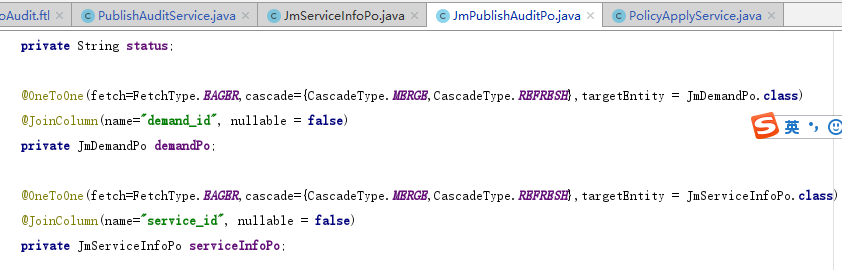
# 单向一一



# 双向一一

## 概念

基于外键的 1-1 关联关系：在双向的一对一关联中，需要在关系被维护端(inverse side)中的 @OneToOne 注释中指定 mappedBy，以指定是这一关联中的被维护端。同时需要在关系维护端(owner side)建立外键列指向关系被维护端的主键列。

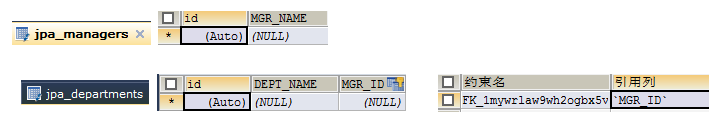
## 配置（建议单向维护）

一个经理只有一个部门，一个部门只有一个经理，部门（外键）



## 数据生成信息

departments表会产生外键



## 测试准备

### Department

@Table(name="JPA\_DEPARTMENTS")

@Entity

**public** **class** Department {

**private** Integer id;

**private** String deptName;

**private** Manager mgr;

@GeneratedValue

@Id

**public** Integer getId() {

**return** id;

}

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

**this**.id = id;

}

@Column(name="DEPT\_NAME")

**public** String getDeptName() {

**return** deptName;

}

**public** **void** setDeptName(String deptName) {

**this**.deptName = deptName;

}

//使用 @OneToOne 来映射 1-1 关联关系。

//若需要在当前数据表中添加主键则需要使用 @JoinColumn 来进行映射.

//注意, 1-1 关联关系, 所以需要添加 unique=true

@JoinColumn(name="MGR\_ID", unique=**true**)

// @OneToOne

@OneToOne(fetch=FetchType.***LAZY***)

**public** Manager getMgr() {

**return** mgr;

}

**public** **void** setMgr(Manager mgr) {

**this**.mgr = mgr;

}

}

### Manager

@Table(name="JPA\_MANAGERS")

@Entity

**public** **class** Manager {

**private** Integer id;

**private** String mgrName;

**private** Department dept;

@GeneratedValue

@Id

**public** Integer getId() {

**return** id;

}

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

**this**.id = id;

}

@Column(name="MGR\_NAME")

**public** String getMgrName() {

**return** mgrName;

}

**public** **void** setMgrName(String mgrName) {

**this**.mgrName = mgrName;

}

//对于不维护关联关系, 没有外键的一方, 使用 @OneToOne 来进行映射, 建议设置 mappedBy="mgr"

@OneToOne(mappedBy="mgr")

// @OneToOne(mappedBy="mgr",fetch=FetchType.LAZY)

**public** Department getDept() {

**return** dept;

}

**public** **void** setDept(Department dept) {

**this**.dept = dept;

}

}

### 准备JPATestOneOne

**public** **class** JPATestOneOne {

**private** EntityManagerFactory entityManagerFactory;

**private** EntityManager entityManager;

**private** EntityTransaction transaction;

@Before

**public** **void** init(){

entityManagerFactory = Persistence.*createEntityManagerFactory*("jpa\_test1");

entityManager = entityManagerFactory.createEntityManager();

transaction = entityManager.getTransaction();

transaction.begin();

}

@After

**public** **void** destroy(){

transaction.commit();

entityManager.close();

entityManagerFactory.close();

}

}

# 操作

## 保存（双向维护）

有4种方式：1.先一再一（外键），先一（外键）再一

2. 主键方放弃维护，维护外键

推荐：先一再一（外键），主键方放弃维护，维护外键

//双向 1-1 的关联关系, 建议先保存不维护关联关系的一方, 即没有外键的一方, 这样不会多出 UPDATE 语句.

