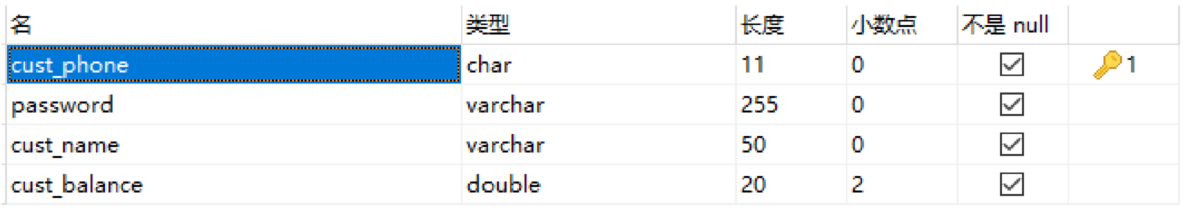
扫码点餐系统接口文档

URL

http://101.132.74.147:8082/

数据库表

customer表(/customer/)



名	类型	长度	小数点	不是 null	
cust_phone	char	11	0	✓	🔑1
password	varchar	255	0	✓	
cust_name	varchar	50	0	✓	
cust_balance	double	20	2	✓	

注：cust_phone为11位字符串；cust_balance为2位小数的双精度浮点数

1.
```
/login
```

接收参数：customer

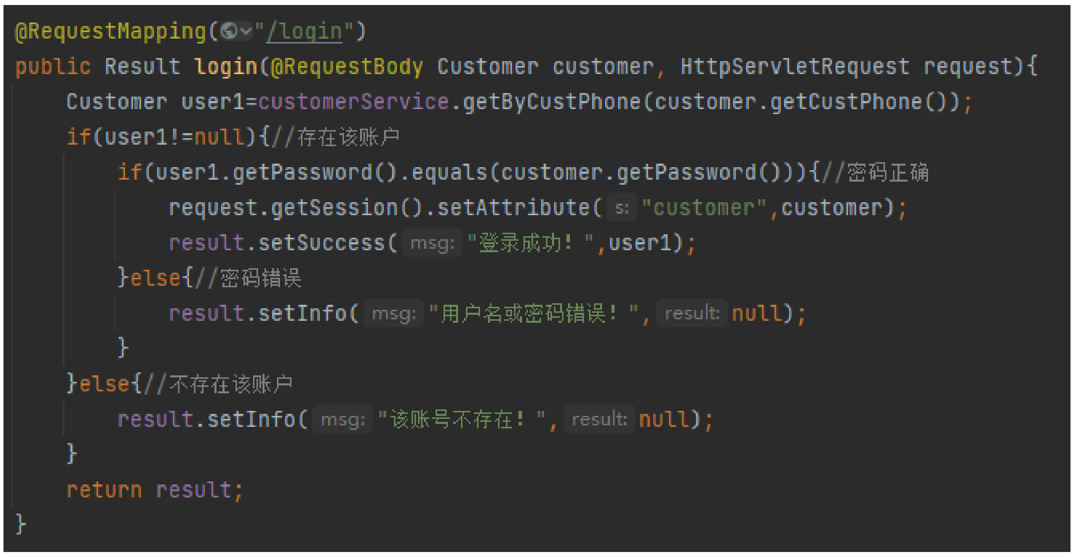
返回参数：customer

服务端逻辑：

1. 根据手机号码查询

```
select * from customer where cust_phone = #{custPhone}
```

2. 查到了匹配密码



```
@RequestMapping("/login")
public Result login(@RequestBody Customer customer, HttpServletRequest request){
    Customer user1=customerService.getByCustPhone(customer.getCustPhone());
    if(user1!=null){//存在该账户
        if(user1.getPassword().equals(customer.getPassword())){//密码正确
            request.getSession().setAttribute( s: "customer",customer);
            result.setSuccess( msg: "登录成功！",user1);
        }else{//密码错误
            result.setInfo( msg: "用户名或密码错误！", result: null);
        }
    }else{//不存在该账户
        result.setInfo( msg: "该账号不存在！", result: null);
    }
    return result;
}
```

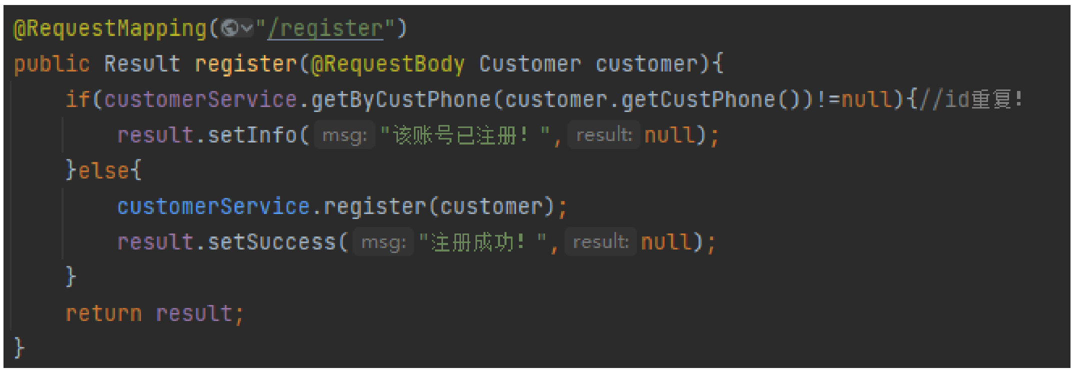
2.
```
/register
```

接收参数：customer

返回参数：null

服务端逻辑：

1. 根据手机号验证账号是否已经注册



```java
@RequestMapping("/register")
public Result register(@RequestBody Customer customer){
    if(customerService.getByCustPhone(customer.getCustPhone())!=null){//id重复!
        result.setInfo( msg: "该账号已注册! ", result: null);
    }else{
        customerService.register(customer);
        result.setSuccess( msg: "注册成功! ", result: null);
    }
    return result;
}
```

2. 然后注册

```sql
insert into customer (cust_phone,cust_name,password) values (#
{custPhone},#{custName},#{password})
```

3.

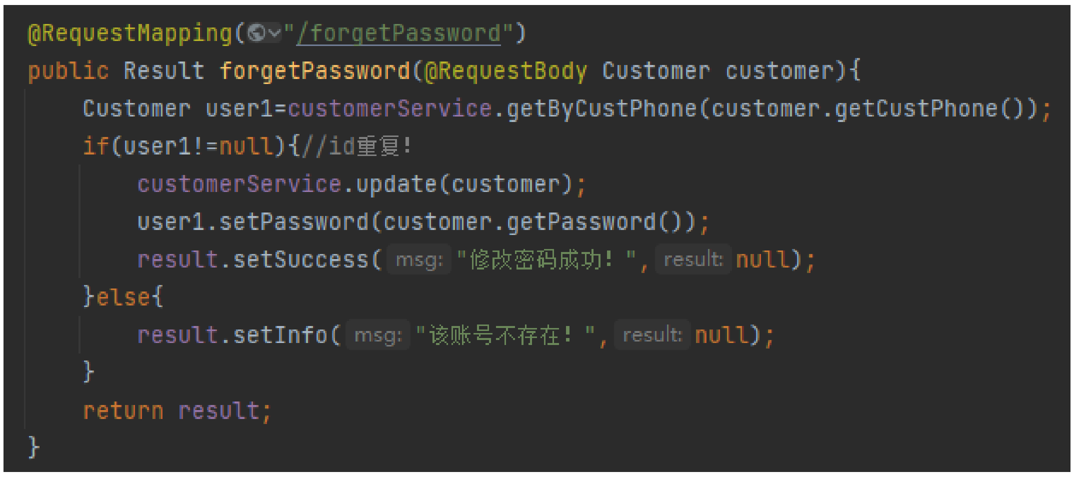
```
/forgetPassword
```

接收参数：customer

返回参数：null

服务端逻辑：

1. 根据手机号查询账号是否存在



```java
@RequestMapping("/forgetPassword")
public Result forgetPassword(@RequestBody Customer customer){
    Customer user1=customerService.getByCustPhone(customer.getCustPhone());
    if(user1!=null){//id重复!
        customerService.update(customer);
        user1.setPassword(customer.getPassword());
        result.setSuccess( msg: "修改密码成功! ", result: null);
    }else{
        result.setInfo( msg: "该账号不存在! ", result: null);
    }
    return result;
}
```

2. 然后修改密码

```sql
update customer set password = #{password} where cust_phone = #
{custPhone}
```

4.

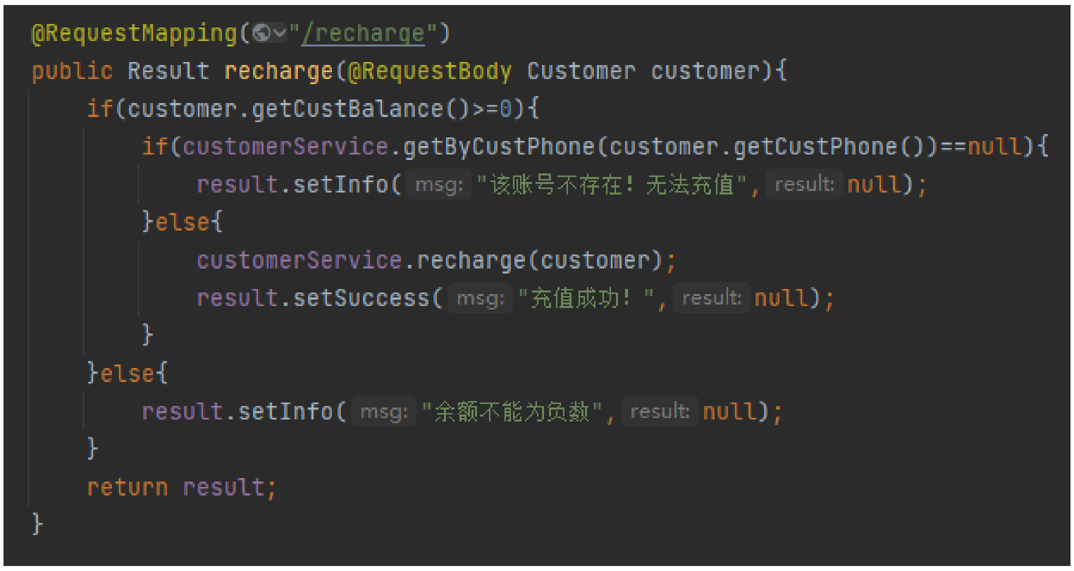
```
/recharge
```

接收参数：customer

返回参数：null

服务端逻辑：

1. 首先判断余额是否为负数，然后再判断手机号码是否存在



```java
@RequestMapping("/recharge")
public Result recharge(@RequestBody Customer customer){
    if(customer.getCustBalance()>=0){
        if(customerService.getByCustPhone(customer.getCustPhone())==null){
            result.setInfo( msg: "该账号不存在！无法充值", result: null);
        }else{
            customerService.recharge(customer);
            result.setSuccess( msg: "充值成功！", result: null);
        }
    }else{
        result.setInfo( msg: "余额不能为负数", result: null);
    }
    return result;
}
```

