## CascadeType（级联操作定义）

1. PERSIST：级联新增
2. MERGE：级联新增和更新
3. REMOVE：级联删除
4. REFRESH：级联刷新。假设场景---有一个订单，订单里面关联了许多商品，这个订单可以被很多人操作。这个时候A对订单和关联的商品进行了修改，B也进行了相同的操作，但是B先A进行保存了这些数据，那么当A保存数据时，就需要先刷新订单信息及关联的商品信息，再将订单及商品保存。
5. DETACH（从2.0版本开始加入）：级联托管/游离操作。如果要删除一个实体，但是它有外键无法删除，就需要这个级联权限。它会撤销所有相关的外键关联。
6. ALL：拥有以上所有级联操作权限。

## **FetchType**

定义从数据库获取数据的策略，有两个值：EAGER立马加载策略，LAZY懒加载策略。

**值介绍：**

1. EAGER：取出这条数据时，它关联的数据也同时取出放入内存中。

2、LAZY：取出这条数据时，它关联的数据并不取出来，在同一个session中，什么时候要用，就什么时候取(再次访问数据库)。但是，在session外，就不能再取了。用EAGER时，因为在内存里，所以在session外也可以取。

## @ManyToMany

**属性介绍：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **属性** | **默认值** | **说明** |
| Class | targetEntity | void.class | 关联目标的实体类，默认为当前关联字段/属性的类型。 |
| CascadeType[] | cascade | {} | 级联操作 |
| FetchType | fetchType | EAGER | 从数据库加载数据策略。默认是EAGER（马上加载） |
| boolean | optional | true | 定义关联关系是否可选的。如果设置为false则表示非空关系必须存在，即n:1时，1的一端必须有数据和n端对应 |

## **@OneToMany**

**属性介绍：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **属性** | **默认值** | **说明** |
| Class | targetEntity | void.class | 关联目标的实体类，默认为当前关联字段/属性的类型。 |
| CascadeType[] | cascade | {} | 级联操作。当关联的目标是java.util.Map时，cascade属性值将应用到map的value值上 |
| FetchType | fetchType | LAZY | 从数据库加载数据策略。默认是LAZY |
| String | mappedBy | “” | 双休关联，指定关系维护方 |
| boolean | orphanRemoval | false | 从2.0版本开始加入。是否将删除应用到级联的对象。即删除1端数据时，是否同时删除关联的n端数据（是否主从删除） |

