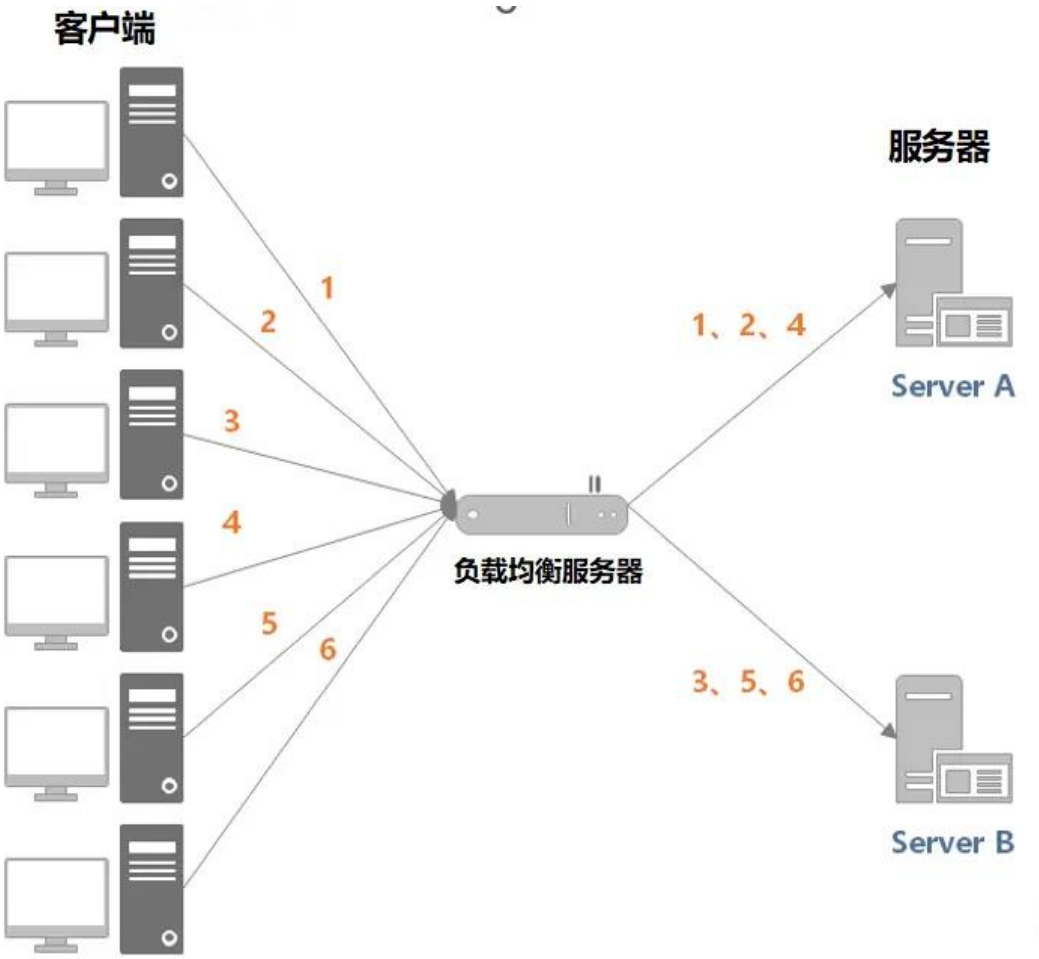
一、技术点

1.Nginx 代理和负载均衡

1.1 负载均衡

负载均衡：负载均衡服务器通过一定的调度算法将客户端的流量分发到不同的应用服务器上面，以实现性能的水平扩展及避免单点故障出现。

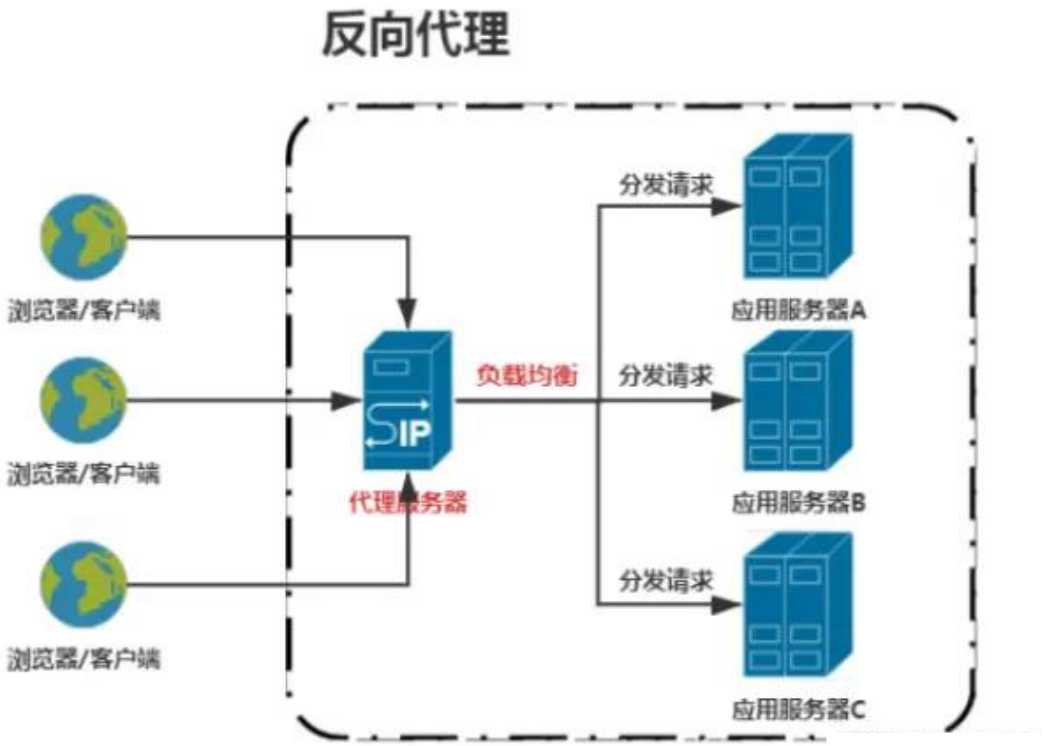


1.2 正向代理

正向代理是指通过代理服务器代理浏览器/客户端去重定向请求访问到目标服务器。

1.3 反向代理　-- 跨域

反向代理，指以代理服务器来接受 Internet 上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给 Internet 上请求连接的客户端，此时代理服务器对外就表现为一个服务器。



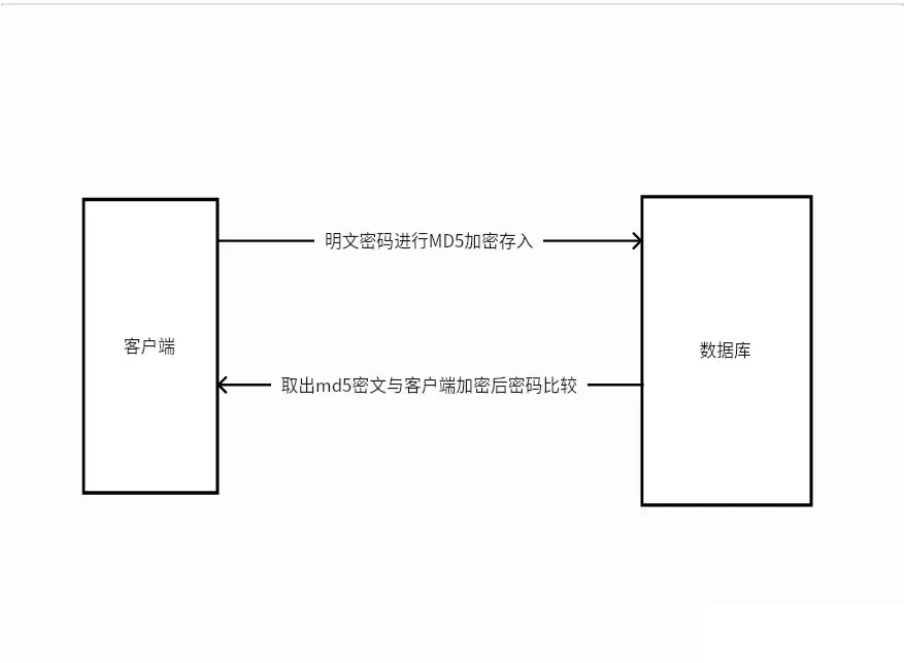
1.4 Nginx 反向代理配置--nginx.conf

```
server {
    listen       92;      前端页面地址
    server_name  localhost;


    location / {
        root   html/backend;
        index  index.html index.htm;
     }


    location ^~ /admin/ {
        rewrite ^/admin/(.*)$ /$1 break;  # 路径重写    /admin/index/login  -->  /index/login
        proxy_pass http://localhost:8080;后台开发代理地址
    }

}
```

2. MD5 密码加密

特点：固定长度，不可逆

API: DigestUtils.md5DigestAsHex(password.getBytes());



