

# 操作记录

---

## 1.关联说明

---

### 1.一对一（单向）

#### 1.关键实体

```
1  People -> t_people
2
3  idCard -> t_idcard 维护外键
4  @OneToOne(cascade = {CascadeType.PERSIST})
5  @JoinColumn(name = "t_people_id")//外键id
6  private People people;
7
```

### 2.自动生成的表结构

```
1  CREATE TABLE `t_person` (
2    `t_id` int(11) NOT NULL,
3    `t_address` varchar(255) DEFAULT NULL,
4    `t_age` int(11) DEFAULT NULL,
5    `t_birthday` datetime(6) DEFAULT NULL,
6    `t_name` varchar(255) DEFAULT NULL,
7    PRIMARY KEY (`t_id`)
8  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
9
10
11 CREATE TABLE `t_idcard` (
12   `t_id` int(11) NOT NULL AUTO_INCREMENT,
13   `t_number` varchar(255) DEFAULT NULL,
14   `t_people_id` int(11) DEFAULT NULL,
15   PRIMARY KEY (`t_id`),
16   KEY `FK7gdvrysil6gxmt806ysqr8atn` (`t_people_id`),
17   CONSTRAINT `FK7gdvrysil6gxmt806ysqr8atn` FOREIGN KEY (`t_people_id`)
18   REFERENCES `t_people` (`t_id`)
19 ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

### 2.一对一（双向）

#### 1.关键实体

```

1  People2 -> t_people2
2  @OneToOne(mappedBy = "people2")//负责维护外键的表对应的实体中，持有的本类类型属性的
   属性名字
3      private IdCard2 idCard2;
4
5
6  idCard2 -> t_idcard2 维护外键
7
8      @OneToOne(cascade = {CascadeType.PERSIST})
9      @JoinColumn(name = "t_people_id")//外键id
10     private People2 people2;

```

## 2.自动生成的表结构

```

1  CREATE TABLE `t_people2` (
2      `t_id` int(11) NOT NULL AUTO_INCREMENT,
3      `t_age` int(11) DEFAULT NULL,
4      `t_name` varchar(255) DEFAULT NULL,
5      PRIMARY KEY (`t_id`)
6  ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
7
8  CREATE TABLE `t_idcard2` (
9      `t_id` int(11) NOT NULL AUTO_INCREMENT,
10     `t_number` varchar(255) DEFAULT NULL,
11     `t_people_id` int(11) DEFAULT NULL,
12     PRIMARY KEY (`t_id`),
13     KEY `FKrfj903v724ch0q2ex5u6m2min` (`t_people_id`),
14     CONSTRAINT `FKrfj903v724ch0q2ex5u6m2min` FOREIGN KEY (`t_people_id`)
15     REFERENCES `t_people2` (`t_id`)
16 ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

```

## 3.一对多（单向）

### 1.主要实体

```

1  Student -> t_student
2  @Id
3      @Column(name = "t_id")
4      @GeneratedValue(strategy = GenerationType.IDENTITY)
5      private Integer id;
6  -----
7  Project -> t_project
8
9  @ManyToOne()
10     @JoinColumn(name = "t_student_id")
11     private Student student;
12     ### 1
13     加了级联
14     测试一个student
15     多个project，设置同一个student
16     会报错
17

```

```

18 解决方法
19 不用级联
20 先保存student，再保存project
21
22 ##### 2
23 尝试加级联看看怎么才能可以同时插入多条project(student为同一个
24 解决方案
25 需要在test方法上加上下面连个注解
26 作用是使得代码在同一个事务中，同时自动提交
27 没有@Commit，则不会提交会回滚
28 @Transactional
29 @Commit

```

## 2.自动生成的表

```

1  CREATE TABLE `t_student` (
2      `t_id` int(11) NOT NULL AUTO_INCREMENT,
3      `t_name` varchar(255) DEFAULT NULL,
4      PRIMARY KEY (`t_id`)
5  ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
6
7  CREATE TABLE `t_project` (
8      `t_id` int(11) NOT NULL AUTO_INCREMENT,
9      `t_sub_name` varchar(255) DEFAULT NULL,
10     `t_student_id` int(11) DEFAULT NULL,
11     PRIMARY KEY (`t_id`),
12     KEY `FKjbi0sj2aqjxfmj2tki0wv33xd` (`t_student_id`),
13     CONSTRAINT `FKjbi0sj2aqjxfmj2tki0wv33xd` FOREIGN KEY (`t_student_id`)
14     REFERENCES `t_student` (`t_id`)
15 ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;

```

## 4, 一对多 (双向)

### 1.主要实体

