hibernate 注解

JPA：java持久化接口，hibernate接口JPA,才能使用注解

注解作用：1. @Override 表示重写方法

[2.@SuppressWarnings("unchecked")](mailto:2.@SuppressWarnings(%22unchecked%22))

当list没用泛型，会报出unchecked错误

List list = new ArrayList(); List.add("1111");

注: Test.java使用了未经检查或不安全的操作。

注: 有关详细信息, 请使用 -Xlint:unchecked 重新编译

3.

所以1，判断是否重写

2，忽略安全警告

自定义注解：新建other-java-Annotation

**@Retention(RetentionPolicy.*RUNTIME*)//指定运行期**

**public** **@interface** A {}

判断是否加注解得不得到：

@A

**public** **class** Test {

**public** **static** **void** main(String[] args) {

A a = Test.**class**.getAnnotation(A.**class**);

**if**(a == **null**){

System.*out*.println("not fores");

}**else**{ } }}

机器语言

JVM

Class文件

Java代码

元注解：注解中的注解

元消息：消息中的消息

1. @Retention(RetentionPolicy.*RUNTIME*)

自定义注解是在编译期后就不会运行了，所以要加上面的语句，就在解释器中运行

2.@Target(ElementType.CONSTRUCTOR) FIELD *METHOD 规定注解放在那*

构造方法 变量 方法

@Retention(RetentionPolicy.*RUNTIME*)

**public** **@interface** A {

**public** **int** value();

} //有且只有一个方法，名为value(),在使用注解的时候才会不需要等于号

@A(14) @A(id=4)

注意：**在用注解时候我们获取SessionFactory 是 只能通过 SessionFactory factory = new AnnotationConfiguration().configure().buildSessionFactory();**

使用JPA文档：

1. 导入jap中包
2. 注解：

**import** java.util.Set;

**import** javax.persistence.\*;

@Entity

@Table(name="t\_user")

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

**private** **int** id;

**private** String name;

**private** Group group;

@ManyToOne()

@JoinColumn(name="gid")

**public** Group getGroup() {

**return** group;

}

**public** **void** setGroup(Group group) {

**this**.group = group;

}

@Id

@GeneratedValue//默认为native

//这是主键模式，由于一开始加载对象，所以get就是得到对象，set就是插入对象

//所以放在get方法上是最好的

// @Column(name="id")

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

@Column(name="username",length=12,unique=**true**,nullable=**false**)

**public** String getName() {

**return** name;

}

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

**this**.name = name;} }

**双向：**

@OneToMany(mappedBy="group")

**public** Set<User> getUsers() {

**return** users;

}

可以知道OneToOne其实就和OneToMany一样类似，当然uiqune=true

OneToMany与OneToOne中 mappedBy=“映射关系实体名”

这个只需要加载数据，不需要创建新的表，所以：

多对多

@ManyToMany

@JoinTable(name="t\_user\_group",joinColumns={@JoinColumn(name="gid")}

,inverseJoinColumns={@JoinColumn(name="uid")} )

**public** Set<Group> getGroup() {

**return** group;

}

**而且采用JoinTable映射，会出现联合主键，即双主键**。

**@JoinTable** 注解可定义关联表。该关联表包含指向实体的外键（通过@JoinTable.joinColumns）以及指向目标实体表的外键（[通过@JoinTable.inverseJoinColumns](mailto:通过@JoinTable.inverseJoinColumns)）.

@ManyToMany

@JoinTable(name="两张表的联系表",joinColumns={@JoinColumn(name="对应实体的外键")} ,inverseJoinColumns={@JoinColumn(name="实体的外键")} )

+many 创建表，

@OneToMany @JoinTable(name="t\_types",joinColumns={@JoinColumn(name="tid")},inverseJoinColumns={@JoinColumn(name="tid")})

这类才不会创建表

+one 创建一列

**@transient** 当不保存数据时，用其就不保存包数据库中去，其中的意思就是数据不会再数据库中增加

14、@Embedded  
  
可选  
@Embedded将几个字段组合成一个类,并作为整个Entity的一个属性.  
例如User包括id,name,city,street,zip属性.  
我们希望city,street,zip属性映射为Address对象.这样,User对象将具有id,name和  
address这三个属性.  
Address对象必须定义为@Embededable   
示例:  
@Embeddable  
public class Address {city,street,zip}  
@Entity  
public class User {