## **@OneToMany和@ManyToMany例子**

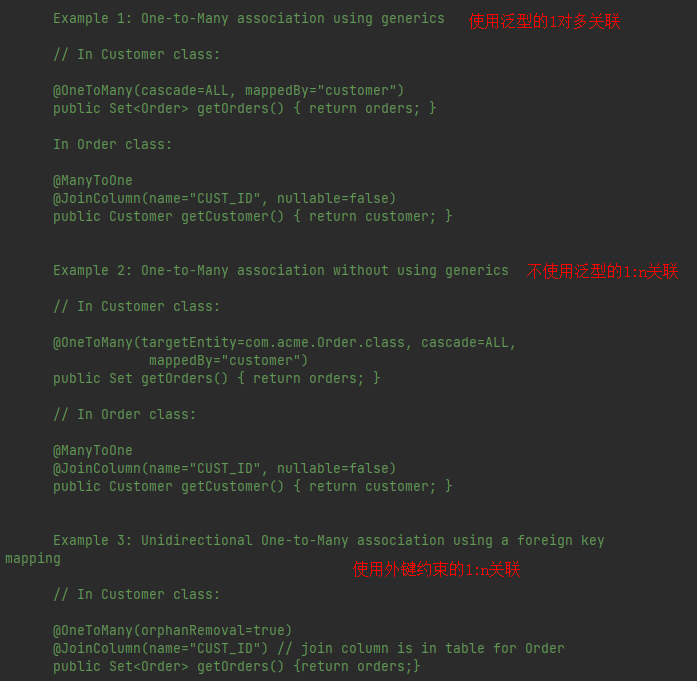


图1-1

注意：Example 2是1对多双向关联，关联关系是交给Order中customer属性来管理。

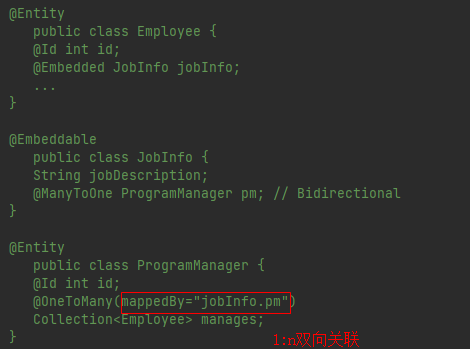


图1-2

## **@MapKey**

[用于@OneToMany时，目标类型为java.util.Map](mailto:用于@OneToMany时，目标类型为java.util.Map)。如果主键以外的持久化字段或属性被用作映射键，那么期望它具有与之相关联的唯一性约束。

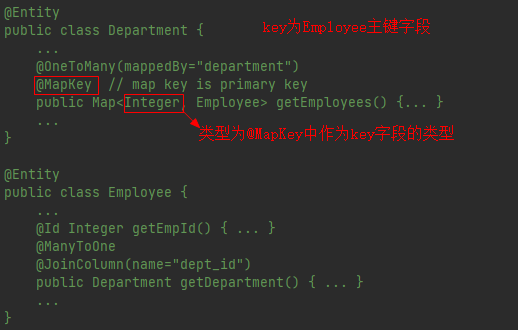


图1-1

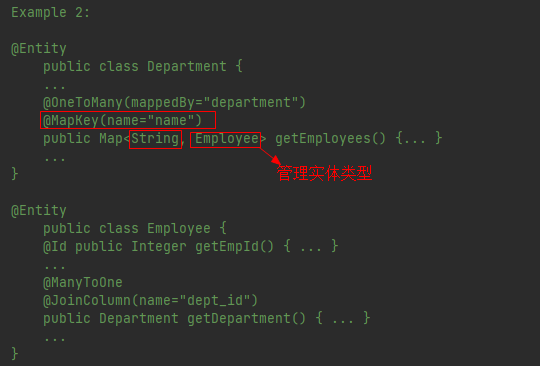


图1-2

使用例子：<https://www.cnblogs.com/xiluhua/p/4385545.html>

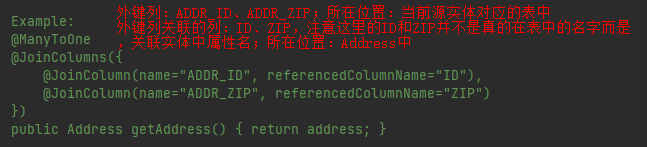
## **@JoinColumn**

**属性介绍：**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **属性** | **默认值** | **说明** |
| String | name | “”;在默认情况下（仅单个连接列时）值为，关联表的名称+“\_”+ 关联表主键的字段名 | 外键列的名称，分4中情况：   1. 如果连接是使用外键映射策略的OneToOne或ManyToOne映射，则外键列在源实体的表中或可嵌入的表中。如：，外键列ADDR\_ID就在当前实体关联的表中。 2. 如果连接是使用外键映射策略的单向OneToMany映射，则外键在目标实体的表中。 3. 如果连接是ManyToMany映射或OneToOne或双向ManyToOne/OneToMany映射，则外键在连接表中。 4. 如果连接是针对元素集合的，则外键在集合表中。如： |
| String | referencedColumnName | “”，默认为被引用实体关联表的主键列 | 外键列引用的列的名称。默认(仅在使用单个连接列时适用):与被引用表的主键列同名。分4种情况：   1. 当与实体关系映射一起使用(这里描述的情况除外)时，引用的列位于目标实体的表中 2. 当与**单向的OneToMany**外键映射一起使用时，引用的列在源实体的表中。 3. 当在JoinTable注释中使用时，引用的列位于所属实体的实体表中，如果连接是逆连接定义的一部分，则位于逆实体中。 4. 当在CollectionTable映射中使用时，引用的列位于包含集合的实体的表中 |
| boolean | unique | false | 属性是否唯一。对于与作为外键一部分的主键相对应的连接列（即外键关联的列时主键），没有必要显式地指定此值 |
| boolean | nullable | true | 外键列是否为空 |
| boolean | insertable | true | 该列是否包含在持久性提供程序生成的SQL INSERT语句中 |
| boolean | updatable | true | 该列是否包含在持久性提供程序生成的SQL UPDATE语句中 |
| String | Table | “” | 外键列所在的表名。分4种情况：   1. 如果连接是使用外键映射策略的OneToOne或ManyToOne映射，则表名为源实体的表名或可嵌入表名。 2. 如果连接是使用外键映射策略的单向OneToMany映射，则为目标实体的表名 3. 如果连接是ManyToMany映射或OneToOne或双向ManyToOne/OneToMany映射，则表名为连接表 4. 如果连接是一个元素集合，则为集合表泛型类型对应的表的表名 |
| ForeignKey | foreignKey | ConstraintMode  .PROVIDER\_DEFAULT | 从2.1版本开始加入这个属性。用于在表生成生效时指定或控制外键约束的生成。如果未指定此元素，则数据库中默认外键策略 |

