Hibernate注解与JPA

JPA注解是JAVAEE的规范和标准，JPA是标准接口，hibernate是实现，但是其功能是JPA的超集。一般在实际开发中,优先考虑使用JPA注解，这样更有利于程序的移植和扩展。 Hibernate如何实现与JPA的关系

通过hibernate-annotation, hibernate-entitymanager, hibernate-core这三个组件来实现。

使用注解的目的：为了简化繁琐的ORM文件(\*.hbm)的配置

# 类级别注解

## @Entity 注解

@Entity(name="tableName"):映射实体类，name可选，对应数据库中的一个表，若表名与实体类名相同，则name可省略。

注意，使用@Entity时必须指定实体类的主键属性。

## @Table 注解

@Table(name="",catalog="", schema=""):与@Entity配合使用，只能标注在实体的class定义处，表示实体对应的数据库表的信息。Name可选，映射表的名称，若表名与实体类名相同，则name可省略。catalog可选，表示catalog名称，默认为Catalog("")。Schema可选，表示schema名称，默认为schma("")。

从实现的角度看，各种数据库系统对catalog和schema的支持和实现方式各不一样。

|  |  |  |
| --- | --- | --- |
| **数据库** | **catalog** | **schema** |
| oracle | 不支持 | Oracle User ID |
| mysql | 不支持 | 数据库名 |
| SQL Server | 数据库名 | 对象属主名，2005ban开始有变 |

## @Embeddable注解

@Embeddable表示一个非Entity类可以嵌入到另一个Entity类中作为属性而存在。

# 属性级别注解

添加方式：一、可以在属性字段上添加；二、可以在属性的get访问器的上面

## @id注解

必须的，定义了映射到数据库表的主键的属性，一个实体类可以有一个或者多个属性被映射为主键，可置于主键属性或者getXxxx()前。

注意：如果有多个属性定义为主键属性，该实体类必须实现serializable接口。

## @GeneratedValue注解

@GeneratedValue(strategy=GenerationType,generator=""):可选，用于定义主键生成策略。

Strategy表示主键生成策略，取值有：

1. GenerationType.AUTO:根据底层数据库自动选择（默认）
2. GenerationType.INDENTITY:根据数据库的Identity字段生成
3. GenerationType.SEQUENCE:使用Sequence来决定主键的取值
4. GenerationType.TABLE:使用指定表来决定主 键取值，结合@TableGenerator使用

|  |
| --- |
| @TableGenerator(name="tab\_cat\_gen", allocationSize=1)  @GeneratedValue(Strategy=GenerationType.Table)  Generator-表示主键生成器的名称，这个属性通常和ORM框架相关，如Hibernate可以指定uuid等主键生成方式  将字符串类型的变量设成主键使用下面的注解：  @Id  @GeneratedValue(generator="sid")  @GenericGenerator(name="sid", strategy="assigned") //Hibernate注解  @Column(length=8) |

## @Column注解

可将属性映射到列，使用该注解来覆盖默认值，@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。 常用属性：

* name:可选，表示数据库表中该字段的名称，默认情形属性名称一致。
* nullable：可选，表示该字段是否允许为null,默认为true。
* unique:可选，表示该字段是否是唯一标识，默认为false。
* length:可选，表示该字段的大小，仅对String类型的字段有效，默认值255。
* inserable:可选，表示在ORM框架执行插入操作时，该字段是否应出现INSERT语句中，默认为true。
* updateable:可选，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，改属性非常有用，如对于birthday字段。

## @Embedded注解

@Embedded是注释属性的，表示该属性的类时嵌入类。

注意：同时嵌入类也必须标注@Embeddable注解。

## @EmbeddedId注解

@EmbeddedId使用嵌入式主键类实现复合主键。嵌入式主键类必须实现Serializable接口，必须有默认的public无参的构造方法、必须覆盖equals和hashCode方法。

## @Transient注解

可选，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，就务必将其标识为@Transient,否则ORM框架默认其注解为@Basic。

## @Temporal注解

显示日期：@Temporal(TemporalType.DATE)

显示时间：@Temporal(TemporalType.TIME)

显示日期时间：@Temporal(TemporalType.TIMESTAMP)