2. 然后再修改账户余额

```sql
update customer set cust_balance = #{custBalance} where cust_phone = #{custPhone}
```

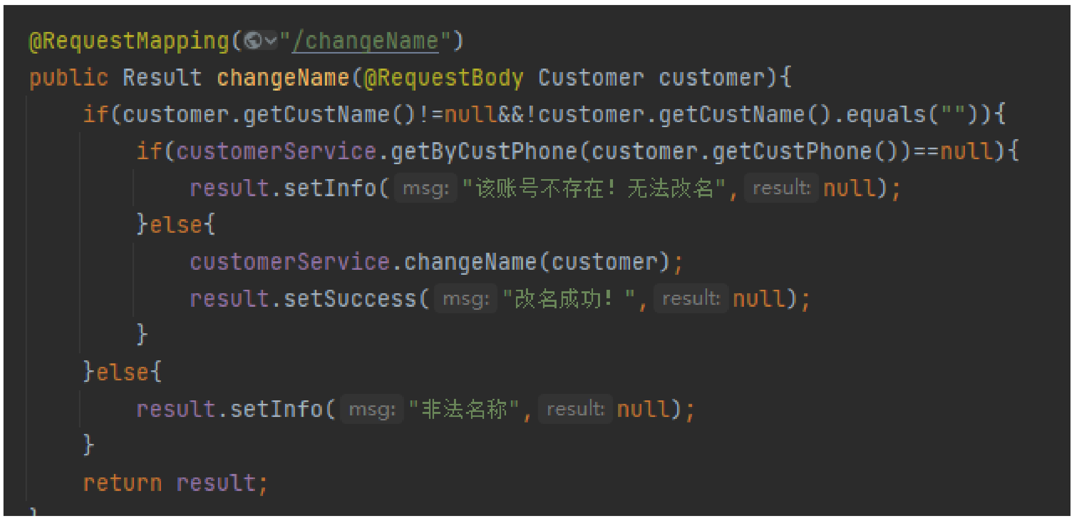
5.
```
/changeName
```

接收参数：customer

返回参数：null

服务端逻辑：

1. 首先判断名称是否合法，再判断账号是否存在



```java
@RequestMapping("/changeName")
public Result changeName(@RequestBody Customer customer){
    if(customer.getCustName()!=null&&!customer.getCustName().equals("")){
        if(customerService.getByCustPhone(customer.getCustPhone())==null){
            result.setInfo( msg: "该账号不存在！无法改名", result: null);
        }else{
            customerService.changeName(customer);
            result.setSuccess( msg: "改名成功！", result: null);
        }
    }else{
        result.setInfo( msg: "非法名称", result: null);
    }
    return result;
```

2. 然后进行改名

```sql
update customer set cust_name = #{custName} where cust_phone = #{custPhone}
```

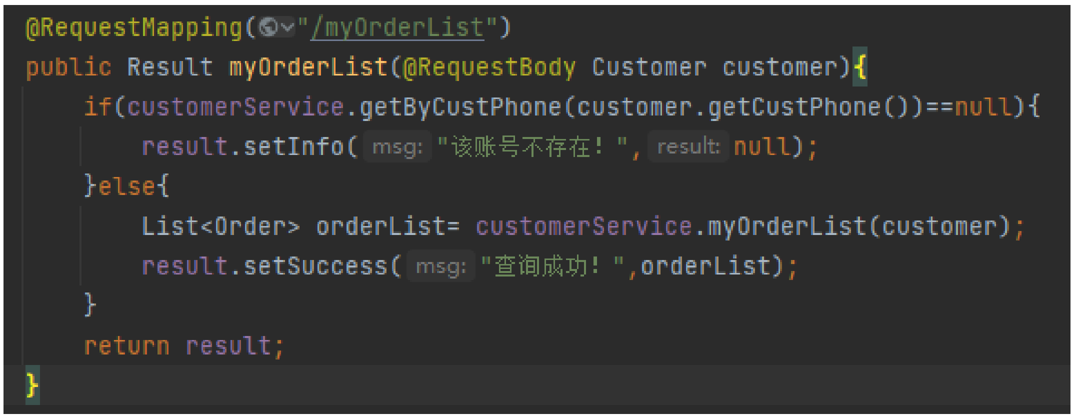
6.
```
/myOrderList
```

接收参数：customer

返回参数：List orderList

服务端逻辑：

1. 首先判断用户名是否存在



```java
@RequestMapping("/myOrderList")
public Result myOrderList(@RequestBody Customer customer){
    if(customerService.getByCustPhone(customer.getCustPhone())==null){
        result.setInfo( msg: "该账号不存在! ", result: null);
    }else{
        List<Order> orderList= customerService.myOrderList(customer);
        result.setSuccess( msg: "查询成功! ",orderList);
    }
    return result;
}
```

2. 然后进行查询

```sql
select * from `order` where cust_phone = #{custPhone}
```

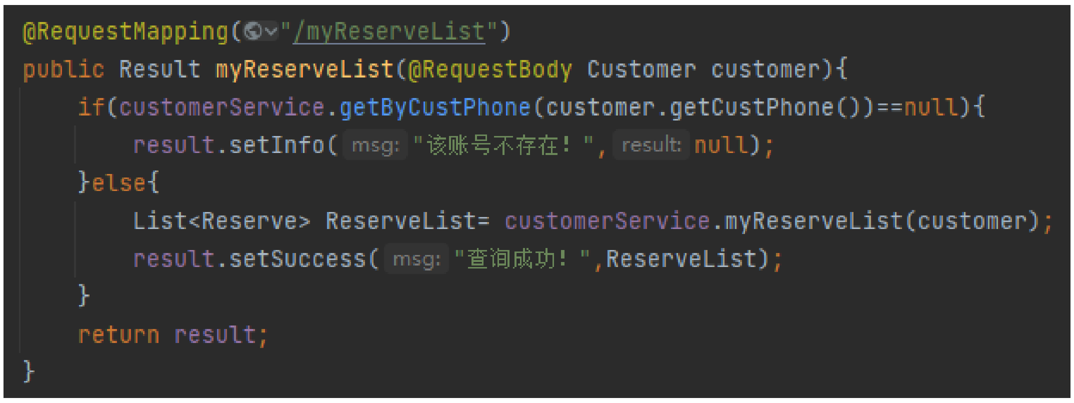
7.
```
/myReserveList
```

接收参数：customer

返回参数：List reserveList

服务端逻辑：

1. 首先判断用户名是否存在

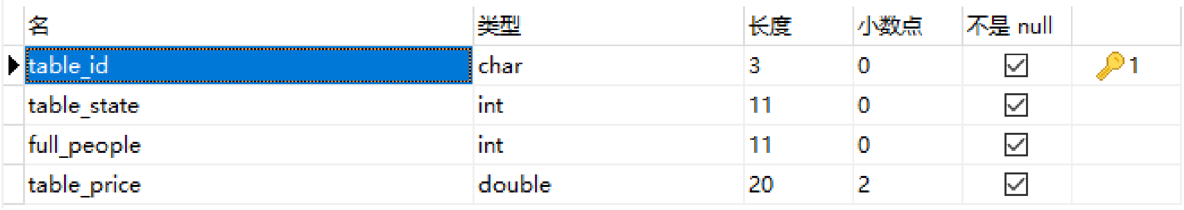


```java
@RequestMapping("/myReserveList")
public Result myReserveList(@RequestBody Customer customer){
    if(customerService.getByCustPhone(customer.getCustPhone())==null){
        result.setInfo( msg: "该账号不存在! ", result: null);
    }else{
        List<Reserve> ReserveList= customerService.myReserveList(customer);
        result.setSuccess( msg: "查询成功! ",ReserveList);
    }
    return result;
}
```

2. 然后进行查询

```sql
select * from reserve where cust_phone = #{custPhone}
```

table表(/table/)



名	类型	长度	小数点	不是 null	
table_id	char	3	0	✓	🔑1
table_state	int	11	0	✓	
full_people	int	11	0	✓	
table_price	double	20	2	✓	

