提问：

1. Hibernate与ORM 关系

2. Hibenate开发步骤、执行流程

3. Hibernate 中方言作用

4. 映射文件起什么作用？

5. 如何进行联合主键映射？

<composite-id>

执行流程

1. 创建Configuration对象

2. 加载主配置文件:hibernate.cfg.xml

3. 接着，创建SessionFactory对象

4. 创建Session

5. 开启事务

6. – 执行操作 –

7. 提交/关闭

目标：

1. hibernate对象状态

2. hibenate 一级缓存

3. 集合映射/多对一/一对多映射

# 1. hibernate对象状态

session.save(对象);

1. 临时状态

1 . 直接new出来的对象

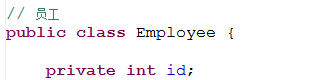
2. 不处于session的管理(即没有与任何一个session关联)

3. 对象在数据库中没有对应的记录！

Employee e = new Employee();

e.setId(2); 只要主键在数据库中存在，就说这个对象在数据库中有对应记录！

 在数据库中叫主键！

OID, object identified 对象的唯一标识 （对应数据库中的主键）

1. 持久化状态

1. 处于session的管理范围

当执行session的方法如：save/update/saveOrUpdate/get/load

对象就会自动转变为持久化状态！

2. 在数据库中有对应的记录

3. 处于持久化状态的对象，当对对象属性进行更改的时候，提交事务更改会反映到数据库中！

1. 游离状态

1. 对象不处于session的管理范围； 通常指session关闭后对象的状态

2. 对象在数据库中有对应的记录

# 2. Session缓存

## 概念:

Session缓存，也叫做一级缓存！

当执行session的相关方法，如: save()/update()/get()/load()等方法的时候，对象会自动放入一级缓存中。

当Session关闭后，一级缓存内容失效！

特点：

1）缓存有效范围，只在当前session范围内有效！ 缓存时间很短、作用范围小！

2）一级缓存，可以在短时间内多次操作数据库的时候，才会明显提升效率！

一级缓存的结构：Ma<主键，对象>

3）在提交事务时候，

Hibernate会同步缓存数据到数据库中，会对比缓存数据与数据库数据是否一致，如果不一致，才提交更改到数据库(生成update)!

4) hibernate提供的一级缓存有hibernate自身维护，如果想操作一级缓存内容，必须通过hibernate提供的方法；

session.flush(); 手动让让一级缓存内容与数据库同步

session.evict(emp1); 清空一级缓存中对象: 清除指定的对象

session.clear(); 清空一级缓存中对象: 清除所有的对象

## list()与iterator()查询区别

list: (通常使用频率较高)

Query接口定义的list查询方法，

一次查询所有满足需要的数据！

iterator:

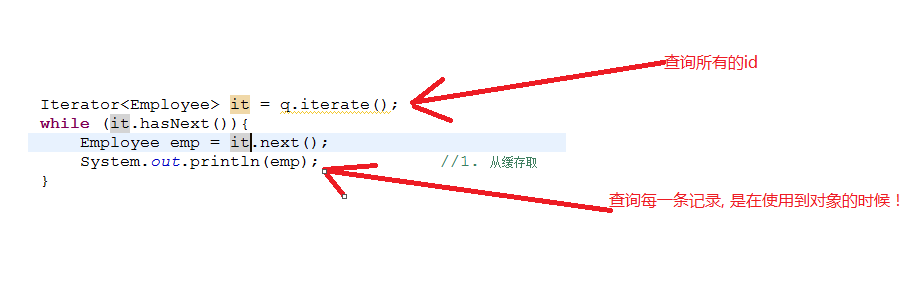
Query接口定义的iterator查询方法，

1. 先查询所有满足条件记录的主键 （查询1次）

2. 再根据每一个id，进行主键查询，有多少记录，查询多少次 （查询n次）

3. iterator查询： N + 1

4. iterator查询，迭代数据的时候，只有用到数据的时候，才会查找数据库(懒加载)