```

1  Student2 -> t_student2
2  @Id
3      @Column(name = "t_id")
4      @GeneratedValue(strategy = GenerationType.IDENTITY)
5      private Integer id;
6
7      @Column(name = "t_name")
8      private String name;
9
10     @OneToMany(mappedBy = "student2")//维护外键的一方对应的实体类中的本类类型的属性
11     字段名称
12     private List<Project2> project2;
13
14  Project2 -> t_project2

```

```

15 @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     @Column(name = "t_id")
18     private Integer id;
19
20     @Column(name = "t_sub_name")
21     private String name;
22
23     @ManyToOne()
24     @JoinColumn(name = "t_student_id")
25     private Student2 student2;
26
27

```

## 2.自动生成的表结构

```

1 CREATE TABLE `t_student2` (
2     `t_id` int(11) NOT NULL AUTO_INCREMENT,
3     `t_name` varchar(255) DEFAULT NULL,
4     PRIMARY KEY (`t_id`)
5 ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
6
7 CREATE TABLE `t_project2` (
8     `t_id` int(11) NOT NULL AUTO_INCREMENT,
9     `t_sub_name` varchar(255) DEFAULT NULL,
10    `t_student_id` int(11) DEFAULT NULL,
11    PRIMARY KEY (`t_id`),
12    KEY `FK148vus0ax7ywxcek420rik40` (`t_student_id`),
13    CONSTRAINT `FK148vus0ax7ywxcek420rik40` FOREIGN KEY (`t_student_id`)
14    REFERENCES `t_student2` (`t_id`)
15 ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;

```

## 5.多对多 (单向)

```

1 @ManyToMany
2 作用：用于映射多对多关系
3 属性：
4 cascade: 配置级联操作。
5 fetch: 配置是否采用延迟加载。
6 targetEntity: 配置目标的实体类。映射多对多的时候不用写。
7
8 @JoinTable
9 作用：针对中间表的配置
10 属性：
11 nam: 配置中间表的名称
12 joinColumns: 中间表的外键字段关联当前实体类所对应表的主键字段
13 inverseJoinColumn: 中间表的外键字段关联对方表的主键字段
14
15 @JoinColumn
16 作用：用于定义主键字段和外键字段的对应关系。
17 属性：
18 name: 指定外键字段的名称

```

19 referencedColumnName: 指定引用主表的主键字段名称  
20 unique: 是否唯一。默认值不唯一  
21 nullable: 是否允许为空。默认值允许。  
22 insertable: 是否允许插入。默认值允许。  
23 updatable: 是否允许更新。默认值允许。  
24 columnDefinition: 列的定义信息。

## 1.主要实体

```
1  @Id
2      @Column(name = "t_id")
3      @GeneratedValue(strategy = GenerationType.IDENTITY)
4      private Integer id;
5
6      @Column(name = "t_name")
7      private String name;
8      -----
9
10 @Id
11     @Column(name = "t_id")
12     @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Integer id;
14
15     @Column(name = "t_name")
16     private String name;
17
18     @ManyToMany(cascade = CascadeType.PERSIST)
19     @JoinTable(name="user_role_rel", //中间表的名称
20               //中间表user_role_rel字段关联sys_role表的主键字段role_id
21               joinColumns=
22               {@JoinColumn(name="role_id",referencedColumnName="t_id")},
23               //中间表user_role_rel的字段关联sys_user表的主键user_id
24               inverseJoinColumns=
25               {@JoinColumn(name="user_id",referencedColumnName="t_id")}
26             )
27     private List<SysUser> sysUserList;
```

## 2.自动生成的表结构

```
1  CREATE TABLE `t_sys_user` (
2      `t_id` int(11) NOT NULL AUTO_INCREMENT,
3      `t_name` varchar(255) DEFAULT NULL,
4      PRIMARY KEY (`t_id`)
5  ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;
6
7
8  CREATE TABLE `t_sys_role` (
9      `t_id` int(11) NOT NULL AUTO_INCREMENT,
10     `t_name` varchar(255) DEFAULT NULL,
11     PRIMARY KEY (`t_id`)
```