注：table_id为3位字符串格式为“A01”首字母为餐桌等级，数字代表编号

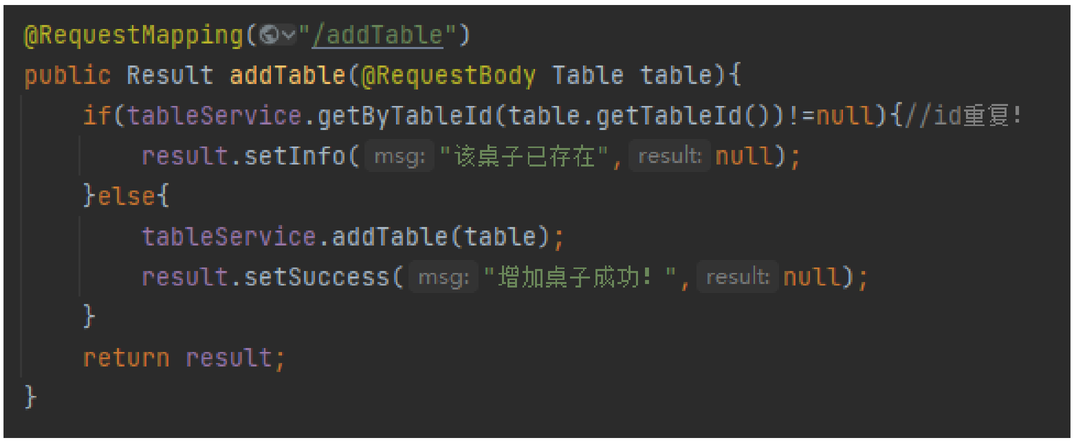
1.
```
/addTable
```

接收参数：table

返回参数：null

服务端逻辑:

1. 首先判断桌子是否重复



```java
@RequestMapping("/addTable")
public Result addTable(@RequestBody Table table){
    if(tableService.getByTableId(table.getTableId())!=null){//id重复!
        result.setInfo( msg: "该桌子已存在", result: null);
    }else{
        tableService.addTable(table);
        result.setSuccess( msg: "增加桌子成功! ", result: null);
    }
    return result;
}
```

```sql
select * from `table` where table_id like #{tableId}
```

2. 然后增加桌子

```sql
insert into `table` (table_id,full_people,table_price) values (#{tableId},#{fullPeople},#{tablePrice})
```

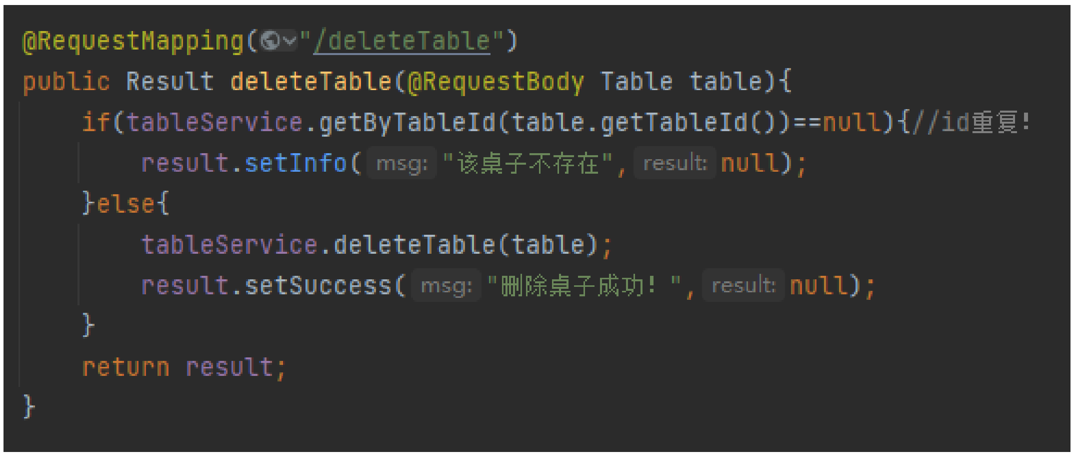
2.
```
/deleteTable
```

接收参数: table

返回参数: null

服务端逻辑:

1. 首先判断桌子是否存在



```java
@RequestMapping("/deleteTable")
public Result deleteTable(@RequestBody Table table){
    if(tableService.getByTableId(table.getTableId())==null){//id重复!
        result.setInfo( msg: "该桌子不存在", result: null);
    }else{
        tableService.deleteTable(table);
        result.setSuccess( msg: "删除桌子成功! ", result: null);
    }
    return result;
}
```

2. 然后删除桌子

```sql
delete from `table` where table_id = #{tableId}
```

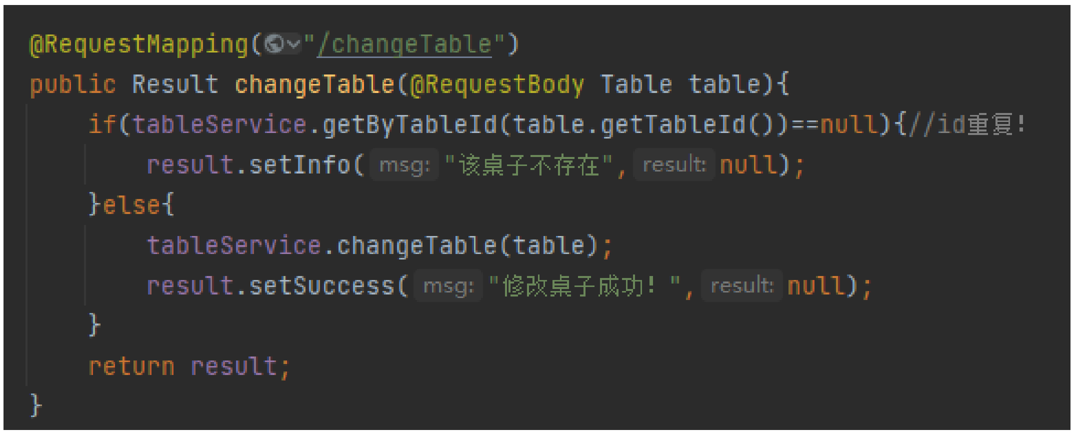
3.
```
/changeTable
```

接收参数: table

返回参数: null

服务端逻辑:

1. 首先判断桌子是否存在



```
@RequestMapping("/changeTable")
public Result changeTable(@RequestBody Table table){
    if(tableService.getByTableId(table.getTableId())==null){//id重复!
        result.setInfo( msg: "该桌子不存在", result: null);
    }else{
        tableService.changeTable(table);
        result.setSuccess( msg: "修改桌子成功! ", result: null);
    }
    return result;
}
```

2. 然后进行修改

```
update `table` set table_price=#{tablePrice},full_people=#{fullPeople} where table_id = #{tableId}
```

4.
```
/findTableList
```

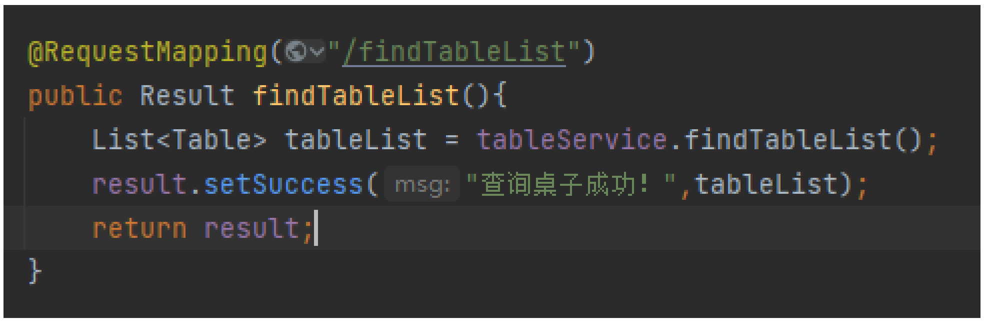
接收参数：null

返回参数：List

tableList

服务端逻辑：

1. 查询所有桌子



```
@RequestMapping("/findTableList")
public Result findTableList(){
    List<Table> tableList = tableService.findTableList();
    result.setSuccess( msg: "查询桌子成功! ",tableList);
    return result;
}
```

```
select * from `table`
```

5.
```
/findFreeTableList
```

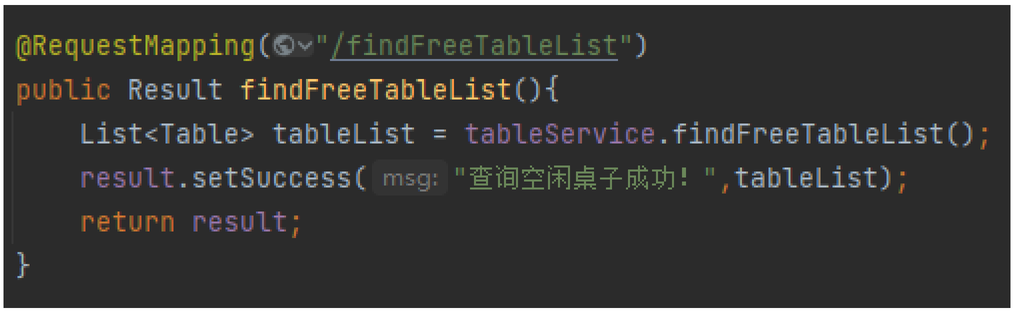
接收参数：null

返回参数：List

tableList

服务端逻辑：

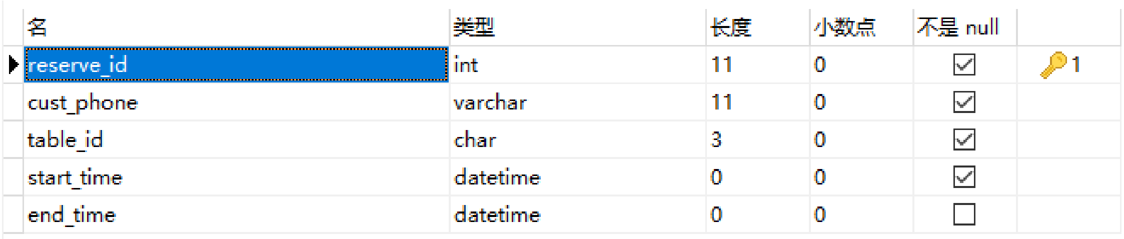
1. 查询当前空闲的桌子



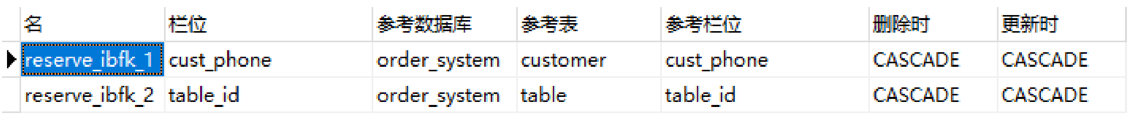
```java
@RequestMapping("/findFreeTableList")
public Result findFreeTableList(){
    List<Table> tableList = tableService.findFreeTableList();
    result.setSuccess( msg: "查询空闲桌子成功！",tableList);
    return result;
}
```

```sql
select * from `table` where table_state = 0
```

reserve表(/reserve/)



