【济南中心】JavaEE就业班同步笔记第三阶段：Hibernate-part04

小鲁哥哥 • 2017-6-25 17:18:22

【济南中心】JavaEE就业班同步笔记第三阶段：Hibernate-part04

1.1 Hibernate的查询的方式

1.1.1 Hibernate的五种查询的方式

对象图导航查询：通过某个对象获得到其关联对象。

```
Customer c = session.get(Customer.class,1l);
Set<LinkMan> linkMans = c.getLinkMans();
```

OID检索：根据对象的ID查询

```
get/load();
```

HQL检索：通过传入的HQL进行查询。

HQL：Hibernate Query Language.语法与SQL类似。面向对象的查询方式。

QBC检索：通过传入Criteria对象进行查询。

QBC：Query By Criteria.更加面向对象的方式查询。

SQL检索：通过SQL查询。

1.1.2.1 基本查询

[mw_shl_code=java,true]@Test

```
/**
 * 基本的查询
```

```
 */
public void demo2(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();
	// HQL的基本查询
	/*Query query = session.createQuery("from Customer");
	List<Customer> list = query.list();

	for (Customer customer : list) {
		System.out.println(customer);
	}*/

	tx.commit();
}
```
[/mw_shl_code]1.1.2.2 别名查询

[mw_shl_code=java,true]@Test
```
/**
 * 别名的查询
 */
public void demo3(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();
	// HQL的别名查询
	/*Query query = session.createQuery("from Customer c");
	List<Customer> list = query.list();

	for (Customer customer : list) {
		System.out.println(customer);
	}*/
	Query query = session.createQuery("select c from Customer c");
	List<Customer> list = query.list();

	for (Customer customer : list) {
		System.out.println(customer);
	}
	tx.commit();
}
```
[/mw_shl_code]1.1.2.3 排序查询

[mw_shl_code=java,true]@Test
/**
 * 排序的查询
 */
public void demo4(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();
	// HQL的排序查询
	// 排序默认是升序的默认值是asc　降序是desc
	Query query = session.createQuery("from Customer c order by cust_id desc");
	List<Customer> list = query.list();

	for (Customer customer : list) {
		System.out.println(customer);
	}
	tx.commit();
}[/mw_shl_code]1.1.2.4 条件查询

[mw_shl_code=java,true]@Test
/**
 * 条件查询
 */
public void demo5(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();
	// HQL的条件查询
	// 1.按位置绑定参数
	/*Query query = session.createQuery("from Customer where cust_name = ?");
	query.setParameter(0, "梁宝强");
	List<Customer> list = query.list();*/

	/*Query query = session.createQuery("from Customer where cust_name = ? and cust_source= ?");
	query.setParameter(0, "梁宝强");
	query.setParameter(1, "网络营销");
	List<Customer> list = query.list();
	for (Customer customer : list) {
		System.out.println(customer);

```
        }*/

        // 2.按名称绑定参数
        Query query = session.createQuery("from Customer where cust_name = :aaa and cust_source
= :bbb");
        query.setParameter("aaa", "宋天一");
        query.setParameter("bbb", "电话销售");
        List<Customer> list = query.list();
        for (Customer customer : list) {
            System.out.println(customer);
        }
        tx.commit();
    }[/mw_shl_code]
```

1.1.2.5 分组统计查询

```
    [mw_shl_code=java,true]@Test
    /**
     * HQL的分组统计查询
     */
    public void demo6(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        // 聚集函数的使用
        /*Query query = session.createQuery("select count(*) from Customer");
        Long count = (Long) query.uniqueResult();

        System.out.println(count);*/
        // 分组统计每个客户ID下的联系人。
        /*Query query = session.createQuery("select count(*) from LinkMan l group by l.customer.cust_i
d");
        List<Long> list = query.list();
        for (Long count : list) {
            System.out.println(count);
        }*/

        Query query = session.createQuery("select l.customer.cust_id,count(*) from LinkMan l group by
l.customer.cust_id");
```

```
        List<Object[]> list = query.list();
        for (Object[] obj : list) {
                System.out.println(Arrays.toString(obj));
        }
        tx.commit();
}
```
[/mw_shl_code]1.1.2.6 分页查询

