表设计

1. 多对多

Class M1{

static hasMany = [m2s:M2]

static mapping = {

m2s joinTable: [name: "tb\_m1\_m2", key: 'm1\_id']

}

static belongsTo = [M2]

}

Class M2{

static hasMany = [m1s:M1]

static mapping = {

m1s joinTable: [name: "tb\_m1\_m2", key: 'm2\_id']

}

}

持有的那一方(M2)删除的时候会自动删除中间表的数据,但是被持有的那一方(M1)删除之前必须手动删除中间表的数据.(在M1 domain类中添加下列方法)

def beforeDelete(){

M2.withNewSession {

M2.createCriteria().list{m1s{eq("id",id)}}?.each { M2 m2->

m2.m1s.removeAll {m1->

m1.id==id

}

println m2.m1s

m2.save(flush:true)

}

}

}

1. 一对多

Class M1{

static hasMany = [m2s:M2]

static mapping = {

m2s cascade: 'all-delete-orphan'

}

}

Class M2{

static belongsTo = [m1:M1]

}

持有的那一方(M1)删除的时候会自动删除m2的对象,被持有的那一方(M2)删除之前必须手动删除(在m2中定义如下方法即可)

def beforeDelete(){

M1.withNewSession {

M1.createCriteria().list{m2s{eq("id",id)}}?.each { M1 m1->

m1. m2s.removeAll {m2->

m2.id==id

}

m1.save(flush:true)

}

}

}

1. 一对一

class P{

C c

}

class C{

static belongsTo = [p:P]

}

当C的对象要删除的时候需要先将关联的p里的c属性设置为null ,因此需要修改在控制器的delete

def cDelete(){

def model=[:]

def result=true

params.ids.split(',').grep()?.each{

def c =C.get(it)

if(c){

c.P?c.p.c=null:null

c.delete()

}else{

result=false

}

}

model["result"]=result

render ( model as JSON)

}