HQL

了解HQL语句
准备查询
查询子句

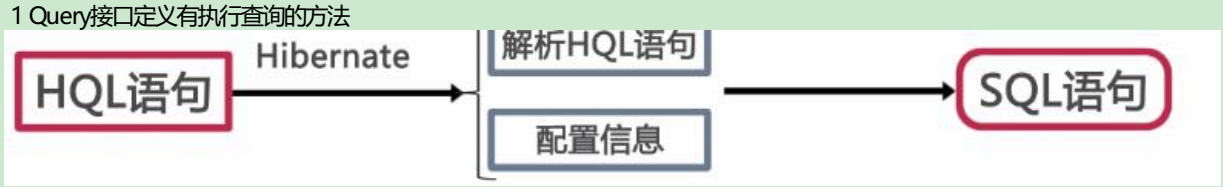
检索对象--from子句
选择--select子句
限制--where子句
排序-order by子句

HQL定义

HQL语句形式
select。。。from。。。where。。。group by。。。having。。。order by。。。

HQL对关键字大小写是不敏感的（并不区分大小写）

习惯小写
1 Query接口定义有执行查询的方法



2 Query借口支持方法链编程风格，使得程序代码更为简洁（动态设置）
Query实例的创建
1 session的createQuery（）方法创建Query实例
2createQuery方法包含一个HQL语句参数，createQuery（hql）

2 Query接口中的list方法执行HQL查询
list方法返回结果数据类型为java.util.list，list集合中存放符合查询条件的是持久化对象

```
@Test
public  void testAccount(){
String hql="from Account ";

Query query = session.createQuery(hql);

List<Account> accountList=query.list();

for (Account account : accountList) {

System.out.println(account);
    }

}
```

查询对象-from子句

1.from子句

2.from子句中持久化类的引用

3.别名的使用

这里体现了hibernate的懒加载

```
System.out.println("seller's name :"+c.getSeller().getName());
```

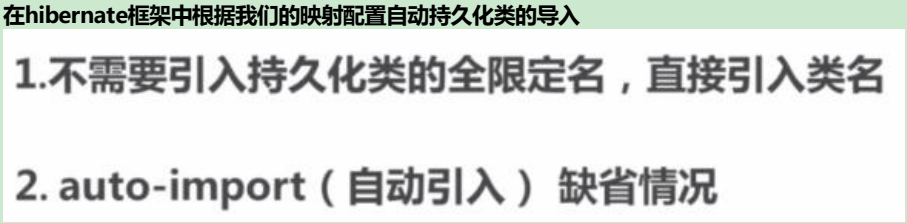
```
        seller0_.NAME as NAME0_0_,
        seller0_.TEL as TEL0_0_,
        seller0_.ADDRESS as ADDRESS0_0_,
        seller0_.WEBSITE as WEBSITE0_0_,
        seller0_.STAR as STAR0_0_,
        seller0_.BUSINESS as BUSINESS0_0_
    from
        SELLER seller0_
    where
        seller0_.ID=?
seller's name :A服装店
name:女士套装
seller's name :A服装店
name:男士西服
seller's name :A服装店
name:笔记本电脑
Hibernate:
    select
        seller0_.ID as ID0_0_,
        seller0_.NAME as NAME0_0_,
```

默认是不需要查询外键对应信息对应的数据的 只有在具体需要的时候才查询

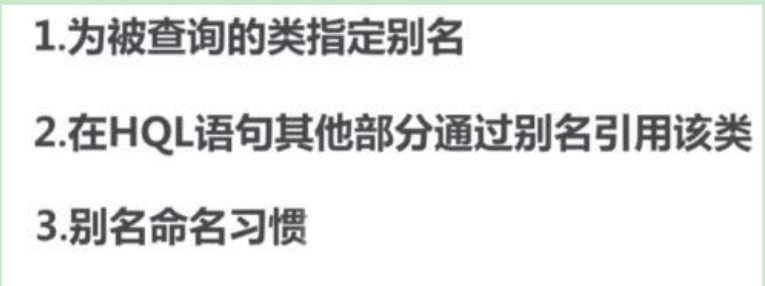
1 不需要引入持久化类的全限类名，直接引入类名(在java代码中，必须指定全限类名，才知道去哪里获取这个类)
全限定名：com.imooc.model.Seller

```
from Seller
```