```

12 ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
13
14
15 CREATE TABLE `user_role_rel` (
16   `role_id` int(11) NOT NULL,
17   `user_id` int(11) NOT NULL,
18   KEY `FKm28g58dhcs5u9asuuww8ui43w` (`user_id`),
19   KEY `FKi2omtqgkldjbgukc3ry5hsdf` (`role_id`),
20   CONSTRAINT `FKi2omtqgkldjbgukc3ry5hsdf` FOREIGN KEY (`role_id`)
   REFERENCES `t_sys_role` (`t_id`),
21   CONSTRAINT `FKm28g58dhcs5u9asuuww8ui43w` FOREIGN KEY (`user_id`)
   REFERENCES `t_sys_user` (`t_id`)
22 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
23
24

```

## 6.多对多 (多向)

### 1.主要的实体类

```

1  @Id
2      @Column(name = "t_id")
3      @GeneratedValue(strategy = GenerationType.IDENTITY)
4      private Integer id;
5
6      @Column(name = "t_name")
7      private String name;
8
9      @ManyToMany(mappedBy = "sysUserList2")//维护外键的表的对应实体中的属性字段名
10     private List<SysRole2> sysRole2;
11
12
13  @Id
14      @Column(name = "t_id")
15      @GeneratedValue(strategy = GenerationType.IDENTITY)
16      private Integer id;
17
18      @Column(name = "t_name")
19      private String name;
20
21      @ManyToMany(cascade = CascadeType.PERSIST)
22      @JoinTable(name="user_role_rel2",//中间表的名称
23                //中间表user_role_rel字段关联sys_role表的主键字段role_id
24                joinColumns=
25                {@JoinColumn(name="role_id",referencedColumnName="t_id")},
26                //中间表user_role_rel的字段关联sys_user表的主键user_id
27                inverseJoinColumns=
28                {@JoinColumn(name="user_id",referencedColumnName="t_id")})
29      )
30      private List<SysUser2> sysUserList2;

```

### 2.自动生成的表结构

```

1 CREATE TABLE `t_sys_user2` (
2   `t_id` int(11) NOT NULL AUTO_INCREMENT,
3   `t_name` varchar(255) DEFAULT NULL,
4   PRIMARY KEY (`t_id`)
5 ) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;
6
7
8 CREATE TABLE `t_sys_role2` (
9   `t_id` int(11) NOT NULL AUTO_INCREMENT,
10  `t_name` varchar(255) DEFAULT NULL,
11  PRIMARY KEY (`t_id`)
12 ) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8;
13
14
15 CREATE TABLE `user_role_rel2` (
16   `role_id` int(11) NOT NULL,
17   `user_id` int(11) NOT NULL,
18   KEY `FK5o1if0v99f02hwcvmityah1j7` (`user_id`),
19   KEY `FKkjme7hdhxcgkbka7kox6r9u1` (`role_id`),
20   CONSTRAINT `FK5o1if0v99f02hwcvmityah1j7` FOREIGN KEY (`user_id`)
21   REFERENCES `t_sys_user2` (`t_id`),
22   CONSTRAINT `FKkjme7hdhxcgkbka7kox6r9u1` FOREIGN KEY (`role_id`)
23   REFERENCES `t_sys_role2` (`t_id`)
24 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

## 2.联合主键

### 1.方式一

```

1 1.通过 主键实体打上注解@Embeddabl
2 表实体中主键属性打上注解 @EmbeddedId
3
4 2.主要的实体
5 @Data
6 @AllArgsConstructor
7 @NoArgsConstructor
8 @Builder
9 @Embeddable
10 public class ComputerPK implements Serializable {
11
12     @Column(name = "t_ip")
13     private String ip;
14
15     @Column(name = "t_owner_id")
16     private String ownerId;
17
18
19 }
20
21
22 @Data
23 @AllArgsConstructor
24 @NoArgsConstructor
25 @Builder
26 @Entity

