Hibernate联合主键三种方式

1. 建主键类

将需要两个属性成为联合主键包装在同一类中也是UserPK主键类

**主键类：（UserPK）**

@Embeddable//**在主键加@Embeddable**

**public** **class** UserPK {

**private** **int** sid;

**private** **int** cid;

**public** **int** getSid() {

**return** sid;

}

**public** **void** setSid(**int** sid) {

**this**.sid = sid;

}

**public** **int** getCid() {

**return** cid;

}

**public** **void** setCid(**int** cid) {

**this**.cid = cid;

}

}

**实体类：(User)**

@Entity

@Table(name="t\_user2")

**public** **class** User {

**private** UserPK pk;

**private** String name;

@Id

@GeneratedValue

**public** UserPK getPk() {

**return** pk;

}

**public** **void** setPk(UserPK pk) {

**this**.pk = pk;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

**生成的表：**

create table t\_user2 (

cid integer not null,

sid integer not null,

name varchar(255),

primary key (cid, sid)

)

联合主键用Hibernate注解映射方式主要有三种：  
第一、将联合主键的字段单独放在一个类中，该类需要实现java.io.Serializable接口并重写equals和hascode，再将该类注解为@Embeddable,最后在主类中(该类不包含联合主键类中的字段)保存该联合主键类的一个引用，并生成set和get方法，并将该引用注解为@Id

第二、将联合主键的字段单独放在一个类中，该类需要实现java.io.Serializable接口并重写equals和hascode，最后在主类中(该类不包含联合主键类中的字段)保存该联合主键类的一个引用，并生成set和get方法，并将该引用注解为@EmbeddedId

第三、将联合主键的字段单独放在一个类中，该类需要实现java.io.Serializable接口并要重写equals和hashcode.最后在主类中(该类包含联合主键类中的字段)将联合主键字段都注解为@Id,并在该类上方将上这样的注解：@IdClass(联合主键类.class)

转自：<http://blog.sina.com.cn/s/blog_8297f0d00101012c.html>

使用方式：

1、model类：

1. @Entity
2. @Table(name="JLEE01")
3. **public** **class** Jlee01 **implements** Serializable{
5. **private** **static** **final** **long** serialVersionUID = 3524215936351012384L;
6. **private** String address ;
7. **private** **int** age ;
8. **private** String email ;
9. **private** String phone ;
10. @Id
11. **private** JleeKey01 jleeKey ;

主键类：JleeKey01.java

1. @Embeddable
2. **public** **class** JleeKey01  **implements** Serializable{
4. **private** **static** **final** **long** serialVersionUID = -3304319243957837925L;
5. **private** **long** id ;
6. **private** String name ;
7. /\*\*
8. \* @return the id
9. \*/
10. **public** **long** getId() {
11. **return** id;
12. }
13. /\*\*
14. \* @param id the id to set
15. \*/
16. **public** **void** setId(**long** id) {
17. **this**.id = id;
18. }
19. /\*\*
20. \* @return the name
21. \*/
22. **public** String getName() {
23. **return** name;
24. }
25. /\*\*
26. \* @param name the name to set
27. \*/
28. **public** **void** setName(String name) {
29. **this**.name = name;
30. }
32. @Override
33. **public** **boolean** equals(Object o) {
34. **if**(o **instanceof** JleeKey01){
35. JleeKey01 key = (JleeKey01)o ;
36. **if**(**this**.id == key.getId() && **this**.name.equals(key.getName())){
37. **return** **true** ;
38. }
39. }
40. **return** **false** ;
41. }
43. @Override
44. **public** **int** hashCode() {
45. **return** **this**.name.hashCode();
46. }
48. }

2、model类：

1. @Entity
2. @Table(name="JLEE02")
3. **public** **class** Jlee02 {
5. **private** String address ;
6. **private** **int** age ;
7. **private** String email ;
8. **private** String phone ;
9. @EmbeddedId
10. **private** JleeKey02 jleeKey ;

主键类：JleeKey02.java

普通java类即可。

3、model类：

1. @Entity
2. @Table(name="JLEE03")
3. @IdClass(JleeKey03.**class**)
4. **public** **class** Jlee03 {
5. @Id
6. **private** **long** id ;
7. @Id
8. **private** String name ;

主键类：JleeKey03.java

普通java类。