[mw_shl_code=java,true]@Test
```
/**
 * HQL的分页查询
 */
public void demo7(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        Query query = session.createQuery("from LinkMan");
        query.setFirstResult(20);
        query.setMaxResults(10);
        List<LinkMan> list = query.list();
        for (LinkMan linkMan : list) {
                System.out.println(linkMan);
        }

        tx.commit();
}
```
[/mw_shl_code]1.1.2.7 投影查询

[mw_shl_code=java,true]@Test
```
/**
 * HQL的投影查询：查询部分属性
 */
public void demo8(){
        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        // 查询某个属性
        /*Query query = session.createQuery("select cust_name from Customer");
        List<String> list = query.list();
        for (String name : list) {
```

```
            System.out.println(name);
        }*/

        // 查询多个属性
        /*Query query = session.createQuery("select cust_name,cust_source from Customer");
        List<Object[]> list = query.list();
        for (Object[] objs : list) {
            System.out.println(Arrays.toString(objs));
        }*/

        // 查询多个属性：封装到对象中.
        Query query = session.createQuery("select new Customer(cust_name,cust_source) from Custo
mer");
        List<Customer> list = query.list();
        for (Customer customer : list) {
            System.out.println(customer);
        }
        tx.commit();
    }
```
[/mw_shl_code]1.1.3 QBC的检索方式：

1.1.3.1 基本查询

[mw_shl_code=java,true]
```
@Test
/**
 * 基本查询
 */
public void demo1(){
    Session session = HibernateUtils.getCurrentSession();
    Transaction tx = session.beginTransaction();

    Criteria criteria = session.createCriteria(Customer.class);
    List<Customer> list = criteria.list();

    for (Customer customer : list) {
        System.out.println(customer);
    }
    tx.commit();
}
```
[/mw_shl_code]1.1.3.2 排序查询

[mw_shl_code=java,true]@Test
```
/**
 * 排序查询
 */
public void demo2(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();

	Criteria criteria = session.createCriteria(Customer.class);
	// 设置排序:
	//criteria.addOrder(Order.asc("cust_id"));// 升序
	criteria.addOrder(Order.desc("cust_id"));// 降序
	List<Customer> list = criteria.list();

	for (Customer customer : list) {
		System.out.println(customer);
	}
	tx.commit();
}
```
[/mw_shl_code]

1.1.3.3 条件查询

[mw_shl_code=java,true]@Test
```
/**
 * 条件查询
 */
public void demo3(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();

	Criteria criteria = session.createCriteria(Customer.class);
	// criteria.add(Restrictions.like("cust_name", "%强%"));
	criteria.add(Restrictions.like("cust_name", "强", MatchMode.ANYWHERE));
	criteria.add(Restrictions.eq("cust_source", "网络营销"));
	List<Customer> list = criteria.list();
	for (Customer customer : list) {
		System.out.println(customer);
	}
	tx.commit();
```

}[/mw_shl_code]1.1.3.4 分页查询

[mw_shl_code=java,true]@Test

```
/**
 * 分页查询
 */
public void demo4(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();

	Criteria criteria = session.createCriteria(LinkMan.class);
	criteria.setFirstResult(10);
	criteria.setMaxResults(10);
	List<LinkMan> list = criteria.list();
	for (LinkMan linkMan : list) {
		System.out.println(linkMan);
	}
	tx.commit();
```

}[/mw_shl_code]1.1.3.5 统计查询

[mw_shl_code=java,true]@Test

```
/**
 * 统计查询
 */
public void demo5(){
	Session session = HibernateUtils.getCurrentSession();
	Transaction tx = session.beginTransaction();

	Criteria criteria = session.createCriteria(Customer.class);
	criteria.setProjection(Projections.rowCount());
	Long count = (Long) criteria.uniqueResult();
	System.out.println(count);
	tx.commit();
```

}[/mw_shl_code]1.1.3.6 离线条件查询：DetachedCriteria.