3. Swagger - 框架

3.1 作用：接口文档的在线生成—doc.html

3.2 集成：1.导入 Knife4j 依赖　2.加入配置，设置资源映射　3.使用注解在对应位置

3.3 注解:

@Api(tags = “”)—用在类上

@ApiModel—类上

@ApiModelProperty--用在属性上

@ApiOperation（方法上）

4.JWT

4.1 概述：将原始 JSON 进行安全封装，由 Header（令牌类型、签名算法等），载荷（携带信息），签名（防伪）组成

4.2 应用场景 ：身份验证、授权、信息交换

4.3 开发流程： 1.导入 JWT 依赖 2.编写工具类 3.利用工具类下发/解析 JWT 实现功能

5.过滤器 Filter - servlet 容器

5.1 应用场景：防止未登录就进入界面 验证用户身份 缓存控制

5.2 开发流程：

1.定义一个实现类实现 Filter 接口，并重写方法（init 初始化方法、doFilter 拦截到请求时调用、destroy 销毁方法）-- 生命周期

2.配置过滤器：加上@WebFliter（urlPatterns = “”）注解，配置拦截器路径

3.启动类中通过@ServletComponentScan 开启扫描

4.放行：filterChain.doFilter(servletRequest,servletResponse);

5.3 FilterChain:在 WEB 应用中异常编写多个过滤器,过滤器按顺序执行

6.拦截器 Interceptor - IOC 容器

6.1 概念：由 Spring 框架提供，动态拦截请求，本质是面向切面编程（AOP）的

6.2 应用场景：登录验证，权限验证，日志记录，性能监控……

6.3 开发流程：

1. 自定义类实现接口 HandlerInterceptor，重写 preHandle 方法，交给 IOC 容器管理。

2.自定义拦截器注册类

7.异常处理 - 全局异常处理器

7.1 @RestControllerAdvice 被用来定义全局异常处理程序和全局响应结果处理程序。

7.2 @ExceptionHandler 用来自定义异常

8.AOP 切面

8.1 概念：将横切关注点与业务逻辑分离，将其通用行为封装至横向模块，以达到对业务逻辑的增强。

8.2 应用场景：记录日志操作、权限管理、事务管理

8.3 操作日志使用案例：1、起步依赖 2、编写自定义注解类，确定使用时间和空间 3、定义切面类（@Aspect），确定通知类型（@Before/@After/@Around），在方法中编写代码 4、根据注解类的限制，在需要使用的方法/类上增加注解进行使用