@Test

**public** **void** testOneToOnePersistence(){

Manager mgr = **new** Manager();

mgr.setMgrName("M-BB");

Department dept = **new** Department();

dept.setDeptName("D-BB");

//设置关联关系

mgr.setDept(dept);

dept.setMgr(mgr);

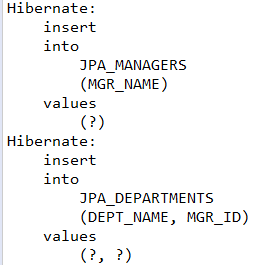
//执行保存操作

entityManager.persist(mgr);

entityManager.persist(dept);

}

使用此种方式不会产生update语句



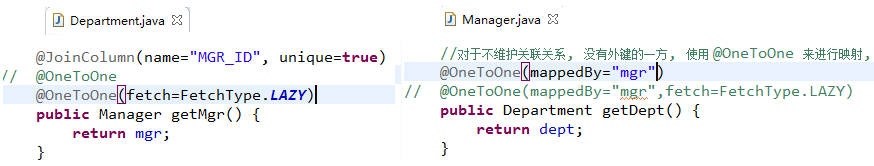
## 查询（oid）

四种方式：1. 获取维护关联关系的一方，发送左外连接

2.获取维护不关联关系的一方，发送左外连接

3. 获取维护关联关系的一方，维护方懒加载，发送单表查询

4. 获取维护不关联关系的一方，不建议懒加载，发送左外连接



### 默认加载，维护

获取维护关联关系的一方：发送左外连接

//1.默认情况下, 若获取维护关联关系的一方, 则会通过左外连接获取其关联的对象.

//但可以通过 @OntToOne 的 fetch 属性来修改加载策略.

@Test

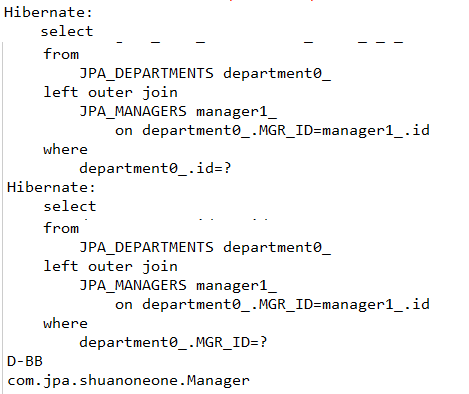
**public** **void** testOneToOneFind(){

Department dept = entityManager.find(Department.**class**, 1);

System.***out***.println(dept.getDeptName());

System.***out***.println(dept.getMgr().getClass().getName());

}



### 默认加载，不维护

获取不维护关联关系的一方：发送左外连接

//1. 默认情况下, 若获取不维护关联关系的一方, 则也会通过左外连接获取其关联的对象.

//可以通过 @OneToOne 的 fetch 属性来修改加载策略. 但依然会再发送 SQL 语句来初始化其关联的对象

//这说明在不维护关联关系的一方, 不建议修改 fetch 属性.

@Test

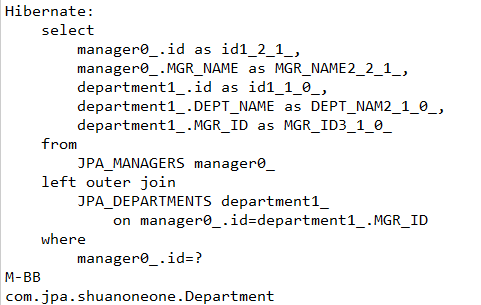
**public** **void** testOneToOneFind2(){

Manager mgr = entityManager.find(Manager.**class**, 1);

System.***out***.println(mgr.getMgrName());

System.***out***.println(mgr.getDept().getClass().getName());

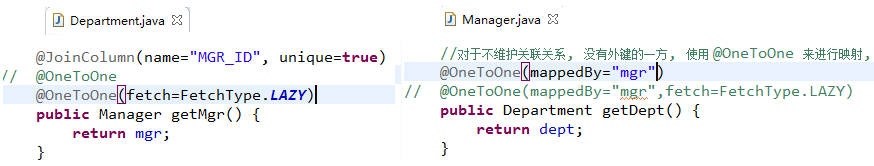
}

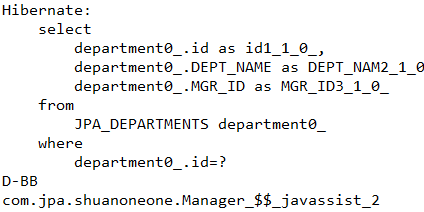


### 懒加载，获取维护关联关系方—单表

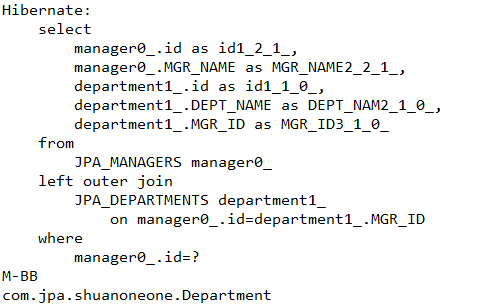
主键方：懒加载，外键方不设置

查询一方；懒加载：当调用时，才发送查询sql语句





### 懒加载，获取不维护关联关系方



## 删除（一方）

### 外键置空，只删一方

### 外键置空，都删

## 更新

# 双向 1-1 不延迟加载的问题

如果延迟加载要起作用, 就必须设置一个代理对象.

Manager 其实可以不关联一个 Department

如果有 Department 关联就设置为代理对象而延迟加载, 如果不存在关联的 Department 就设置 null, 因为外键字段是定义在 Department 表中的,Hibernate 在不读取 Department 表的情况是无法判断是否有关联有 Deparmtment, 因此无法判断设置 null 还是代理对象, 而统一设置为代理对象,也无法满足不关联的情况, 所以无法使用延迟加载,只 有显式读取 Department.

