

22 }

23

24 public String getUserName() {

25 return userName;

26 }

27 public void setUserName(String userName) {

28 this.userName = userName;

29 }

30 public String getPassword() {

31 return password;

32 }

33 public void setPassword(String password) {

34 this.password = password;

35 }

36 @Override

37 public String toString() {

38 return "UserBean [id=" + id + ", userName=" + userName + ", password="

39 + password + "]";

40 }

41

42

43 }

[复制代码](javascript:void(0);)

**先要定义IUserDao.java的接口。**

**一：OID查询：就是根据对象的id来查询（主要看看UserDaoImpl.java中实现的方法）**

[复制代码](javascript:void(0);)

1 public UserBean getUser(Class<?> cls, Serializable pk) {

2 // Class<?> cls 泛型，表示某一类的Object。

3 Session session = null;//获得session

4 Transaction tx = null;//事务

5 UserBean user=null; //对象

6 try {

7 session=HibernateUtils.getSessionFactory().openSession();

8 tx=session.beginTransaction();//开启事务

9 user=(UserBean)session.get(cls, pk);//查询 这里的pk指的就是id

10 tx.commit();//提交事务

11 } catch (Exception e) {

12 e.printStackTrace();

13 tx.rollback();//回滚事务

14 }finally{

15 session.close();//关闭session

16 }

17 return user;

18 }

[复制代码](javascript:void(0);)

**二：HQL查询：**

**hql（Hibernate Query Language）是一种面向对象的查询语言。也是在Hibernate应用最广的一种查询方式。**

**完整的HQL语法：**

**Select/update/delete…… from …… where …… group by …… having …… order by …… asc/desc**

**示例1：**

**UserDaoImpl.java**

[复制代码](javascript:void(0);)

1 public List<?> findUser(String sql, Map<String, Object> pram) {

2 Session session = null;

3 Transaction tx = null;

4 List<?> list=null;

5 try {

6 session=HibernateUtils.getSessionFactory().openSession();

7 tx=session.beginTransaction();

8 Query query= session.createQuery(sql).setCacheable(true);

9 String userName=pram.get("userName").toString();

10 String password=pram.get("password").toString();

11 query.setString("userName", '%'+userName+'%');

12 query.setString("password",password);

13 list=query.list();

14 tx.commit();

15

16 } catch (Exception e) {

17 e.printStackTrace();

18 tx.rollback();

19 }finally{

20 session.close();

21 }

22 return list;

23 }

[复制代码](javascript:void(0);)

**UserServerImpl.java**

1 @Override

2 public List<?> findUser(Map<String,Object>pram) {

3 //模糊查询

4 String hql="from UserBean where 1=1 and userName like :userName and password =:password";

5 return iusUserDao.findUser(hql, pram);

6 }

**示例2：（这个就直接写了）**

**String hql="from UserBean as user where user.userName='kitty2'";**

**Query query=session.createQuery(hql);**

**List userList=query.list();**

**通过上面的例子可以看出HQL查询的基本操作步骤为：**

**1 创建Query对象：** Query query= session.createQuery(sql)

2  动态绑定参数

3 通过调用Query对象的list()方法执行查询数据

注意：HQL语句本身大小写无关，但是其中出现的类名和属性名必须注意大小写区分。

## ****三：条件查询：主要用了Criteria，Criterion接口，Expression和Restrictions类组成。****

UserDaoImpl.java

[复制代码](javascript:void(0);)

1 @Override

2 public List<?> findUser(Class<?> cls, String userName) {

3 Session session = null;

4 Transaction tx = null;

5 List<?> list=null;

6 try {

7 session=HibernateUtils.getSessionFactory().openSession();

8 tx=session.beginTransaction();

9 //通过session的createCriteria()方法创建Criteria对象

10 Criteria criteria = session.createCriteria(cls);

11 //设定查询条件

12 //(String propertyName, String value, MatchMode matchMode)

13 /\* MatchMode.START：字符串在最前面的位置.相当于"like 'value%'"

14 MatchMode.END：字符串在最后面的位置.相当于"like '%value'"

15 MatchMode.ANYWHERE：字符串在中间匹配.相当于"like '%value%'"

16 MatchMode.EXACT：字符串精确匹配.相当于"like 'value'" \*/

17 criteria.add(Restrictions.like("userName", userName, MatchMode.ANYWHERE));

18 //调用Criteria类的list()方法执行查询

19 list = criteria.list();

20 tx.commit();

21

22 } catch (Exception e) {

23 e.printStackTrace();

24 tx.rollback();

25 }finally{

26 session.close();

27 }

28 return list;

29 }

[复制代码](javascript:void(0);)

通过以上可以看出使用这个方法来查询数据有个局限就是要在session的条件下才可以。所以我们通常是使用离线条件查询

## ****离线条件查询：DetachedCriteria(离线条件查询)，可以脱离Session对象单独存在。（在项目中也多用离线条件查询）。****

示例：

[复制代码](javascript:void(0);)

1 @Override

2 public List<?> findUser(Class<?> cls, String userName) {

3 Session session = null;

4 Transaction tx = null;

5 List<?> list=null;

6 try {

7 session=HibernateUtils.getSessionFactory().openSession();

8 tx=session.beginTransaction();

9 DetachedCriteria dc=DetachedCriteria.forClass(cls);

10 //创建条件查询

11 dc.add(Restrictions.eq("userName", "kitty"));

12 //绑定当前的session

13 Criteria c=dc.getExecutableCriteria(session);

14 list=c.list();

15 tx.commit();

16

17 } catch (Exception e) {

18 e.printStackTrace();

19 tx.rollback();

20 }finally{

21 session.close();

22 }

23 return list;

24 }

[复制代码](javascript:void(0);)

Restrictions常用限定查询方法：

Restrictions.eq 等于

Restrictions.allEq 使用Map，使用key/value进行多个等于的比对

Restrictions.gt 大于 >

Restrictions.ge 大于等于 >=

Restrictions.lt 小于 <

Restrictions.le 小于等于 <=

Restrictions.between 对应SQL的BETWEEN子句

Restrictions.like 对应SQL的LIKE子句

Restrictions.in 对应SQL的in子句

Restrictions.and and关系

Restrictions.or or关系

## ****四：本地SQL查询****

对原生SQL查询执行的控制是通过SQLQuery接口进行的，通过执行Session.createSQLQuery()获取这个接口。

String sql="select \*　from t\_user";  
SQLQuery query=session.createSQLQuery(sql);  
query.addEntity(UserBean.class);  
list=query.list();

## 