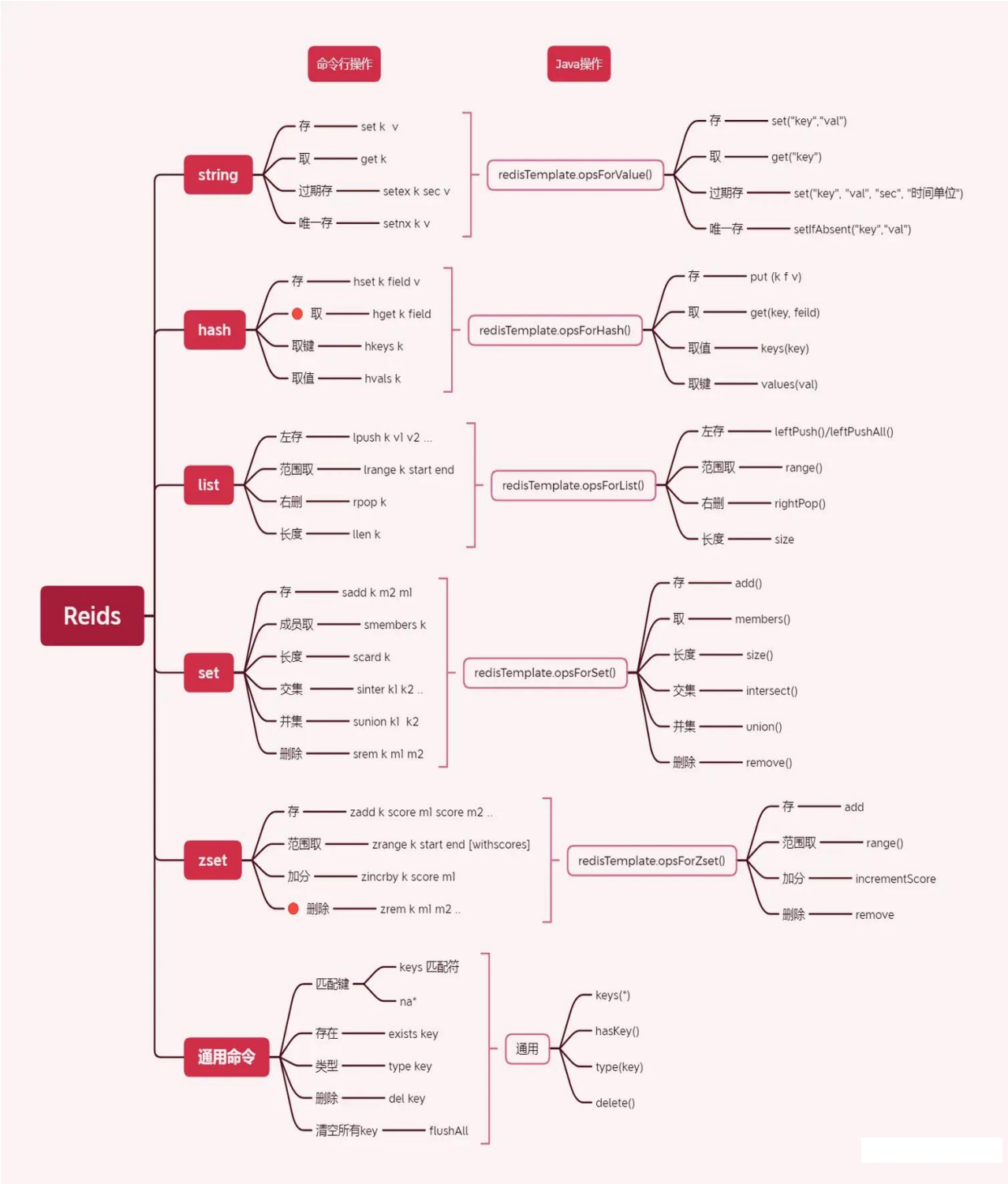
9.redis 非关系型数据库

9.1 概念：基于内存的 key-value 结构数据库

9.2 应用场景：缓存，消息队列，排行榜，分布式锁……

9.3 数据类型及常用命令：

9.4 用法：1、引入依赖 2、配置 redis 数据源 3、编写配置类，配置 RedisTemplate 对象 4、调用 RedisTemplate 对象，根据相关命令操作 redis



10.HttpClient

10.1 作用：发送 Http 请求，接收响应数据

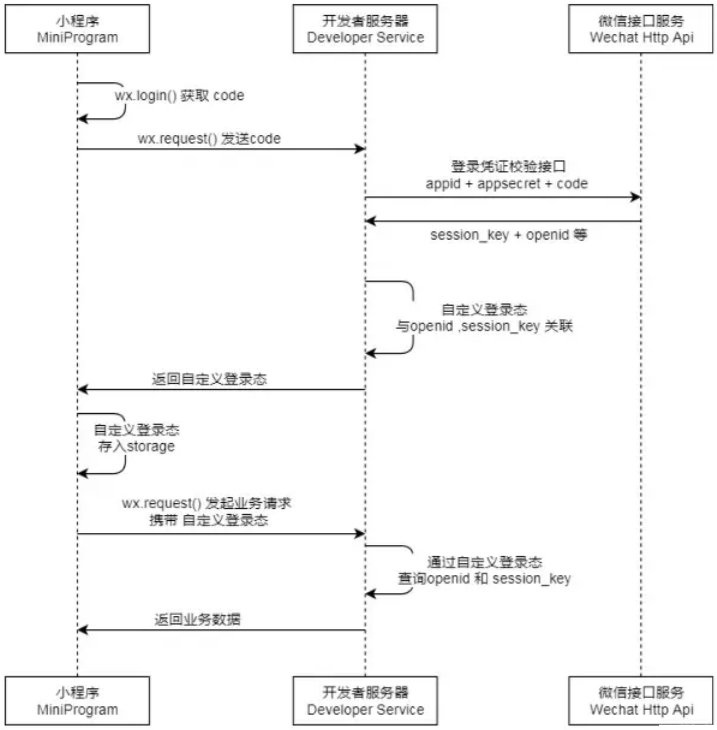
10.2 官网：[Apache HttpComponents – HttpClient Quick Start]

10.3 开发流程：1、创建 HttpClient 对象　2、创建请求方法的实例，并指定请求 URL　3、调用 HttpClient 对象的 execute 执行请求 4、释放连接

