回顾Hibernate第二天：

1. 一对多与多对一

2. 多对多

3. inverse/cascade

4. 关联关系的维护

|  |
| --- |
| 一对多:  <set name="映射的集合属性" table="(可选)集合属性对应的外键表">  <key column="外键表的，外键字段" />  <one-to-many class="集合元素的类型" />  </set>  多对一：  <many-to-one name="对象属性" class="对象类型" column="外键字段字段" />  多对多  <set name="" table="">  <key column="" />  <many-to-many column="" class="">  </set> |

目标：

第1部分: 对象的状态：

第2部分：缓存

1) 一级缓存

2) 相关知识

----懒加载---

第3部分：映射

一对一映射

组件映射

继承映射

# 一、对象的状态

举例： User user = new User();

Hibernate中对象的状态： 临时/瞬时状态、持久化状态、游离状态。

* **临时状态**

**特点：**

**直接new出来的对象;**

**不处于session的管理;**

**数据库中没有对象的记录;**

* **持久化状态**

**当调用session的save/saveOrUpdate/get/load/list等方法的时候，对象就是持久化状态。**

**处于持久化状态的对象，当对对象属性进行更改的时候，会反映到数据库中!**

**特点：**

**处于session的管理;**

**数据库中有对应的记录;**

* **游离状态**

**特点**

不处于session的管理；

数据库中有对应的记录

Session关闭后，对象的状态；

对象状态的转换，

# 二、一级缓存

为什么要用缓存？

目的：减少对数据库的访问次数！从而提升hibernate的执行效率！

Hibernate中缓存分类：

**一级缓存**

二级缓存

* 概念

**1）Hibenate中一级缓存，也叫做session的缓存，它可以在session范围内减少数据库的访问次数！ 只在session范围有效！ Session关闭，一级缓存失效！**

**2）当调用session的save/saveOrUpdate/get/load/list/iterator方法的时候，都会把对象放入session的缓存中。**

**3）Session的缓存由hibernate维护， 用户不能操作缓存内容； 如果想操作缓存内容，必须通过hibernate提供的evit/clear方法操作。**

**特点：**

**只在(当前)session范围有效，作用时间短，效果不是特别明显！**

**在短时间内多次操作数据库，效果比较明显！**

* 缓存相关几个方法的作用

session.flush(); 让一级缓存与数据库同步

session.evict(arg0); 清空一级缓存中指定的对象

session.clear(); 清空一级缓存中缓存的所有对象

在什么情况用上面方法？

批量操作使用使用：

Session.flush(); // 先与数据库同步

Session.clear(); // 再清空一级缓存内容

* 面试题1： 不同的session是否会共享缓存数据?

**不会。**

**User1 u1 = Session1.get(User.class,1); 把u1对象放入session1的缓存**

**Session2.**update(u1); 把u1放入session2的缓存

U1.setName(‘new Name’);

如果生成2条update sql， 说明不同的session使用不同的缓存区，不能共享。

* 面试题2： list与iterator查询的区别？ yite wei te

list()

一次把所有的记录都查询出来，

**会放入缓存，但不会从缓存中获取数据**

Iterator

N+1查询； N表示所有的记录总数

即会先发送一条语句查询所有记录的主键（1），

再根据每一个主键再去数据库查询（N）！

**会放入缓存，也会从缓存中取数据！**

# 三、懒加载

* 面试题3： get、load方法区别？

get: 及时加载，只要调用get方法立刻向数据库查询

load:默认使用懒加载，当用到数据的时候才向数据库查询。

* 懒加载：(lazy)

概念：当用到数据的时候才向数据库查询，这就是hibernate的懒加载特性。

目的：提高程序执行效率！

* lazy 值

true 使用懒加载

false 关闭懒加载

extra (在集合数据懒加载时候提升效率)

在真正使用数据的时候才向数据库发送查询的sql；

如果调用集合的size()/isEmpty()方法，只是统计，不真正查询数据！

* 懒加载异常
  + Session关闭后，不能使用懒加载数据！
  + 如果session关闭后，使用懒加载数据报错：

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

如何解决session关闭后不能使用懒加载数据的问题？

// 方式1： 先使用一下数据

//dept.getDeptName();

// 方式2：强迫代理对象初始化

Hibernate.initialize(dept);

// 方式3：关闭懒加载

设置lazy=false;