名	类型	长度	小数点	不是 null	
reserve_id	int	11	0	✓	🔑1
cust_phone	varchar	11	0	✓	
table_id	char	3	0	✓	
start_time	datetime	0	0	✓	
end_time	datetime	0	0	☐	



名	栏位	参考数据库	参考表	参考栏位	删除时	更新时
reserve_ibfk_1	cust_phone	order_system	customer	cust_phone	CASCADE	CASCADE
reserve_ibfk_2	table_id	order_system	table	table_id	CASCADE	CASCADE

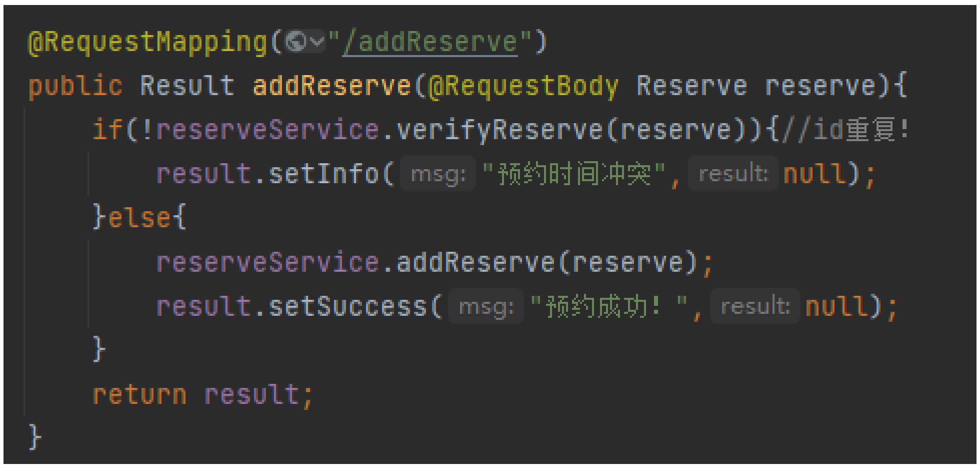
1.
   ```
   /addReserve
   ```

   接收参数：reserve

   返回参数：null

   服务端逻辑：

   1. 首先判断是否预约时间段冲突



```java
@RequestMapping("/addReserve")
public Result addReserve(@RequestBody Reserve reserve){
    if(!reserveService.verifyReserve(reserve)){//id重复!
        result.setInfo( msg: "预约时间冲突", result: null);
    }else{
        reserveService.addReserve(reserve);
        result.setSuccess( msg: "预约成功！", result: null);
    }
    return result;
}
```

```sql
select * from reserve as A where A.reserve_id not in (select B.reserve_id from reserve as B where B.start_time <![CDATA[ >= ]]> #{endTime} or B.endTime <![CDATA[ <= ]]> #{startTime}) and A.table_id = #{tableId}
```

   2. 再进行预约

```sql
insert into reserve (cust_phone,table_id,start_time,end_time) values (#{custPhone},#{tableId},#{startTime},#{endTime})
```

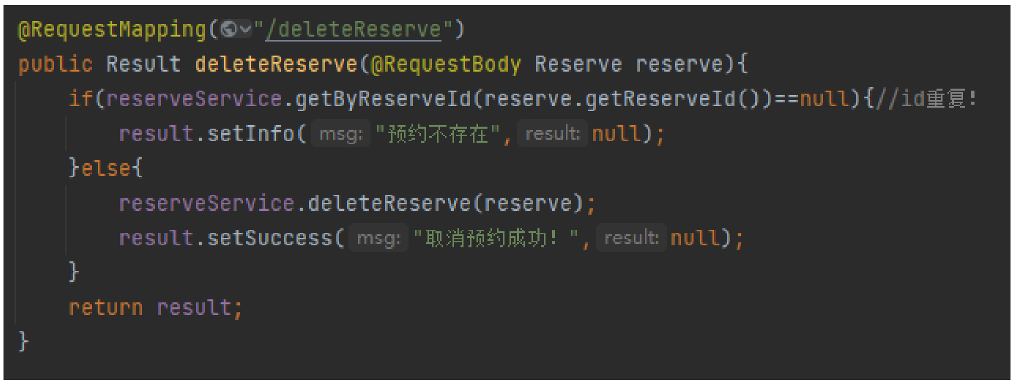
2.
   ```
   /deleteReserve
   ```

   接收参数：reserve

   返回参数：null

   服务端逻辑：

   1. 判断预约是否存在

      ```java
      @RequestMapping("/deleteReserve")
      public Result deleteReserve(@RequestBody Reserve reserve){
          if(reserveService.getByReserveId(reserve.getReserveId())==null){//id重复!
              result.setInfo( msg: "预约不存在", result: null);
          }else{
              reserveService.deleteReserve(reserve);
              result.setSuccess( msg: "取消预约成功! ", result: null);
          }
          return result;
      }
      ```

   2. 删除预约

      ```sql
      delete from reserve where reserve_id = #{reserveId}
      ```

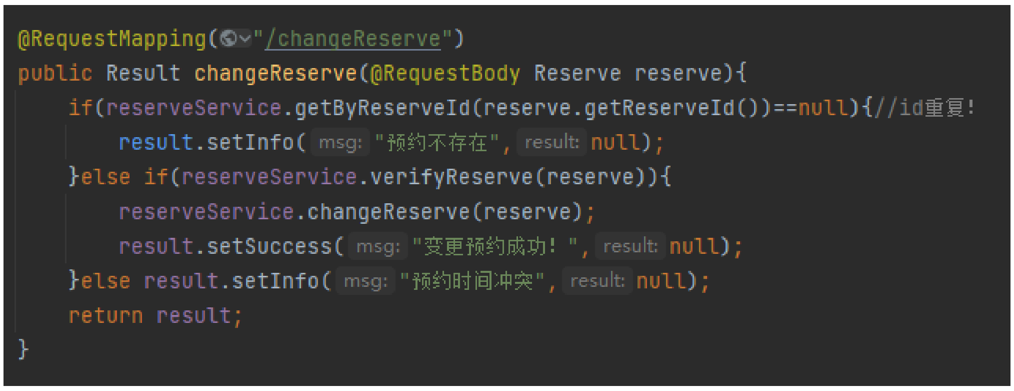
3.
   ```
   /changeReserve
   ```

   接收参数：reserve

   返回参数：null

   服务端逻辑：

   1. 首先判断预约是否存在，若存在再对新预约进行时间冲突校验。

      ```java
      @RequestMapping("/changeReserve")
      public Result changeReserve(@RequestBody Reserve reserve){
          if(reserveService.getByReserveId(reserve.getReserveId())==null){//id重复!
              result.setInfo( msg: "预约不存在", result: null);
          }else if(reserveService.verifyReserve(reserve)){
              reserveService.changeReserve(reserve);
              result.setSuccess( msg: "变更预约成功! ", result: null);
          }else result.setInfo( msg: "预约时间冲突", result: null);
          return result;
      }
      ```

   2.
      ```sql
      update reserve set table_id = #{tableId},start_time = #{startTime},end_time = #{endTime} where reserve_id = #{reserveId}
      ```

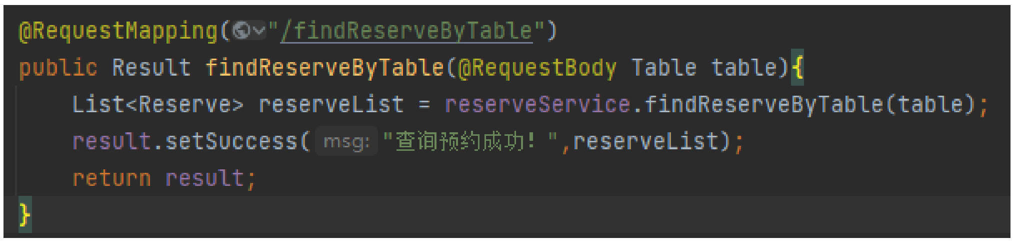
4.
   ```
   /findReserveByTable
   ```

   接收参数：table

   返回参数：List reserveList

   服务端逻辑：

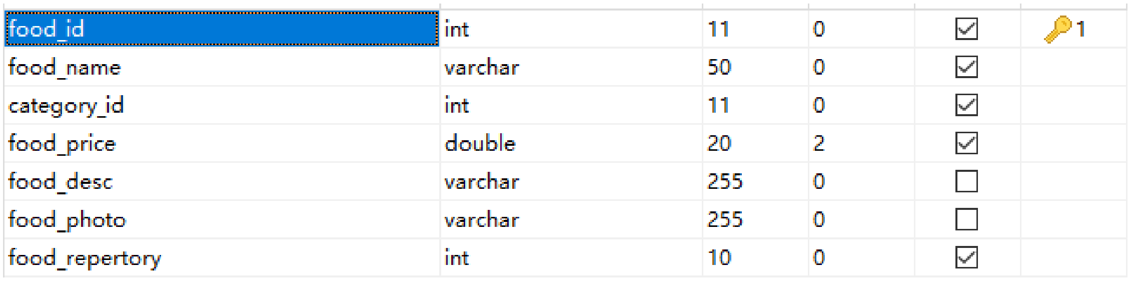
   1. 查询餐桌的预约



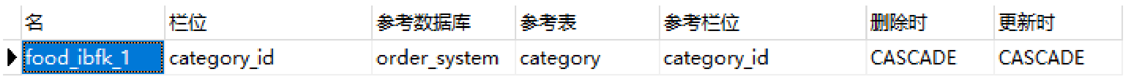
```java
@RequestMapping("/findReserveByTable")
public Result findReserveByTable(@RequestBody Table table){
    List<Reserve> reserveList = reserveService.findReserveByTable(table);
    result.setSuccess( msg: "查询预约成功！",reserveList);
    return result;
}
```

```sql
select * from reserve where table_id = #{tableId}
```

food表(/food/)



food_id	int	11	0	☑	🔑1
food_name	varchar	50	0	☑	
category_id	int	11	0	☑	
food_price	double	20	2	☑	
food_desc	varchar	255	0	☐	
food_photo	varchar	255	0	☐	
food_repertory	int	10	0	☑	



