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）

```
@Test
public void testQuery() {
    Session session = null;
    Transaction transaction = null;
    try {

        session = HibernateUtils.getSessionObj();
        transaction = session.beginTransaction();
        Query query = session.createQuery("from User");
        List<User> list = query.list();
        for (User user : list) {
            System.out.println(user);
        }

        transaction.commit();
    } catch (Exception e) {
        transaction.rollback();
    }
}
```

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。

```
@Test
public void testCriteria() {
    Session session = null;
    Transaction tx = null;
    try {
        session = HibernateUtils.getSessionObj();
        tx = session.beginTransaction();
        tx.begin();
        Criteria criteria = session.createCriteria(User.class);
        List<User> list = criteria.list();
        tx.commit();

        for (User user : list) {
            System.out.println(user);
        }
    } catch (Exception e) {
        tx.rollback();
        e.printStackTrace();
    }
}
```

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）

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

```
        try {
            session = HibernateUtils.getSessionObj();
            tx = session.beginTransaction();
            tx.begin();

            SQLQuery sqlQuery = session.createSQLQuery("SELECT * FROM
`tb_user`");
            //返回的结果是一个对象数组
            List<Object[]> list = sqlQuery.list();
            for (Object[] objects : list) {
                System.out.println(Arrays.toString(objects));
            }
            tx.commit();
        } catch(Exception e){
            tx.rollback();
            e.printStackTrace();
        }
    }
```

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<>);

```

    @Test
    public void testSqlQuery() {
        Session session = null;
        Transaction tx = null;
        try {
            session = HibernateUtils.getSessionObj();
            tx = session.beginTransaction();
            tx.begin();

            SQLQuery sqlQuery = session.createSQLQuery("SELECT * FROM
`tb_user`");

//          返回对象是user对象的集合，设定实体对象
            sqlQuery.addEntity(User.class);

            List<User> list = sqlQuery.list();
            for (User user : list) {
                System.out.println(user);
            }

```

```
            //返回的结果是一个对象数组
//          List<Object[]> list = sqlQuery.list();
//          for (Object[] objects : list) {
//              System.out.println(Arrays.toString(objects));
//          }
            tx.commit();

        } catch(Exception e){
            tx.rollback();
            e.printStackTrace();
        }
    }
```