Hibernate的检索方式（查询的方式，抓取方式）

明确：

以后实际开发中实现功能的手段。

共有5中检索方式：

OID查询：

根据ID查询一个实体

涉及的方法：

get(Class clazz,Serializable id)：

永远都是立即加载，返回的是实体类对象

load(Class clazz,Serializable id)：

默认情况下是延迟加载，返回的是实体类对象的代理对象。可以通过配置改为立即加载

配置方式：

修改的位置：查哪个表在哪个表映射的实体类映射文件　class标签。

添加的属性：lazy

取值：true延迟加载(默认值) false立即加载

对象导航查询：根据任意条件查询出主表或者从表的获取对象再通过对象get对象的方法获取别一个表的数据

明确：

只有有关联关系的实体，才可以使用对象导航查询。

对象导航查询是需要有思想的转变。

用法：

使用对象.对象的方式。

例如：customer.getLinkMans()

linkman.getCustomer()

HQL查询

就是使用HQL语句查询

HQL：Hibernate Query Language

写法：就是把表的名称换成实体类的名称，把表中列的名称换成实体类中属性（get/set方法后面的部分）的名称。

其他的和结构化查询语言(sql)一样。

涉及的对象：Query对象

获取的方法：Session对象的createQuery(String hql);

hibernate的官方推荐的查询方式：

QBC查询

Query By Criteria　它是一种更加面向对象的查询方式。

它把查询的条件和结构都封装成了方法。

明确：凡是用HQL语句能实现了,QBC都能实现。反之亦然。

通常情况下，我们用QBC查询，一般都是用在多条件查询上。（离线查询：放到CRM案例的第二天讲解）

涉及的对象：Criteria对象。（离线对象：DetachedCriteria）

获取的方法：Session的createCriteria(Class clazz)。

参数的含义：要查询的实体类字节码

原生SQL语句查询

要求：

查询客户名称为　修正药业　下的所有联系人

涉及了两种表：cst_customer　　cst_linkman

/*子查询语句*/　把查主表的结果做为查子表的条件

```
select * from cst_linkman where lkm_cust_id in (select cust_id from cst_customer where cust_name = '修正药业');
```

/*表连接语句*/ 同时查主从表 条件：查主表的ID等于从表的外键 并且 主表的 名称字段＝修正药业

```
select l.* from cst_customer c ,cst_linkman l where c.cust_id = l.lkm_cust_id and c.cust_name = '修正药业';
```

Hibernate的检索策略（查询的策略，抓取策略）

明确：

是对我们查询方式的优化。让我们能更合理的执行查询。

检索策略分为两部分：

类级别的检索策略：

解决的问题：

什么时候真正发起查询

涉及的方法：

get：立即加载

load：默认是延迟加载，可以通过配置来改为立即加载。

查询的内容：

当前实体中包含的数据。

当前查询实体所对应的表中包含的数据。

关联级别的检索策略：

解决的问题：

什么时候真正的发起查询（查询的是当前实体中的关联对象）

立即加载，延迟加载

采用什么方式去查询

多条语句，子查询，表连接

分为的情况：

有少的一方，查询多的一方

一对多，多对多

有多的一方，查询少的一方

多对一，一对一

使用关联级别的检索策略：

实际开发中只有两种能配置：

有少的要不要查多的：

配置就是：延迟加载，多条语句

有多的要不要查少的：

配置就是：立即加载，多条语句

涉及的配置：

一对多，多对多：

lazy：是否是延迟加载。

取值：

true：延迟加载 默认值

false：立即加载

extra：极懒加载　　用什么数据查询什么数据，不用的都不查。

fetch：采用何种方式加载。

取值：

select：多条语句 默认值

subselect：子查询

join：表连接。当配置了它之后，lazy属性就失效了。

多对一，一对一：

lazy：是否是延迟加载。

取值：

false：立即加载

proxy：看关联对象的类级别检索策略。

no-proxy：我们不用管

fetch：采用何种方式加载。

取值：

select：多条语句

join：表连接

实际开发中的配置：

一对多，多对多：

lazy="true"

fetch="select"

多对一，一对一：

lazy="false"

fetch="select"