名	栏位	参考数据库	参考表	参考栏位	删除时	更新时
food_ibfk_1	category_id	order_system	category	category_id	CASCADE	CASCADE

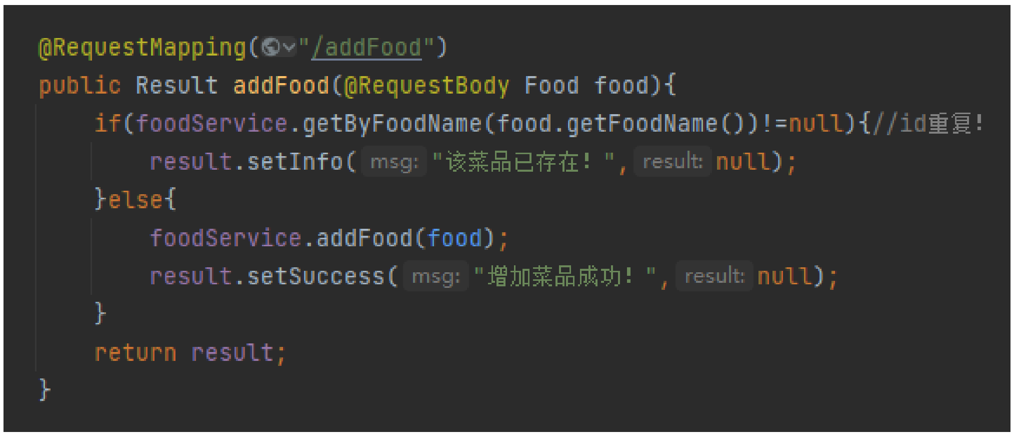
1.
```
/addFood
```

接收参数：food

返回参数：null

服务端逻辑：

1. 首先查询菜品是否重复



```java
@RequestMapping("/addFood")
public Result addFood(@RequestBody Food food){
    if(foodService.getByFoodName(food.getFoodName())!=null){//id重复!
        result.setInfo( msg: "该菜品已存在！", result: null);
    }else{
        foodService.addFood(food);
        result.setSuccess( msg: "增加菜品成功！", result: null);
    }
    return result;
}
```

```sql
select * from food where food_name like #{foodName}
```

2. 然后增加菜品

```sql
insert into food
(food_name,food_price,food_desc,food_photo,food_repertory,catego
ry_id) values (#{foodName},#{foodPrice},#{foodDesc},#
{foodPhoto},#{foodRepertory},#{categoryId})
```

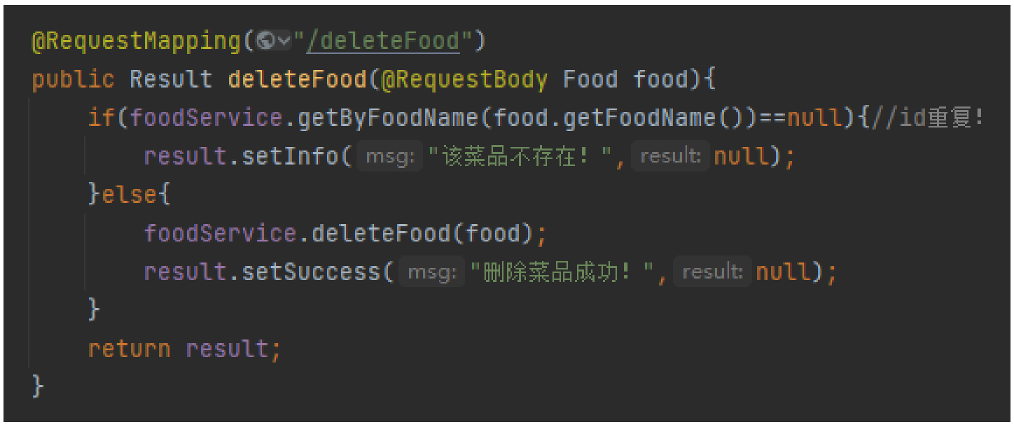
2.
```
/deleteFood
```

接收参数：food

返回参数：null

服务端逻辑：

1. 首先判断菜品是否存在



```
@RequestMapping("/deleteFood")
public Result deleteFood(@RequestBody Food food){
    if(foodService.getByFoodName(food.getFoodName())==null){//id重复!
        result.setInfo( msg: "该菜品不存在！", result: null);
    }else{
        foodService.deleteFood(food);
        result.setSuccess( msg: "删除菜品成功！", result: null);
    }
    return result;
}
```

2. 然后再删除菜品

```
delete from food where food_id = #{foodId}
```

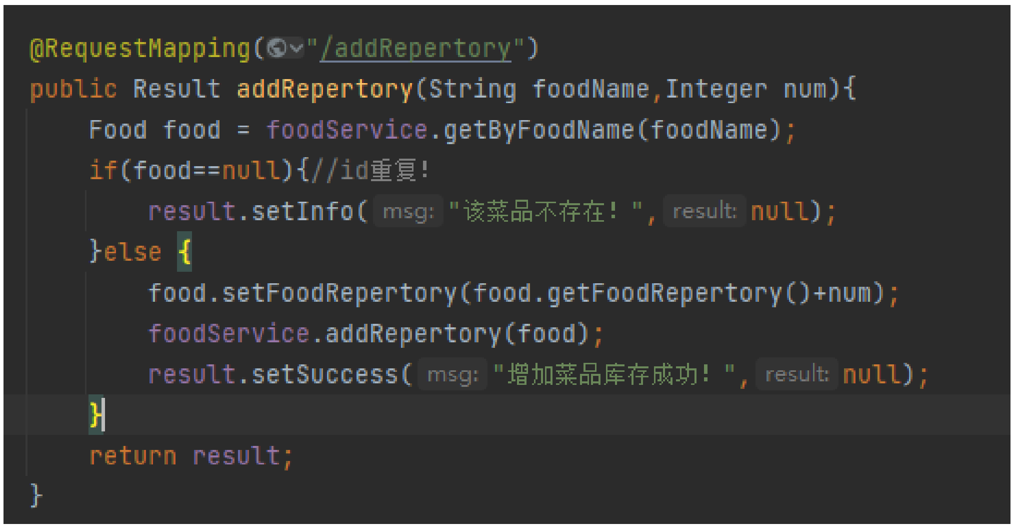
3.
```
/addRepertory
```

接收参数(get方法)： (String foodName,Integer num)

返回参数：null

服务端逻辑：

1. 首先判断菜品是否存在



```
@RequestMapping("/addRepertory")
public Result addRepertory(String foodName,Integer num){
    Food food = foodService.getByFoodName(foodName);
    if(food==null){//id重复!
        result.setInfo( msg: "该菜品不存在！", result: null);
    }else {
        food.setFoodRepertory(food.getFoodRepertory()+num);
        foodService.addRepertory(food);
        result.setSuccess( msg: "增加菜品库存成功！", result: null);
    }
    return result;
}
```

2. 然后再增加库存

```
update food set food_repertory = #{foodRepertory} where food_id = #{foodId}
```

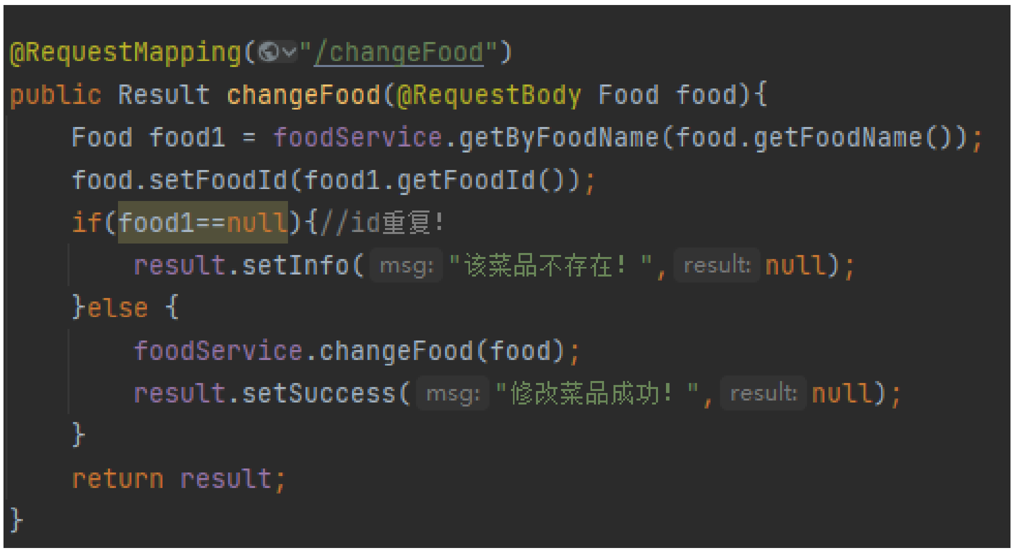
4.
```
/changeFood
```

接收参数：food

返回参数：null

服务端逻辑：

1. 首先判断菜品是否存在



```java
@RequestMapping("/changeFood")
public Result changeFood(@RequestBody Food food){
    Food food1 = foodService.getByFoodName(food.getFoodName());
    food.setFoodId(food1.getFoodId());
    if(food1==null){//id重复!
        result.setInfo( msg: "该菜品不存在！", result: null);
    }else {
        foodService.changeFood(food);
        result.setSuccess( msg: "修改菜品成功！", result: null);
    }
    return result;
}
```

2. 然后进行修改

```sql
update food set food_name = #{foodName},food_price =#{foodPrice},food_desc = #{foodDesc},food_photo = #{foodPhoto},category_id = #{categoryId} where food_id = #{foodId}
```