解决JPA2中@ManyToMany设置了ConstraintMode.NO\_CONSTRAINT依旧产生外键的坑：https://blog.csdn.net/ohskean/article/details/89841347（未尝试过）

## **@JoinColumns**



## **双向关联和单向关联**

**定义区别**

**单向关联：**

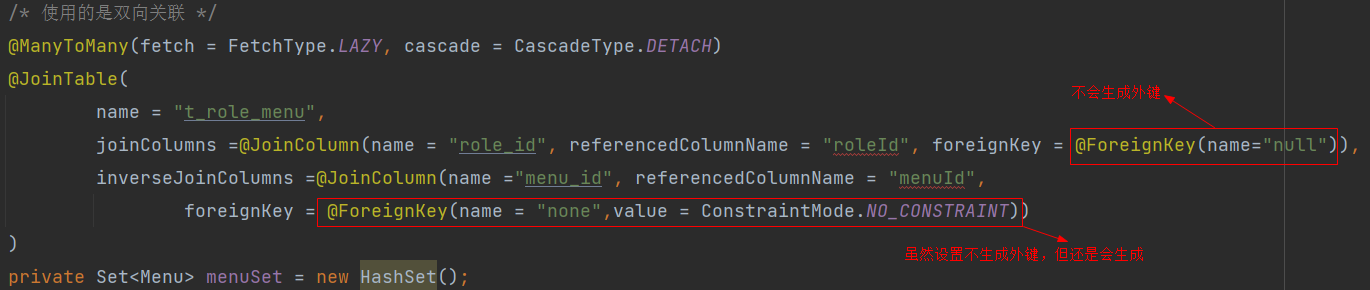
1. @OneToMany/@ManyToOne：在1方使用@OneToMany和@JoinColumn注解。
2. @ManyToMany：只在其中的一方使用@ManyToMany/@JoinTable注解。

**双向关联：**

1. @OneToMany/@ManyToOne：在1方使用@OneToMany（并定义mappedBy属性的值）和@JoinColumn注解，多的一方使用@ManyToOne注解。
2. @ManyToMany：在其中的一方使用@ManyToMany/@JoinTable注解，另一方使用@ManyToMany并且定义mappedBy属性的值。

## **设置数据库不生成外键关系**

**@ManyToMany**



之所以role\_id列不会生成外键，是因为设置了外键约束的名字为null，而null在数据库中是不允许作为外键约束名的，导致sql语句报错所以不会生成外键约束。生成外键约束的sql语句如下图-外键约束生成1、图-外键约束生成2，所示：

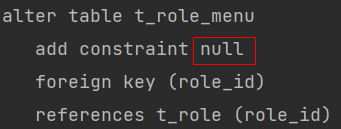


图-外键约束生成1

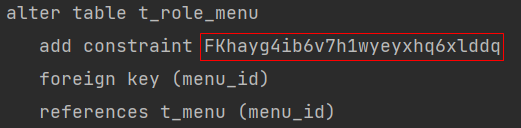


图-外键约束生成2

关于@ForeignKey给value属性设置了值ConstraintMode.NO\_CONSTRAINT，还是会生成外键约束的原因猜想：value属性的介绍中说了“如果没有指定foreignKeyDefinition元素，提供程序将生成一个约束，它认为该约束的更新和删除操作最适合应用外键注释的连接列。”所以当设置ConstraintMode.NO\_CONSTRAINT相当于没有指定foreignKeyDefinition元素，此时程序又将会生成一个约束。