@Embedded  
public Address getAddress() {  
..........  
}  
}

@Transient  
    可选  
    @Transient表示该属性并非一个到数据库表的字段的映射,ORM框架将忽略该属性.  
    如果一个属性并非数据库表的字段映射,就务必将其标示为@Transient,否则,ORM框架默认其注解为@Basic  
    示例:  
    //根据birth计算出age属性  
    @Transient  
    public int getAge() {  
       return getYear(new Date()) - getYear(birth);  
    }

1. 怎么使用自增加并且不是从一开始
2. 怎么实现成绩表中字段

  @Column描述了数据库表中该字段的详细定义,这对于根据JPA注解生成数据库表结构的工具非常有作用.   
    name:表示数据库表中该字段的名称,默认情形属性名称一致   
    nullable:表示该字段是否允许为null,默认为true   
    unique:表示该字段是否是唯一标识,默认为false   
    length:表示该字段的大小,**仅对String类型的字段有效**   **insertable**:表示在ORM框架执行插入操作时,该字段是否应出现INSETRT语句中,默认为true，也就是可以为这个字段赋值，若想用预先设定好的值，那么可以设置insertable=false   
    **updateable:**表示在ORM框架执行更新操作时,该字段是否应该出现在UPDATE语句中,默认为true.对于一经创建就不可以更改的字段,该属性非常有用,如对于birthday字段，我们规定它创建后不给修改，那就可以设置这个属性的updateable=false.   
    **columnDefinition:**表示该字段在数据库中的实际类型.通常ORM框架可以根据属性类型自动判断数据库中字段的类型,但是对于Date类型仍无法确定数据库中字段类型究竟是DATE,TIME还是TIMESTAMP.此外,String的默认映射类型为VARCHAR,如果要将String类型映射到特定数据库的BLOB或TEXT字段类型,该属性非常有用.

**开始我认为能生成联合主键方法**

**private** Set<Group> group;

@ManyToMany @JoinTable(name="t\_user\_group",joinColumns={@JoinColumn(name="gid")}

,inverseJoinColumns={@JoinColumn(name="uid")} )

**public** Set<Group> getGroup() {

**return** group;

}

**private** Set<User> users;

**@ManyToMany @JoinTable(name="t\_user\_group",joinColumns={@JoinColumn(name="uid")}**

**,inverseJoinColumns={@JoinColumn(name="gid")} )**

**public Set<User> getUsers() {**

**return users;**

**}**

生成的表为：

create table t\_user (

id integer not null auto\_increment,

username varchar(12) not null unique,

primary key (id)

)

**create table t\_user\_group (**

**uid integer not null,**

**gid integer not null,**

**primary key (gid, uid)**

**)**

alter table t\_user\_group

add index FK300645B6FC2415A9 (uid),

add constraint FK300645B6FC2415A9

foreign key (uid)

references t\_group (id)

alter table t\_user\_group

add index FK300645B6AD535D73 (gid),

add constraint FK300645B6AD535D73

foreign key (gid)

references t\_user (id)

**1.如果把上述的Set改为List，则不会生成联合主键**

**由于set和list区别在于，set不可以重复，而list可以，list生成联合主键会导致不唯一，所以不会生成联合主键**

**2.多对多，中创建第三方的表时，要使用@JoinTable(name=”第三方表名”，joinColumns={@JoinColumn(name=”关于此表外键”),inverJoinColimns={@JoinColumn(name=”该表与多对多关系外键”)})**

No identifyer specified for entity:

缺少主键

1. @Column(name = "stat\_date")
2. @Temporal(TemporalType.DATE)
3. **private** Date statDate;

@Id

@ManyToMany

@JoinTable(name="t\_student",joinColumns={@JoinColumn(name="sno,cno")}

,inverseJoinColumns={@JoinColumn(name="sno,cno")} )

**private** Set<Student> students;