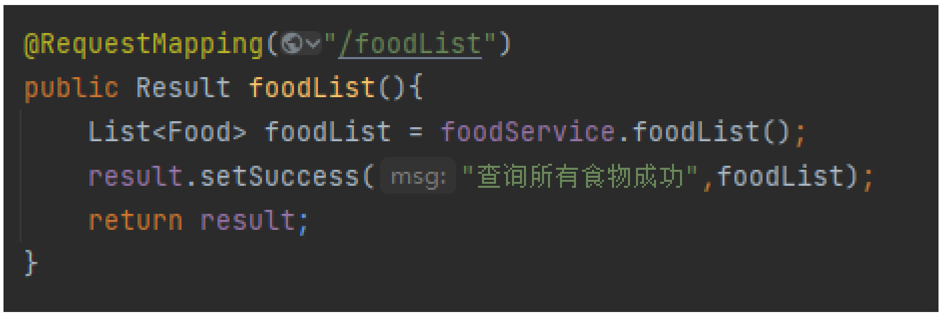
5.
```
/foodList
```

接收参数：null

返回参数：List foodList

服务端逻辑：

1. 查询所有菜品



```java
@RequestMapping("/foodList")
public Result foodList(){
    List<Food> foodList = foodService.foodList();
    result.setSuccess( msg: "查询所有食物成功",foodList);
    return result;
}
```

```sql
select * from food
```

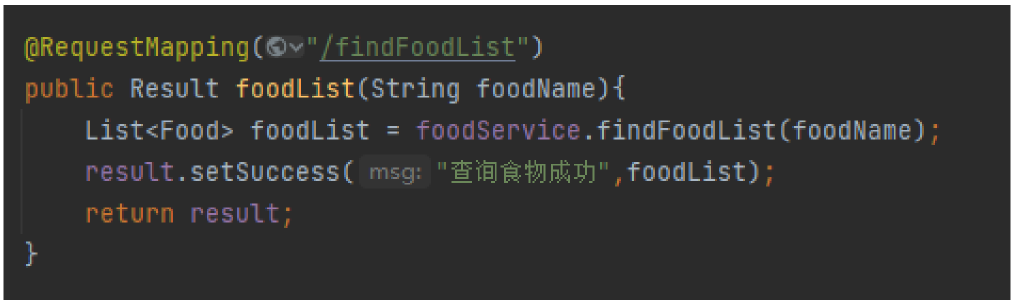
6.
```
/findFoodList
```

接收参数(get方法)：String foodName

返回参数：List foodList

服务端逻辑：

1. 根据菜品名查询菜品



```java
@RequestMapping("/findFoodList")
public Result foodList(String foodName){
    List<Food> foodList = foodService.findFoodList(foodName);
    result.setSuccess("查询食物成功",foodList);
    return result;
}
```

```sql
select * from food where food_name like concat('%',#
{foodName},'%')
```

7.
```
/findFoodListByCategory
```

接收参数：category

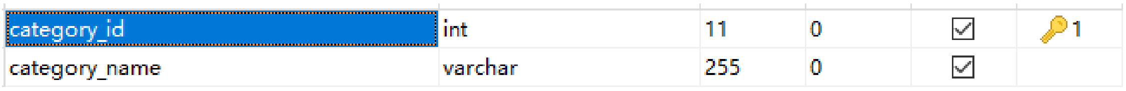
返回参数：List foodList

服务端逻辑：

1. 根据菜品类目查询菜品

```sql
select food.* from food,category where food.category_id =
category.category_id and category.category_name like #
{categoryName}
```

category表(/category/)



category_id	int	11	0	✓	🔑1
category_name	varchar	255	0	✓	

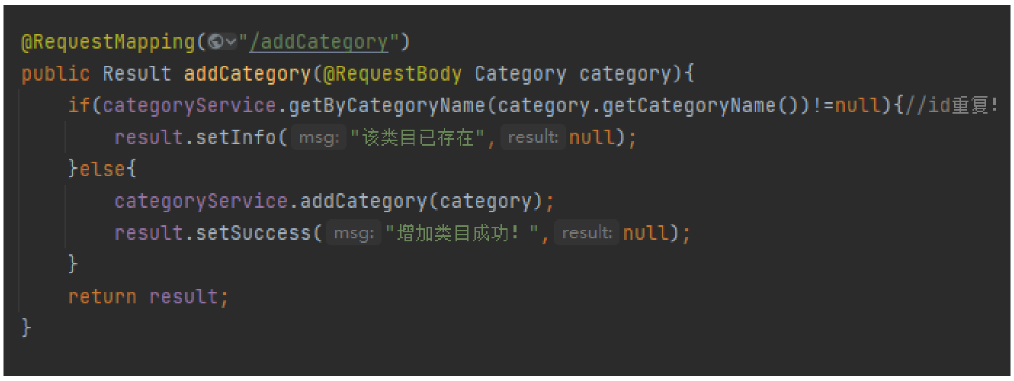
1.
```
/addCategory
```

接收参数：category

返回参数：null

服务端逻辑：

1. 判断菜品类目是否重复



```java
@RequestMapping("/addCategory")
public Result addCategory(@RequestBody Category category){
    if(categoryService.getByCategoryName(category.getCategoryName())!=null){//id重复!
        result.setInfo("该类目已存在", null);
    }else{
        categoryService.addCategory(category);
        result.setSuccess("增加类目成功! ", null);
    }
    return result;
}
```

```sql
select * from category where category_name like #{categoryName}
```

2. 增加类目

```
insert into category (category_id,category_name) values (#{categoryId},#{categoryName})
```

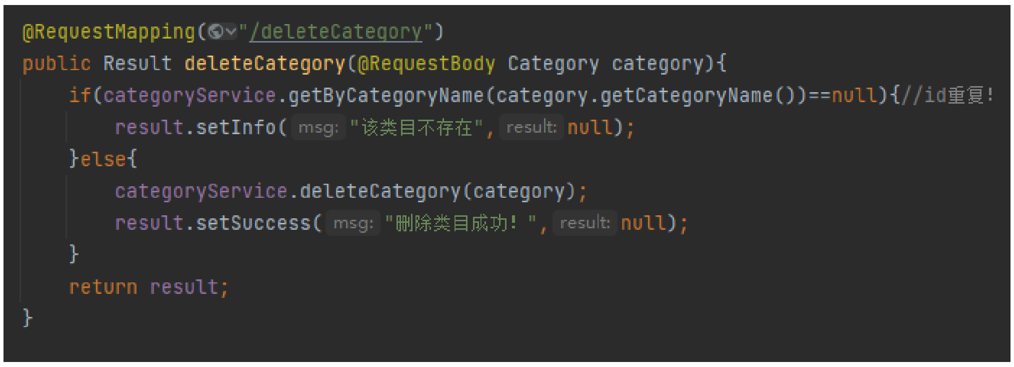
2.
   ```
   /deleteCategory
   ```

   接收参数：category

   返回参数：null

   服务端逻辑：

   1. 判断菜品类目是否存在



```
@RequestMapping("/deleteCategory")
public Result deleteCategory(@RequestBody Category category){
    if(categoryService.getByCategoryName(category.getCategoryName())==null){//id重复!
        result.setInfo( msg: "该类目不存在", result: null);
    }else{
        categoryService.deleteCategory(category);
        result.setSuccess( msg: "删除类目成功! ", result: null);
    }
    return result;
}
```

   2. 删除类目

```
delete from category where category_id = #{categoryId}
```

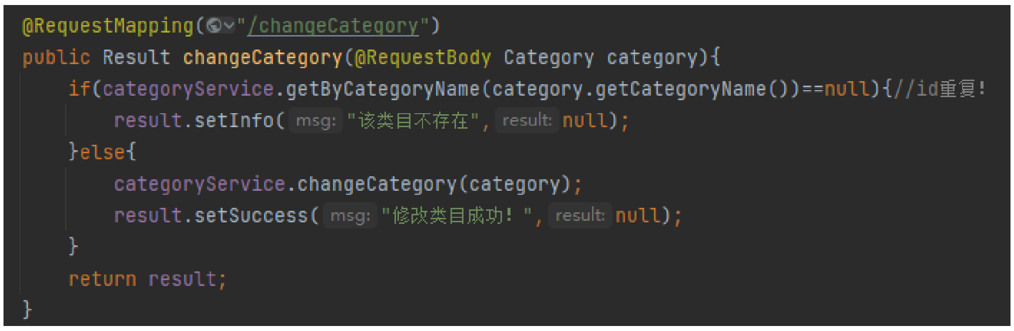
3.
   ```
   /changeCategory
   ```

   接收参数：category

   返回参数：null

   服务端逻辑：

   1. 判断类目是否存在



```
@RequestMapping("/changeCategory")
public Result changeCategory(@RequestBody Category category){
    if(categoryService.getByCategoryName(category.getCategoryName())==null){//id重复!
        result.setInfo( msg: "该类目不存在", result: null);
    }else{
        categoryService.changeCategory(category);
        result.setSuccess( msg: "修改类目成功! ", result: null);
    }
    return result;
}
```

   2. 修改类目

```
update category set category_name = #{categoryName} where category_id = #{categoryId}
```