DetachedCriteria对象可以脱离session使用。使用离线条件查询对象的优点：

[mw_shl_code=java,true]@Test

```
/**
 * 离线条件查询
```

```
     */
    public void demo6(){
        DetachedCriteria detachedCriteria = DetachedCriteria.forClass(Customer.class);
        detachedCriteria.add(Restrictions.like("cust_name", "%强%"));

        Session session = HibernateUtils.getCurrentSession();
        Transaction tx = session.beginTransaction();

        Criteria criteria = detachedCriteria.getExecutableCriteria(session);
        List<Customer> list = criteria.list();
        for (Customer customer : list) {
            System.out.println(customer);
        }
        tx.commit();
    }
```
[/mw_shl_code]1.1.4 Hibernate的多表查询

1.1.4.1 SQL中多表查询

【交叉连接】

select * from A,B;

【内连接】

显示内连接:inner join(inner 可以省略)

Select * from A inner join B on 条件;

隐式内连接:

Select * from A,B where 条件;

【外连接】

左外连接：left outer join

Select * from A left outer join B on 条件;

右外连接：right outer join

Select * from A right outer join B on 条件;

1.1.4.2 Hibernate中的多表连接查询

【交叉连接】

交叉连接

【内连接】

显示内连接

隐式内连接

迫切内连接
【外连接】
左外连接
右外连接
迫切左外连接

1.2 Hibernate的抓取策略

1.2.1 延迟加载

1.2.1.1 延迟加载的概述

lazy延迟加载：什么时候使用什么时候再去查询。

1.2.1.2 延迟加载的分类

【类级别延迟】

使用延迟加载的方法查询某个类的时候是否采用的延迟称为是类级别的延迟。默认值是true。

Customer customer = session.load(Customer.class,1l);// 默认就会采用延迟加载，这种称为是类级别的延迟。

类级别延迟加载失效：

* final修饰这个类,不能产生代理类,延迟加载就会失效。
* 在<class>上配置lazy=”false”

【关联级别的延迟】

查询到某个对象以后，获得其关联的对象。查询其关联对象的时候是否采用的延迟。称为是关联级别的延迟。

Customer c = session.get(Customer.class,1l);

c.getLinkMans(); // 查询关联对象的时候，是否采用延迟加载。

关联级别的延迟往往会与抓取策略一起使用，优化程序。（关联级别的延迟在<set>或者是<many-to-one>标签上的延迟加载）

1.2.2 抓取策略

抓取策略指的是查找到某个对象后，抓取其关联的对象的时候采用的策略。抓取策略就是在关联对象的配置上（<set>和<many-to-one>）配置fetch属性。

1.2.2.1 set上的fetch和lazy

fetch:抓取策略，控制SQL语句的发送的格式。

* select　　:默认值。发送一条select语句查询关联对象。
* join　　:发送一条迫切左外连接查询关联对象。
* subselect　　:发送一条子查询查询关联对象。

lazy:延迟加载，控制SQL语句的发送的时候。
* true　　:默认值。采用延迟加载。
* false　　:不采用延迟加载。
* extra　　:及其懒惰。

1.2.2.2 many-to-one上的fetch和lazy
fetch:抓取策略，控制SQL语句的发送的格式。
* select ： 默认值.发送一条select语句查询关联对象。
* join　　： 发送一条迫切左外连接查询关联对象。

lazy：延迟加载，控制SQL的发送的时机。
* proxy　　： 默认值。是否采用延迟，需要由另一方类上的延迟加载来决定。
* false　　： 不采用延迟加载。
* no-proxy：不用研究

1.2.3 批量抓取
批量抓取：查询了多个客户的时候，查询多个客户下的所有联系人。
1.2.3.1 查询客户批量抓取联系人
在Customer.hbm.xml中<set>上配置batch-size
1.2.3.2 查询联系人批量抓取客户
在Customer.hbm.xml中<class>上配置batch-size

回帖

1条回帖

feiling

总结的不错

沙发 • 2017-6-25 17:40:29

回帖

来自宇宙超级黑马专属安卓客户端