```

```

27 @Table(name = "t_computer")
28 public class Computer {
29
30     @EmbeddedId
31     private ComputerPK computerPK;
32
33     @Column(name="t_brand_name")
34     private String brandName;
35
36 }
37 3.自动生成的表结构
38 CREATE TABLE `t_computer` (
39   `t_ip` varchar(255) NOT NULL,
40   `t_owner_id` varchar(255) NOT NULL,
41   `t_brand_name` varchar(255) DEFAULT NULL,
42   PRIMARY KEY (`t_ip`,`t_owner_id`)
43 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

## 2.方式二

```

1 1. @IdClass 配合 @Id 和上面的效果差不多，可能根据方法名字操作方便点
2
3 2.主要的实体
4 @Data
5 @AllArgsConstructor
6 @NoArgsConstructor
7 @Builder
8 @Entity
9 @Table(name = "t_computer2")
10 @IdClass(Computer2PK.class)
11 public class Computer2 {
12
13     @Id
14     @Column(name = "t_ip")
15     private String ip;
16
17     @Id
18     @Column(name = "t_owner_id")
19     private String ownerId;
20
21     @Column(name="t_brand_name")
22     private String brandName;
23
24 }
25 -----
26 @Data
27 @AllArgsConstructor
28 @NoArgsConstructor
29 @Builder
30 public class Computer2PK implements Serializable {
31
32     private String ip;
33
34     private String ownerId;
35
36
37 }

```



```

38
39
40
41 3.自动生成的表结构
42
43 CREATE TABLE `t_computer2` (
44     `t_ip` varchar(255) NOT NULL,
45     `t_owner_id` varchar(255) NOT NULL,
46     `t_brand_name` varchar(255) DEFAULT NULL,
47     PRIMARY KEY (`t_ip`,`t_owner_id`)
48 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

## 3.审计

### 1.添加注解

```

1 1.启动类
2
3 @EnableJpaAuditing
4 2.注解类
5 @EntityListeners({AuditingEntityListener.class})

```

### 2.自定义实现接口AuditorAware的类

```

1 @Component
2 public class AuditConfig implements AuditorAware {
3     /**
4      * Returns the current auditor of the application.
5      *
6      * @return the current auditor
7      */
8     @Override
9     public Optional getCurrentAuditor() {
10         return Optional.of("allen");
11     }
12 }
13
14

```

### 3.在实体类字段加上注解

```

1 @CreatedBy : 由谁创建这条记录
2 @LastModifiedBy: 是谁最后更新了这条记录
3 @CreatedDate: 创建时间
4 @LastModifiedDate: 最后更新时间

```

t\_my\_audit @bilibili\_data\_jpa (mySql) - 表

t_id	cre_date	cre_user	modi_date	modi_user	t_name
1	(Null)	(Null)	2022-11-14 23:37:52.785000	allen	jack

## 4.jpa继承

### 1.SINGLE\_TABLE

#### 1.简单说明

- 1 单表继承策略 SINGLE\_TABLE
- 2
- 3 父类实体和子类实体共用一张数据库表，在表中通过一个辨别字段的值来区分不同类别的实体。

#### 2表对应的实体

##### 1.父类

```

1  @Data
2  @AllArgsConstructor
3  @NoArgsConstructor
4  @Builder
5  @Entity
6  @Inheritance(strategy = InheritanceType.SINGLE_TABLE)//继承的策略
7  @Table(name = "WINDOW_FILE")
8  @DiscriminatorColumn(name = "DISCRIMINATOR", discriminatorType =
DiscriminatorType.STRING, length = 30) // 指定辨别字段的类型为String，长度30
9  @DiscriminatorValue("windowFile")//指定辨别的字段值
10 public class WindowFile {
11
12     @Id
13     @GeneratedValue(strategy = GenerationType.AUTO)
14     private Integer id;
15
16     @Basic
17     @Column(name = "NAME")
18     private String name;
19
20     @Basic
21     @Column(name = "TYPE")
22     private String type;
23
24     @Basic
25     @Column(name = "DATE")
26     private Date date;
27

```

## 2.子类1

```

1  @Entity
2  @DiscriminatorValue("Document")
3  @Data
4  @AllArgsConstructor
5  @NoArgsConstructor
6  @Builder
7  public class Document extends WindowFile {
8
9      @Basic
10     @Column(name = "SIZE")
11     private String size;
12
13 }