**// 方式4： 在使用数据之后，再关闭session！**

# 四、一对一映射

需求: 用户与身份证信息

一条用户记录对应一条身份证信息！ 一对一的关系！

设计数据库：

JavaBean：

映射：

## 基于外键的映射

|  |
| --- |
| // 身份证  **public** **class** IdCard {  // 身份证号(主键)  **private** String cardNum;// 对象唯一表示(Object Identified, OID)  **private** String place; // 身份证地址  // 身份证与用户，一对一的关系  **private** User user; |
| // 用户  **public** **class** User {  **private** **int** userId;  **private** String userName;  // 用户与身份证信息， 一对一关系  **private** IdCard idCard; |
|  |
| <?xml version=*"1.0"*?>  <!DOCTYPE hibernate-mapping PUBLIC  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">  <hibernate-mapping package=*"cn.itcast.c\_one2one"*>    <class name=*"IdCard"* table=*"t\_IdCard"*>  <id name=*"cardNum"*>  <generator class=*"assigned"*></generator>  </id>  <property name=*"place"* length=*"20"*></property>    <!--  一对一映射，有外键方  unique="true" 给外键字段添加唯一约束  -->  <many-to-one name=*"user"* unique=*"true"* column=*"user\_id"* class=*"User"* cascade=*"save-update"*></many-to-one>    </class>    </hibernate-mapping> |
|  |
| <?xml version=*"1.0"*?>  <!DOCTYPE hibernate-mapping PUBLIC  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">  <hibernate-mapping package=*"cn.itcast.c\_one2one"*>    <class name=*"User"* table=*"t\_user"*>  <id name=*"userId"*>  <generator class=*"native"*></generator>  </id>  <property name=*"userName"* length=*"20"*></property>  <!--  一对一映射： 没有外键方  -->  <one-to-one name=*"idCard"* class=*"IdCard"*></one-to-one>    </class>    </hibernate-mapping> |

## 基于主键的映射

|  |
| --- |
| // 身份证  **public** **class** IdCard {  **private** **int** user\_id;  // 身份证号  **private** String cardNum;  **private** String place; // 身份证地址  // 身份证与用户，一对一的关系  **private** User user; |
|  |
| <?xml version=*"1.0"*?>  <!DOCTYPE hibernate-mapping PUBLIC  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">  <hibernate-mapping package=*"cn.itcast.c\_one2one2"*>    <class name=*"IdCard"* table=*"t\_IdCard"*>  <id name=*"user\_id"*>  <!--  id 节点指定的是主键映射, 即user\_id是主键  主键生成方式： foreign 即把别的表的主键作为当前表的主键；  property (关键字不能修改)指定引用的对象 对象的全名 cn..User、 对象映射 cn.User.hbm.xml、 table(id)  -->  <generator class=*"foreign"*>  <param name=*"property"*>user</param>  </generator>  </id>  <property name=*"cardNum"* length=*"20"*></property>  <property name=*"place"* length=*"20"*></property>    <!--  一对一映射，有外键方  （基于主键的映射）  constrained="true" 指定在主键上添加外键约束  -->  <one-to-one name=*"user"* class=*"User"* constrained=*"true"* cascade=*"save-update"*></one-to-one>    </class>    </hibernate-mapping> |

# 五、组件映射与继承映射

类的关系

组合关系

一个类中包含了另外一个类。这2个类中就是组合关系。

需求： 汽车与车轮

继承关系

一个类继承另外一个类。这2个类中就是继承关系。

需求：动物

猫

猴子

## 组件映射

类组合关系的映射，也叫做组件映射！

注意：组件类和被包含的组件类，共同映射到一张表！

需求： 汽车与车轮

数据库

T\_car

主键 汽车名称 轮子大小 个数

Javabean：

|  |
| --- |
| **public** **class** Car {  **private** **int** id;  **private** String name;  // 车轮  **private** Wheel wheel;  } |
| // 车轮  **public** **class** Wheel {  **private** **int** count;  **private** **int** size;  } |
| <hibernate-mapping package=*"cn.itcast.d\_component"*>    <class name=*"Car"* table=*"t\_car"*>  <id name=*"id"*>  <generator class=*"native"*></generator>  </id>  <property name=*"name"* length=*"20"*></property>    <!-- 组件映射 -->  <component name=*"wheel"*>  <property name=*"size"*></property>  <property name=*"count"*></property>  </component>      </class>    </hibernate-mapping> |
|  |