在hibernate框架中根据我们的映射配置自动持久化类的导入



from子句中别名的应用



要在原时，保持代码的可读性

查询对象select子句

1、String hql="select id,name from Test1 as s ";

2、String hql="select s.id,s.name from Test1 as s ";

Ps1：别名使用不是必须的，但是后期多个表结合起来时，避免多个表中有字段名重复，推荐还是养成别名操作的习惯。

Ps2：当只查询一个属性时，即select 只有一个的时候，返回的类型是对象类型，而不是对象数组。

在HQL语句中的select中，如果查询的是多个字段，则返回的是Object[] 如果查询的是一个字段，只返回一个Object.

distinct

三种查询结果的返回，根据自己的习惯


```
String hql="select s.name from Seller s";
        Query query=session.createQuery(hql);
        List<Object> list = query.list();

        for(Object obj : list){
            System.out.println("name:"+obj);
        }
    }
```


```
String hql="select new list(s.name,s.tel,s.address) from Seller s";
        Query query=session.createQuery(hql);
        List<List>lists =query.list();
        for (List list : lists) {
            System.out.println("name"+list.get(0));
            System.out.println("tel"+list.get(1));
            System.out.println("address"+list.get(2));
```


```
String hql="select new Map(s.name as name,s.tel as tel,s.address as address) from Seller s";
            Query query=session.createQuery(hql);
        List<Map> maps=query.list();
        for (Map map : maps) {
            System.out.println("name:"+map.get("name"));
            System.out.println("tel:"+map.get("tel"));
            System.out.println("address"+map.get("address"));
        }
```

通过MAP来获取结果，就可以使用别名来获取对应的属性信息

以自定义类型返回

1.持久化类中定义对应的构造器

2.select子句中调用定义的构造器

我们也可以通过**构造器来设置返回的类型**，比如说：select new Seller(s.name,s.tel,s.addredd) from Seller s

如果我们制定默认构造器，就会指定默认构造器，来进行对象的创建与封装，如果没有指定构造器的话，就会出错
默认构造器是需要的，因为，在Hibernate没有指定的查询的放回集合时候，Hibernate会自动去找默认构造器，如果不存在，则会出现异常

distinct关键字
1 使用distinct关键字来去出查询结果中的重复元素

```
select  c.sex  from  Customer c;
```

这样查询会有重复出现

```
select  distinct  c.sex  from  Customer c;
```

这样就没有重复出现了

查询对象where子句

比较运算
1.= <>,<,>,>=,<=
form Commdity c where c.price>400;
2 null值判断--is [not] null

```
form Commdity c where  c.description is  null;
form Commdity c where  c.description  =  null;
```

这样写在HQL中是可以的，在sql中是不行的，在HQL中内部解决了这个问题
范围运算

1.[not] in 列表

```
form Commdity c where  c. age  in(20,40);
form Commdity c where  c. age   not in(20,40);
```

2.[not] between 值1 and 值2

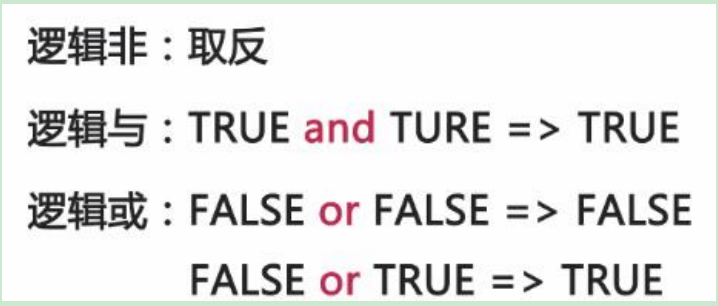
```
form Commdity c where  c. age  between 20 and 40;
form Commdity c where  c. age  not  between 20 and 40;
```

字符串模式匹配
1 like关键字
2 通配符 %_
%通配符匹配任意个字符
_:通配符匹配一个字符

```
form Commdity c where  c. name like '张_';

form Commdity c where  c. address like '%北京%";
```

逻辑运算
1 and(逻辑与) or(逻辑非)
2 not(逻辑非)




```
form Commdity c where  c. price between 100 and 5000 and c.category like '%电脑%";
 form Commdity c where  c. price between 100 and 5000 or c.category like '%电脑%";
```