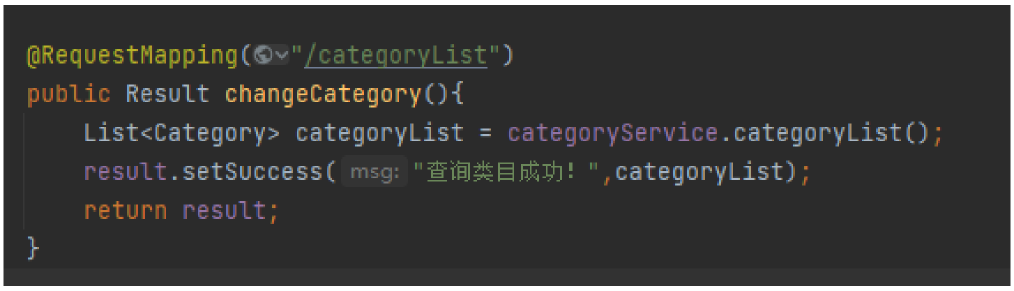
4.
   ```
   /categoryList
   ```

   接收参数：null

   返回参数：List categoryList

   服务端逻辑：

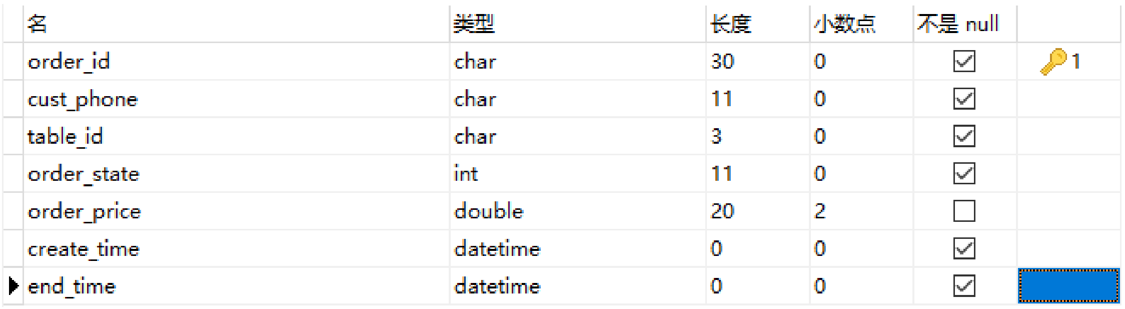
1. 查询所有类目



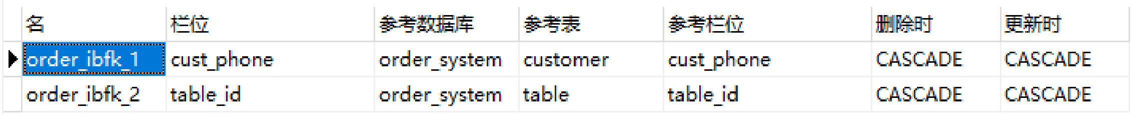
```java
@RequestMapping("/categoryList")
public Result changeCategory(){
    List<Category> categoryList = categoryService.categoryList();
    result.setSuccess( msg: "查询类目成功! ",categoryList);
    return result;
}
```

```sql
select * from category
```

order表(/order/)



名	类型	长度	小数点	不是 null	
order_id	char	30	0	✓	🔑1
cust_phone	char	11	0	✓	
table_id	char	3	0	✓	
order_state	int	11	0	✓	
order_price	double	20	2	☐	
create_time	datetime	0	0	✓	
end_time	datetime	0	0	✓	



名	栏位	参考数据库	参考表	参考栏位	删除时	更新时
order_ibfk_1	cust_phone	order_system	customer	cust_phone	CASCADE	CASCADE
order_ibfk_2	table_id	order_system	table	table_id	CASCADE	CASCADE

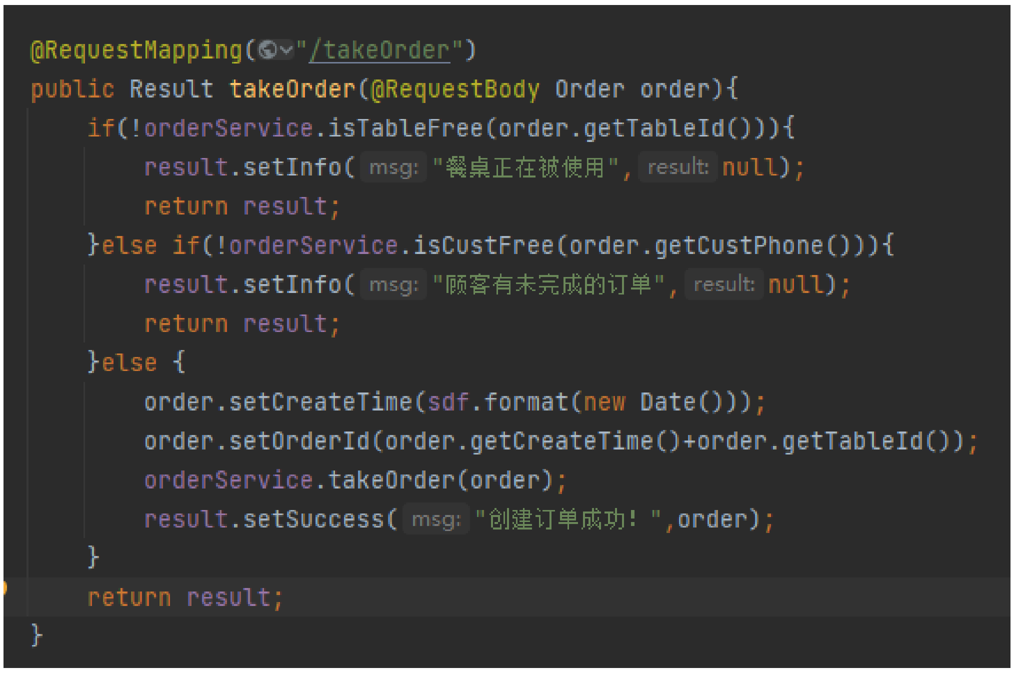
1.
```
/takeOrder
```

接收参数：order

返回参数：order

服务端逻辑：

1. 判断当前餐桌是否可以使用
2. 判断顾客是否有未完成的订单



```java
@RequestMapping("/takeOrder")
public Result takeOrder(@RequestBody Order order){
    if(!orderService.isTableFree(order.getTableId())){
        result.setInfo( msg: "餐桌正在被使用", result: null);
        return result;
    }else if(!orderService.isCustFree(order.getCustPhone())){
        result.setInfo( msg: "顾客有未完成的订单", result: null);
        return result;
    }else {
        order.setCreateTime(sdf.format(new Date()));
        order.setOrderId(order.getCreateTime()+order.getTableId());
        orderService.takeOrder(order);
        result.setSuccess( msg: "创建订单成功! ",order);
    }
    return result;
}
```

```sql
select * from `table` where table_state = 0 and table_id = #
{tableId}
```

```sql
select * from `order` where cust_phone = #{custPhone} and
order_state = 0
```

3. 创建订单

```sql
insert into `order` (order_id,cust_phone,table_id,create_time)
values (#{orderId},#{custPhone},#{tableId},#{createTime})
```

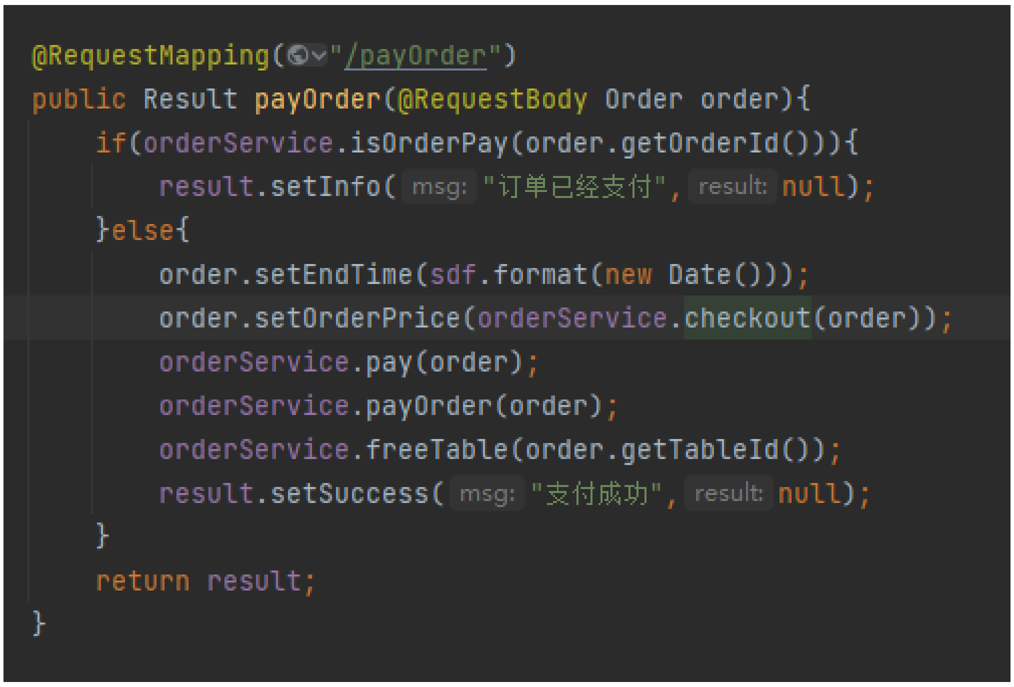
2.
```
/payOrder
```

接收参数：order

返回参数：null

服务端逻辑：

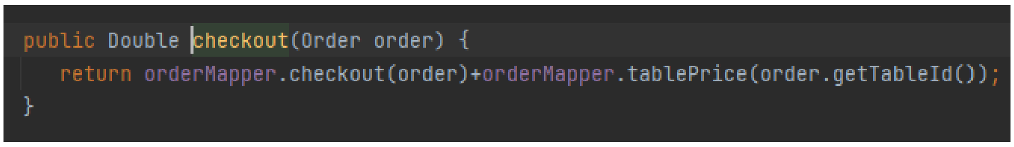
1. 判断订单是否支付



```java
@RequestMapping("/payOrder")
public Result payOrder(@RequestBody Order order){
    if(orderService.isOrderPay(order.getOrderId())){
        result.setInfo( msg: "订单已经支付", result: null);
    }else{
        order.setEndTime(sdf.format(new Date()));
        order.setOrderPrice(orderService.checkout(order));
        orderService.pay(order);
        orderService.payOrder(order);
        orderService.freeTable(order.getTableId());
        result.setSuccess( msg: "支付成功", result: null);
    }
    return result;
}
```

```sql
select * from `order` where order_id = #{orderId} and
order_state = 0
```