# 关系映射注解

实体之间的映射关系

1.一对一：一个公民对应一个身份证号码。

2.一对多（多对一）：一个公民有多个银行账号。

3.多对多：一个学生有多个老师，一个老师教多个学生。

## 一对一关系映射注解

|  |
| --- |
| #一对一单向外键  @OneToOne(cascade=CascadeType.ALL)  @JoinColumn(name="pid",unique=true)  注：保存时应该先保存外键对象，再保存主表对象。  这里写图片描述 |
| #一对一双向外键  主控方的配置同一对一单向外键关联  被控方在主控方的引用上加上注解：@OneToOne(mappedBy="card")  双向关联，必须设置mappedBy属性。因为双向关联关系只能交给一方去控制，不可能在双方都设置外键保存关联关系，否则双方都无法保存。 |

## 一对多关联映射注解

|  |
| --- |
| #多对一单向外键  @ManyToOne(cascade={CasecadeType.All},fetch=FetchType.EAGER)  @JoinColum(name="cid", referencedColumnName="CID")  这里写图片描述 |
| #一对多单向外键，一方持有多方集合（一对多）  @OneToMany(cascade={CasecadeType.All},fetch=FetchType.LAZY)  @JoinColum(name="cid")  抓取策略：多对一时，多方设置为EAGER；一对多时，一方设置为LAZY。  这里写图片描述 |
| #一对多（多对一）双向外键  **多方持有一方的引用**  @ManyToOne(cascade={CasecadeType.All},fetch=FetchType.EAGER)  @JoinColum(name="cid")  **一方持有多方的集合**  @OneToMany(cascade={CasecadeType.All},fetch=FetchType.LAZY)  @JoinColum(name="cid")  保存时无论先保存哪一方都可以。 |

## 多对多关联映射注解

|  |
| --- |
| #多对多单向外键  学生和教师构成多对多的关联关系。（先保存教师再保存学生）  其中一个多方持有另一个多方的集合对象（这里让学生持有教师的集合）  创建中间表  这里写图片描述 |
| #多对多双向外键  双方持有对方的集合对象，其中一方设置  @ManyToMany(mappedBy="teachers")  这里写图片描述 |

# hibernate的级联、抓取策略、inverse属性

## hibernate JPA级联

hibernate的级联是用来说明数据库中二个表之间的相互关系（one---to ---one, one ---to---many, many---to---many)。当对主对象做某种操作时，是否对其相关联的子对象也做相对应的操作。常见的级联（cascade）有：

* none: 表示在保存、修改、删除时不对其级联对象不做任何的操作，它是默认值。
* save-update:表示在保存和更新时，级联保存、更新附属对象。
* delete:表示在删除当前对象时，级联删除附属对象。
* all:表示所有情况下均进行级联操作，比如：save-update, delete操作等。
* delete--orphan: 表示删除当前对象的同时，删除与当前对象解除关系的孤儿对象。（仅限于用在一对多上面。）

JPA中常用值：CascadeType.ALL， CascadeType.MERGE，CascadeType.PERSIST，CascadeType.REMOVE，CascadeType.REFRESH

注意：级联一般使用在父表中（如果子表建立了关联关系，就只要保存子表而不需要保存字表）。如果不使用级联关系的话。在插入数据的时候，我们应该先保存父对象，在保存子对象。

## hibernate JPA延迟加载

在jpa中jpa默认的加载方式是lazy方式也就是在实际使用到数据的时候才加 载相关数据，使用lazy时可以不用显示注明fetch=FetchType.LAZY

fetch=FetchType.LAZY：延迟加载，只有真正获取数据时才发出SQL语句。

fetch=FetchType.LAZY：立即获取。

## hibernate inverse属性

hibernate的inverse属性：Inverse是双向关系中的基本概念，当然对于多数实体我们并不需要双向关联，更多的是选择单项关联。inverse的使用或设置不当会导致性能的下降，既会产生多余的sql，甚至会有jdbc异常产生。例如：在双向关联当中，如果双放inverse都设置为true,那么双方将都不会维护关系的更新，如果双方inverse都设置为false的话，必然后导致双方维护同一个关系。所以双向关联中，最好一端inverse=true,一端inverse=fase。一般情况下会把inverse=fase放在many端（inverse=fase方去维护关系）。

注解使用mappedBy("")：//mappedBy在哪一端，哪一端就不维护关系，它成为了关系的被管理端 。//向当于hibernate中的inverse=true 。

注意：一般mappedBy使用在主表方（表示有子表(多的一方来维护关系)）,使用了mappedBy注解就不能使用@JoinColumn(由子表来时使用，表中name指向自己的外键列)，@JoinTable。另外，使用mapperBy,关联关系的维护只能由另一方来维护(set集合)，使用mapperBy来维护关联是不生效的。