```

## 3.子类2

```

1  @Entity
2  @DiscriminatorValue("Folder")
3  @Data
4  @AllArgsConstructor
5  @NoArgsConstructor
6  @Builder
7  public class Folder extends WindowFile {
8
9      @Basic
10     @Column(name = "FILE_COUNT")
11     private Integer fileCount;
12
13 }

```

 1669373795085

 1669373811076

## 2.JOINED

### 1.简单说明

- 1 父类实体和子类实体分别对应数据库中不同的表，父类定义的内容为子类们公共的属性，子类实体中定义的内容为扩展的属性。
- 2 实际生成的表结构如下：
- 3
- 4 表： T\_ANIMAL， 字段： ID, COLOR, NAME
- 5
- 6 表： T\_BIRD ， 字段： SPEED, ID(既是外键，也是主键)
- 7
- 8 表： T\_DOG， 字段： LEGS, ID(既是外键，也是主键)

## 2.表对应实体

### 1.父类

```

1  import lombok.AllArgsConstructor;
2  import lombok.Builder;
3  import lombok.Data;
4  import lombok.NoArgsConstructor;
5
6  import javax.persistence.*;
7
8  @Entity
9  @Table(name = "t_animal")
10 @Inheritance(strategy = InheritanceType.JOINED)
11 @DiscriminatorColumn(name = "aaa") // 辨别字段 AAA
12 @AllArgsConstructor
13 @NoArgsConstructor
14 @Builder
15 @Data
16 public class Animal {
17
18     @Id
19     @Column(name = "id")
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     private Integer id;
22
23     @Column(name = "name")
24     private String name;
25
26     @Column(name = "color")
27     private String color;
28 }
29
30

```

## 2.子类1

```

1  import lombok.AllArgsConstructor;
2  import lombok.Data;
3  import lombok.NoArgsConstructor;
4
5  import javax.persistence.Column;
6  import javax.persistence.DiscriminatorValue;
7  import javax.persistence.Entity;
8  import javax.persistence.Table;
9
10 @Entity
11 @Table(name = "t_bird")
12 @DiscriminatorValue("bird")
13 @Data
14 @AllArgsConstructor
15 @NoArgsConstructor
16 public class Bird extends Animal {
17
18     @Column(name = "speed")
19     private String speed;
20
21     @Override
22     public String toString() {
23         return super.toString() + "Bird{" +
24             "speed='" + speed + '\'' +

```

```

25         '}'
26     }
27 }

```

### 3.子类2

```

1  import lombok.AllArgsConstructor;
2  import lombok.Data;
3  import lombok.NoArgsConstructor;
4
5  import javax.persistence.Column;
6  import javax.persistence.DiscriminatorValue;
7  import javax.persistence.Entity;
8  import javax.persistence.Table;
9
10 @Entity
11 @Table(name = "t_dog")
12 @DiscriminatorValue("dog")
13 @Data
14 @AllArgsConstructor
15 @NoArgsConstructor
16 public class Dog extends Animal {
17
18     @Column(name = "legs")
19     private Integer legs;
20
21     @Override
22     public String toString() {
23         return super.toString() + "Dog{" +
24             "legs=" + legs +
25             '}';
26     }
27 }
28

```

## 4.自动生成的表结构

### 1.父表 (公共表)

```

1  CREATE TABLE `t_animal` (
2      `aaa` varchar(31) NOT NULL,
3      `id` int(11) NOT NULL,
4      `color` varchar(255) DEFAULT NULL,
5      `name` varchar(255) DEFAULT NULL,
6      PRIMARY KEY (`id`)
7  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
8

```

### 2.子表1

```
1  
2 CREATE TABLE `t_bird` (  
3   `speed` varchar(255) DEFAULT NULL,  
4   `id` int(11) NOT NULL,  
5   PRIMARY KEY (`id`),  
6   CONSTRAINT `FKky0iakih6f0xm2eqtq3p5s8u7` FOREIGN KEY (`id`) REFERENCES  
   `t_animal` (`id`)  
7 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

### 3.子表2

```
1 CREATE TABLE `t_bird` (  
2   `speed` varchar(255) DEFAULT NULL,  
3   `id` int(11) NOT NULL,  
4   PRIMARY KEY (`id`),  
5   CONSTRAINT `FKky0iakih6f0xm2eqtq3p5s8u7` FOREIGN KEY (`id`) REFERENCES  
   `t_animal` (`id`)  
6 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
7  
8
```