Query 对象

- 1、使用 Query 对象,不需要写 sql 语句, 但是需要写 hql 语句
- 1)、hql: hibernate query language, hibernate提供的查询语句, hql语句 与 普通 sql 语句很相似
 - 2)、hql 和 SQL的区别:
 - -使用 SQL 操作表,操作表字段。
 - -使用 hql 操作实体类 和 属性。
 - 2、查询所有表中数据的 hql 语句:
 - 1)、from 实体类名称;
 - 3、Query 对象的使用
 - 1)、创建 Query 对象(参数是写 hql 语句)
 - 2)、调用 query 中的方法得到结果(返回结果是一个 List 结果集)

```
1
       @Test
2
       public void testQuery() {
3
            Session session = null;
            Transaction transaction = null;
4
5
            try {
6
7
                session = HibernateUtils.getSessionObj();
                transaction = session.beginTransaction();
8
9
                Query query = session.createQuery("from User");
                List<User> list = query.list();
10
                for (User user : list) {
11
                    System.out.println(user);
12
13
                }
14
15
                transaction.commit();
            } catch (Exception e) {
16
                transaction.rollback();
17
18
            }
19
        }
```

Criteria 对象

1、使用这个对象查询操作,但是使用这个对象时候,不需要写语句,直接调用方法实现。

2、实现过程:

- 1)、创建 criteria 对象 (传入参数是返回类型名)。
- 2)、调用对象里面的方法得到结果。

```
1
       @Test
2
       public void testCriteria() {
3
            Session session = null;
            Transaction tx = null;
4
5
            try {
                session = HibernateUtils.getSessionObj();
6
7
                tx = session.beginTransaction();
8
                tx.begin();
9
                Criteria criteria = session.createCriteria(User.class);
                List<User> list = criteria.list();
10
                tx.commit();
11
12
                for (User user : list) {
13
                    System.out.println(user);
14
15
                }
            } catch (Exception e) {
16
17
                tx.rollback();
                e.printStackTrace();
18
19
            }
        }
20
```

SQLQuery 对象

- 1、使用 hibernate 的时候,调用底层 SQL 实现操作
- 2、实现过程:
 - 1)、创建对象 SQLQuery 对象
 - 2)、调用对象方法得到结果(返回的 list 结果是一个数组)

```
1    @Test
2    public void testSqlQuery() {
3         Session session = null;
4         Transaction tx = null;
```

```
5
           try {
                session = HibernateUtils.getSessionObj();
6
7
                tx = session.beginTransaction();
8
               tx.begin();
9
                SQLQuery sqlQuery = session.createSQLQuery("SELECT * FROM
10
   `tb_user`");
                //返回的结果是一个对象数组
11
                List<Object[]> list = sqlQuery.list();
12
               for (Object[] objects : list) {
13
                    System.out.println(Arrays.toString(objects));
14
15
                }
                tx.commit();
16
           } catch(Exception e){
17
18
               tx.rollback();
                e.printStackTrace();
19
20
           }
       }
21
```

如果需要返回实体对象需要进行一定的包装处理

先调用 sqlQuery 中的addEntity(Class<>);

```
1
       @Test
2
       public void testSqlQuery() {
3
           Session session = null;
           Transaction tx = null;
4
5
           try {
                session = HibernateUtils.getSessionObj();
6
7
               tx = session.beginTransaction();
               tx.begin();
8
9
               SQLQuery sqlQuery = session.createSQLQuery("SELECT * FROM
10
   `tb_user`");
11
               返回对象是user对象的集合,设定实体对象
   //
12
                sqlQuery.addEntity(User.class);
13
14
               List<User> list = sqlQuery.list();
15
               for (User user : list) {
16
                   System.out.println(user);
17
               }
18
19
```

```
//返回的结果是一个对象数组
20
   //
               List<Object[]> list = sqlQuery.list();
21
              for (Object[] objects : list) {
22 //
                  System.out.println(Arrays.toString(objects));
23 //
24 //
               }
25
               tx.commit();
26
           } catch(Exception e){
27
               tx.rollback();
28
               e.printStackTrace();
29
30
           }
       }
31
```