集合运算

在HQL中存在一对多的属性映射配置
可以通过集合运算符进行相关的判定运算
1 is[not] empty 集合【不】为空，不包含任何元素
2member of 元素属于集合


```
 from Order o where  o.orderitems is empty  //为空的集合
 from Order o where  o.orderitems is not empty  //不为空的集合
```

四则运算
1 HQL语句中也可以使用+ - x /四则运算
2 四则运算可以在where子句和select子句中使用

```
form Commdity c where  c. price*5>3000;
```

查询单个对象
1 Query接口的uniqueResult方法

```
 String hql="from Customer c wherer c.name="张三";
 Query query=session.createQuery(h1l);
Customer c=(Customer)query.uniqueResult();
 System.out.println(c.getName());
```

2 where子句条件的设置
uniqueResult必须保证查询的结果只有一个或没有结果 不能多个结果

ORDER BY 子句

使用order by子句对查询结果排序

如果我们不做然后排序规则设置的话，就是默认设置升序排序

```
form Commdity order by  price asc;
form Commdity order by  price desc;

form Commdity order by  seller.id asc ,price desc,name asc; (
注意用","隔开
)
```

1 HQL语句形式
2 **HQL语句大小敏感，特别是持久化类及其属性的大小写（sql是不注意大小写 ）**
HQL关键字不关心大小写 习惯小写
3 别名的使用，方便HQL的编写
4 select子句中使用自定义类返回选择属性，持久化类构造器处理
一定要在持久化类中加默认构造器 这样不会出现错误

这里最后有个题目
答案大概就是这么个写法（自己纯手写 没经过实践）
select s.name ,s.price , c.name,c.catogoy from seller s ,customer c where catogoy="%书%" and price>=10 order by c.name asc ,price desc ,s.name asc;

select c.name,c.data,c.staus,s.mony from customer c where data in(2015-05-01 ,2015-06-01),staus in('已发货','已付款') ,mony>1000 order by staus asc ,data desc.mony desc;

第一次 复习

```
package com.Hibernate.Dao;

import com.Hibernate.Dao.entity.Account;
import com.Hibernate.Dao.entity.Address;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;

import java.util.List;
import java.util.Map;
import java.util.Queue;


/**
 * Created by han on 2016/8/25.
 */
public class HqlTest {
static Session session;

public HqlTest( ) {

    }

@BeforeClass
public  static void  BeforeClassTEST() {

session =hibernateDao.getSession();
System.out.println("beforeClass 启动");
    }
```

```
@Test
public  void testAccount(){
String hql="   from Account a where a.id in(0,2)";

Query query = session.createQuery(hql);

List<Account> accountList=query.list();

for (Object account : accountList) {

System.out.println(account);
        }

    }

@Test
public  void testAccount2() {
String hql = " select  new map (a.id as id ,a.name as name)  from Account a";

Query query = session.createQuery(hql);

List<Map> maps = query.list();

for (Map map : maps) {

System.out.println("id" + map.get("id"));
System.out.println("name" + map.get("name"));
        }
    }

@Test
public  void testAccount3(){
String hql="   select distinct  new Account(a.id,a.name) from Account  ";

Query query = session.createQuery(hql);

List<Account> accounts=query.list();

for (Account account : accounts) {

System.out.println("id" + account.getId());
System.out.println("name" + account.getName());
            }

        }

@Test
public  void testAccount4(){
String hql="   from Account a where a.id between  0 and 2";

Query query = session.createQuery(hql);

List<Account> accountList=query.list();

for (Object account : accountList) {

System.out.println(account);
        }

    }

@Test
public  void testAccount5(){
String hql="   from Address a where a.aid=1 and a.postcode like '中_'and a.id is not empty and a.id >0";

Query query = session.createQuery(hql);

List<Address> accountList=query.list();

for (Object address : accountList) {

System.out.println(address);
        }
```

```
    }

@Test
public  void testAccount6(){

String hql="from Address where  aid=1";
Query query= session.createQuery(hql);
Address address = (Address) query.uniqueResult();
System.out.println(address);

    }

@Test
public  void testAccount7(){

String hql="from Address order by aid asc ";
Query query= session.createQuery(hql);
List<Address> addresses=query.list();
for (Address address : addresses) {

System.out.println(address);
        }


    }


@AfterClass
public  static  void AfterClassTest(){
System.out.println("Afterclass启动");

    }
}
```