2. 设置订单的总价



```java
public Double checkout(Order order) {
    return orderMapper.checkout(order)+orderMapper.tablePrice(order.getTableId());
}
```

```sql
select sum(a.food_price * b.food_num) from food as a, record as
b where a.food_id = b.food_id and b.order_id = #{orderId}
```

```sql
select table_price from `table` where table_id = #{tableId}
```

3. 付款

```sql
update customer as a,`order` as b set a.cust_balance =
a.cust_balance - #{orderPrice} where a.cust_phone = #{custPhone}
```

4. 埋单

```sql
update `order` set order_state=1,order_price=#
{orderPrice},end_time=#{endTime} where order_id = #{orderId}
```

5. 改变桌子状态

```sql
update `table` set table_state = 0 where table_id = #{tableId}
```

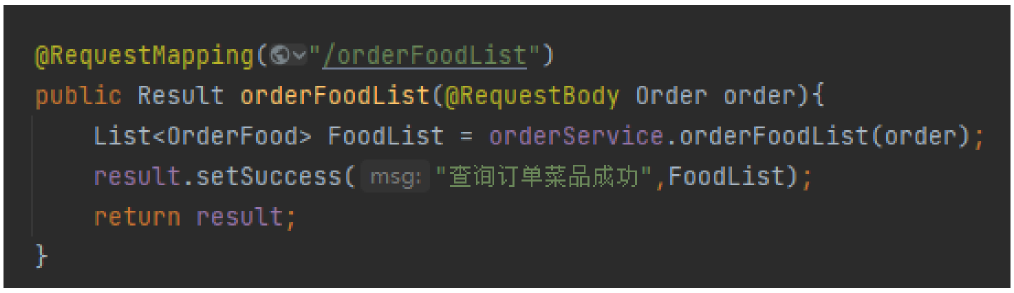
3.
```
/orderFoodList
```

接收参数：order

返回参数：List orderFoodList （注：OrderFood实体类唯一用到的地方）

服务端逻辑：

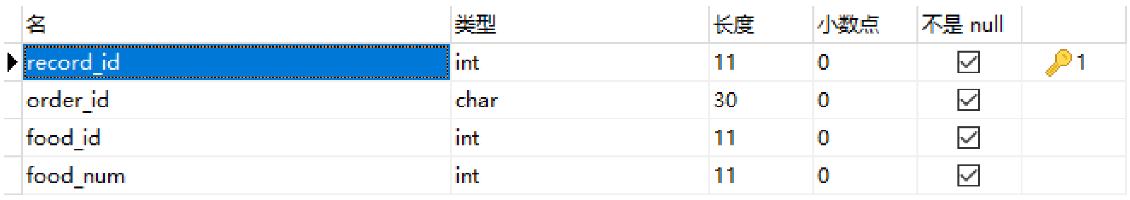
1. 查询订单菜品列表



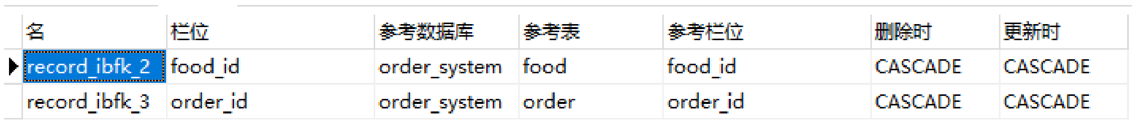
```java
@RequestMapping("/orderFoodList")
public Result orderFoodList(@RequestBody Order order){
    List<OrderFood> FoodList = orderService.orderFoodList(order);
    result.setSuccess( msg: "查询订单菜品成功",FoodList);
    return result;
}
```

```sql
select c.*,a.food_num from record as a,food as b where
a.order_id =#{orderId} and a.food_id = b.food_id
```

record表(/record/)



名	类型	长度	小数点	不是 null	
record_id	int	11	0	✓	🔑1
order_id	char	30	0	✓	
food_id	int	11	0	✓	
food_num	int	11	0	✓	



名	栏位	参考数据库	参考表	参考栏位	删除时	更新时
record_ibfk_2	food_id	order_system	food	food_id	CASCADE	CASCADE
record_ibfk_3	order_id	order_system	order	order_id	CASCADE	CASCADE

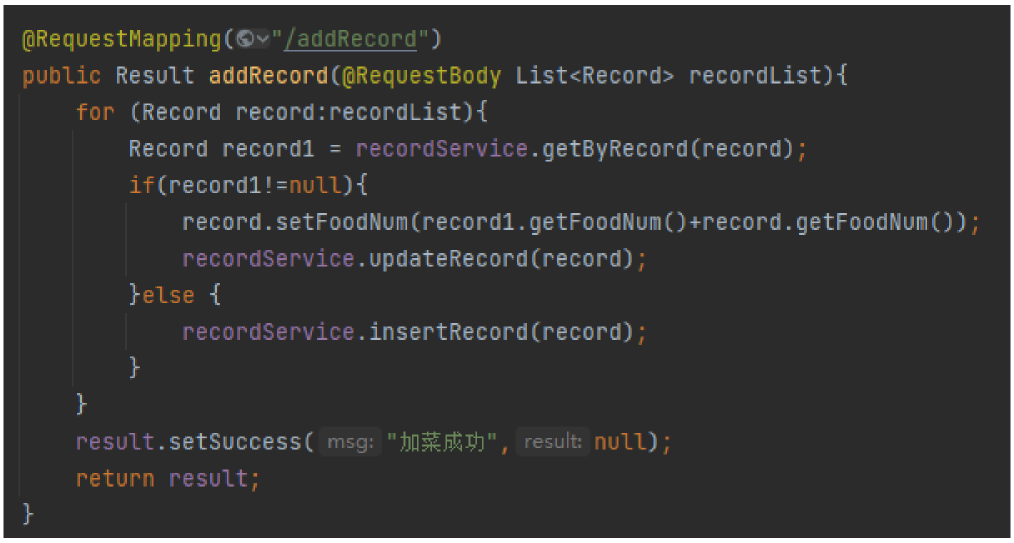
1.
```
/addRecord
```

接收参数：List recordList

返回参数：null

服务端逻辑：

1. 查询记录中是否有重复记录



```java
@RequestMapping("/addRecord")
public Result addRecord(@RequestBody List<Record> recordList){
    for (Record record:recordList){
        Record record1 = recordService.getByRecord(record);
        if(record1!=null){
            record.setFoodNum(record1.getFoodNum()+record.getFoodNum());
            recordService.updateRecord(record);
        }else {
            recordService.insertRecord(record);
        }
    }
    result.setSuccess( msg: "加菜成功", result: null);
    return result;
}
```

```sql
select * from record where order_id = #{orderId} and food_id = #{foodId}
```

2. 如果有，则库存叠加

```sql
update record set food_num = #{foodNum} where order_id = #{orderId} and food_id = #{foodId}
```

3. 如果没有，则增加该记录

```sql
insert into record (order_id,food_id,food_num) values (#{orderId},#{foodId},#{foodNum})
```

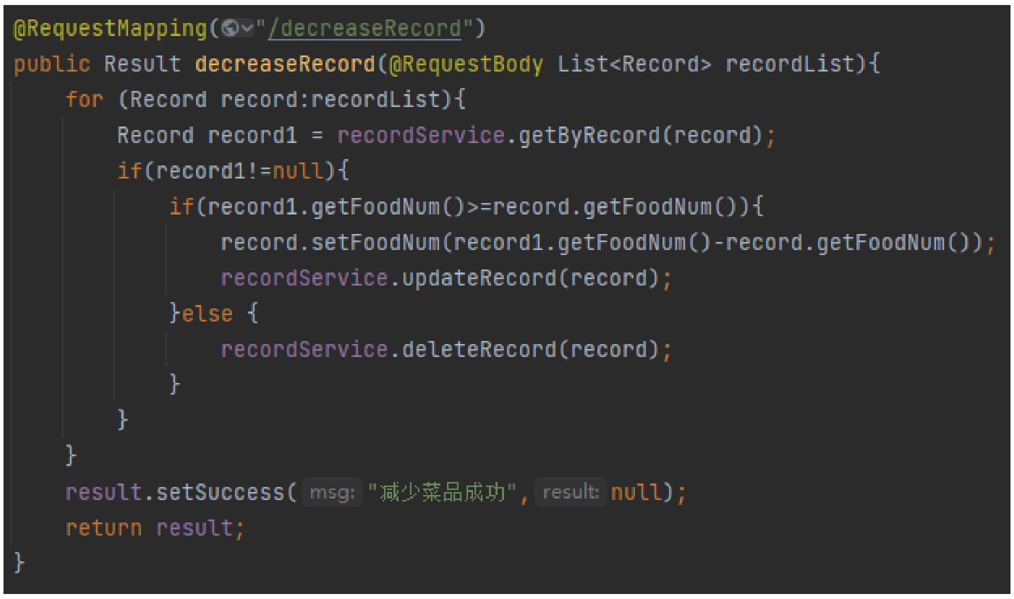
2.
```
/decreaseRecord
```

接收参数：List recordList

返回参数：null

服务端逻辑：

1. 查询记录中是否存在该记录



```java
@RequestMapping("/decreaseRecord")
public Result decreaseRecord(@RequestBody List<Record> recordList){
    for (Record record:recordList){
        Record record1 = recordService.getByRecord(record);
        if(record1!=null){
            if(record1.getFoodNum()>=record.getFoodNum()){
                record.setFoodNum(record1.getFoodNum()-record.getFoodNum());
                recordService.updateRecord(record);
            }else {
                recordService.deleteRecord(record);
            }
        }
    }
    result.setSuccess( msg: "减少菜品成功", result: null);
    return result;
}
```

2. 判断数据库中记录数量若大于减少数量，则直接减少数量

```sql
update record set food_num = #{foodNum} where order_id = #{orderId} and food_id = #{foodId}
```

3. 若小于减少数量，则直接删除该记录

```sql
delete from record where order_id = #{orderId} and food_id = #{foodId}
```