## 继承映射

需求：动物

猫

猴子

### 简单继承映射

|  |
| --- |
| // 动物类  **public** **abstract** **class** Animal {  **private** **int** id;  **private** String name; |
| <!--  简单继承  -->  <hibernate-mapping package=*"cn.itcast.e\_extends1"*>    <class name=*"Cat"* table=*"t\_Cat"*>  <!-- 简单继承映射： 父类属性直接写 -->  <id name=*"id"*>  <generator class=*"native"*></generator>  </id>  <property name=*"na"*></property>  <property name=*"catchMouse"*></property>  </class>    </hibernate-mapping> |
|  |
| @Test  **public** **void** getSave() {    Session session = *sf*.openSession();  session.beginTransaction();    // 保存  // Cat cat = new Cat();  // cat.setName("大花猫");  // cat.setCatchMouse("抓小老鼠");  // session.save(cat);    // 获取时候注意：当写hql查询的使用，通过父类查询必须写上类的全名  Query q = session.createQuery("from cn.itcast.e\_extends1.Animal");  List<Animal> list = q.list();  System.*out*.println(list);    session.getTransaction().commit();  session.close();    } |
|  |

总结：

简单继承映射，有多少个子类，写多少个映射文件！

### 继承映射

需求：猫、猴子、动物。

#### 所有子类映射到一张表 (1张表)

什么情况用？

子类教多，且子类较为简单，即只有个别属性！

好处：因为使用一个映射文件， 减少了映射文件的个数。

缺点：（不符合数据库设计原则）

一个映射文件： Animal.hbm.xml

(如何区分是哪个子类的信息？)

数据库:

T\_animal （要存储所有的子类信息） “鉴别器”

Id name catchMouse eatBanana type\_(区别是哪个子类)

1 大马猴 NULL 吃10个香蕉 猴子

2 大花猫 不抓老鼠 NULL 猫

总结：

写法较为简单：所有子类用一个映射文件，且映射到一张表！

但数据库设计不合理!

（不推荐用。）

#### 每个类映射一张表(3张表)

数据库

T\_anmal （存储父类信息）

1 大花猫

T\_cat （引用父类的主键）

1 抓小老鼠

T\_monkey（引用父类的主键）

|  |
| --- |
| Javabean设计一样，映射实现不同： |
| <!--  继承映射， 每个类对应一张表(父类也对应表)  -->  <hibernate-mapping package=*"cn.itcast.e\_extends3"*>    <class name=*"Animal"* table=*"t\_animal"*>  <id name=*"id"*>  <generator class=*"native"*></generator>  </id>  <property name=*"name"*></property>    <!--  子类：猫 t\_cat  key 指定\_cat表的外键字段  -->  <joined-subclass name=*"Cat"* table=*"t\_cat"*>  <key column=*"t\_animal\_id"*></key>  <property name=*"catchMouse"*></property>  </joined-subclass>    <!-- 子类：猴子 t\_monkey -->  <joined-subclass name=*"Monkey"* table=*"t\_monkey"*>  <key column=*"t\_animal\_id"*></key>  <property name=*"eatBanana"*></property>  </joined-subclass>    </class>    </hibernate-mapping> |

总结：

一个映射文件，存储所有的子类； 子类父类都对应表；

缺点：表结构比较负责，插入一条子类信息，需要用2条sql： 往父类插入、往子类插入！

#### (推荐)每个子类映射一张表， 父类不对应表(2张表)

数据库：

T\_cat

Id name catchMounse

T\_monkey

Id name eatBanana

|  |
| --- |
| <!--  继承映射， 每个类对应一张表(父类不对应表)  -->  <hibernate-mapping package=*"cn.itcast.e\_extends4"*>  <!--  abstract="true" 指定实体类对象不对应表，即在数据库段不生成表  -->  <class name=*"Animal"* abstract=*"true"*>  <!-- 如果用union-subclass节点，主键生成策略不能为自增长！ -->  <id name=*"id"*>  <generator class=*"uuid"*></generator>  </id>  <property name=*"name"*></property>    <!--  子类：猫 t\_cat  union-subclass  table 指定为表名, 表的主键即为id列  -->  <union-subclass name=*"Cat"* table=*"t\_cat"*>  <property name=*"catchMouse"*></property>  </union-subclass>    <!-- 子类：猴子 t\_monkey -->  <union-subclass name=*"Monkey"* table=*"t\_monkey"*>  <property name=*"eatBanana"*></property>  </union-subclass>    </class>    </hibernate-mapping> |

总结：

**所有的子类都写到一个映射文件；**

父类不对应表； 每个子类对应一张表

Hibernate中映射：

多对一

一对多

多对多

一对一 （多对一的特殊应用）

组件

继承