区别：

1. 查询数据方式不同： 查询全部与查询N+1

2. 一级缓存

List查询，查询的结果会放入一级缓存，但不会从一级缓存中获取！

Iteraotro查询，会放入一级缓存，同时也会从一级缓存中获取！

# 3. lazy 属性

Hibernate为了提升程序运行效率，提供了懒加载！

lazy 属性表示懒加载！

true 支持懒加载

false 关闭懒加载

extra (集合属性时候使用)

懒加载： **用到数据的时候，才向数据库发送查询的sql！**

(1) 主键查询：

get/load

get:

及时加载, 只要get就立刻查询数据库

如果查询的主键不存在，返回null

Load:

懒加载， 只有在用到数据的时候，才向数据库发送sql语句！

如果查询的主键不存在，只要使用就报错！

(2) 懒加载作用位置

类级别，默认支持懒加载，但只有在使用load使用才会应用这个特性！

字段级别， 普通字段无效，大数据类型会有影响(long/longtext)

集合属性， 默认支持懒加载

(3) 懒加载异常

org.hibernate.LazyInitializationException: could not initialize proxy - no Session

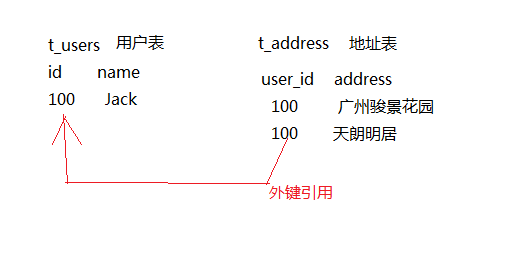
解决思路：

在关闭后，不能能懒加载数据， 那么就要求关闭前把懒加载使用的数据先查询出来！

# 4.集合映射

1. 需求：购物填写地址，一个用户，对应多个地址！

2. 数据库设计：



3. 设计javabean封装数据

**4. 写映射**

5. App.java 测试

思考：

集合映射，如果保存的数据只有一个可以使使用List/Set集合！

如果保存的数据有2个，使用Map集合

如果保存的数据是2个以上，

集合元素类型，能否为对象？

# 5.多对一、一对多映射

需求： 保存用户信息，用户对应的多个地址！

周颖，年龄25, 广州天河，gzth，510000

欧阳光，年龄25, 广州天河，gzth，510000

用户与地址的关系：

一对多 【一个用户对应多个地址】

地址与用户的关系

多对一 【多个地址对应一个用户】

数据库设计:

Javabean封装数据库数据：

映射:

App:

## 多对一， 实现

目的：

通过多对一映射，能完全描述地址表、用户表的数据库的关系！

|  |
| --- |
| **public** **class** Address {    **private** **int** id;  **private** String name;  **private** String shortName;  **private** String code;    // 地址与用户， 是多对一的关系  **private** Users user;  …  } |
| 多对一映射：  <!--  多对一的配置:  1.映射的对象  2.对象对应的外键字段  3.对象的类型  注意：  对象一定是有映射文件进行映射！ (cn.itcast.d\_many2one.Users 对应有映射文件)  -->  <many-to-one name=*"user"* column=*"user\_id"* class=*"Users"*></many-to-one> |
|  |
|  |

## 一对多， 实现

|  |
| --- |
| **public** **class** Users {  **private** **int** id;  **private** String name;  **private** **int** age;  // 用户与地址，是一对多的关系 【注意一定要用接口接收！】  **private** Set<Address> address = **new** HashSet<Address>();  } |
| 一对多映射：  <set name=*"address"* table=*"t\_address"*>  <key column=*"user\_id"*></key>  <one-to-many class=*"Address"*/>  </set> |
|  |
|  |

## 总结：

一对多与多对一映射，

可以只配置多对一， 只能通过的多的一方维护关系!

可以只配置一对多，  只能通过一的一方维护关系

双向配置： 一对多与多对一, 可以通过双向维护关系！

作业：

一对多与多对一练习，需求自拟！

-🡪 用户与地址

🡪 用户与订单

🡪 老师与学员

🡪 部门/ 员工