11.微信小程序

11.1 开发流程：开发文档

developers.weixin.qq.com/miniprogram…



12.Spring Cache

12.1 概念：基于注解的缓存功能，简化开发

12.2 应用场景：缓存数据，防止重复请求，实现分布式系统中的数据共享……

12.3 常用注解：

@EnableCaching 开启缓存 – 加在启动类上

@Cacheable（cacheNames = “”, key = “spel 表达式”）返回缓存数据，无数据则查数据库进行缓存并返回

@CachePut（value = “”,key = “spel 表达式”） 放入缓存

@CacheEvict（cacheNames = " ",key = "spel"） 删除缓存

12.4 开发流程：

1.引入依赖 2.开启缓存 3.使用注解进行缓存

13.redis 序列化方式

1. JdkSerializationRedisSerializer：默认，它使用 Java 的序列化机制将 Java 对象序列化为字节数组
2. StringRedisSerializer ：默认，使用 String 类型作为 Redis 的 key 和 value 的序列化方式。它可以将 Java 对象转换为字符串，也可以将字符串转换回 Java 对象
3. GenericToStringSerializer ：可以将任何对象泛化为字符串并序列化
4. Jackson2JsonRedisSerializer ：一种更加轻量级的序列化方式，它仅仅支持 JSON 格式的序列化和反序列化
5. GenericJackson2JsonRedisSerializer：使用 Jackson 库将 Java 对象序列化为 JSON 格式的字符串，以便在 Redis 中存储和检索 — 常用

14.Spring Task – 任务调度工具

14.1 应用场景：定时推送，系统未支付订单……

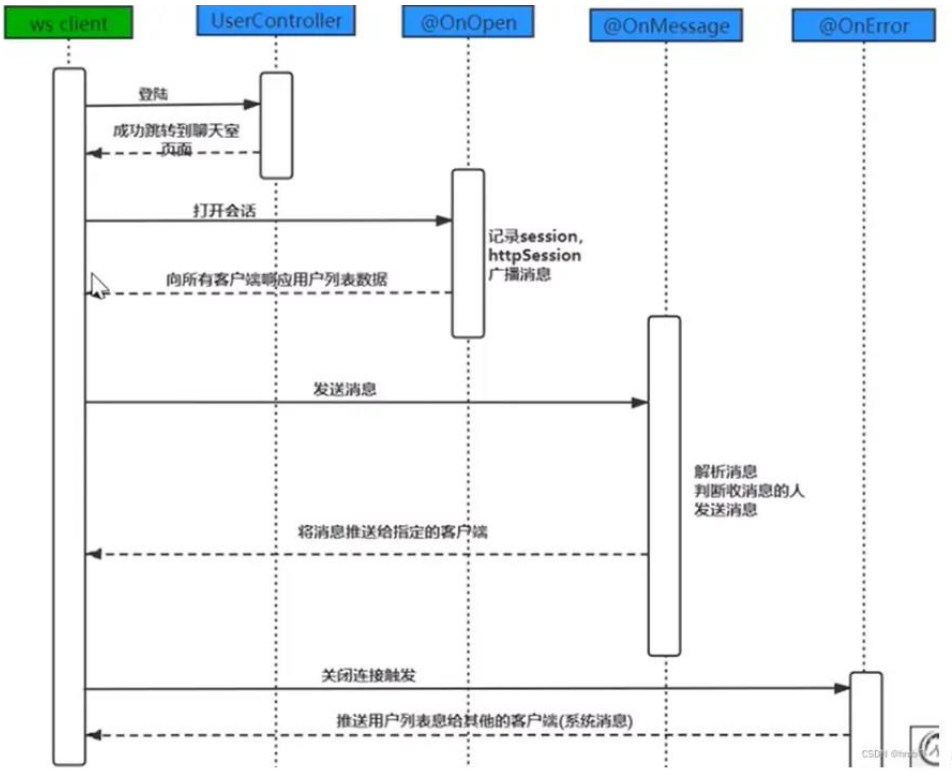
14.2 开发流程：1.@ EnableScheduling 2.新建测试类交给 IOC 容器管理 3. @Scheduled(cron = "cron 表达式")

15.WebSocket

15.1 概念：基于 TCP 连接的全双工通信网络协议

15.2 应用场景：用于实时通信，实时双向传输数据 – 弹幕、网页聊天…

15.3 开发流程：



16.阿里云 OSS

16.1 概念:云存储服务,存储文本、图片、视频

16.2 应用场景:上传文件

16.3 开发流程:

1. 导入依赖　2.引入工具类 Utils　3.编写配置类 config，属性实体类 properties

4.在配置文件 yam 中设置 endpoint、accessKeyId、accessKeySecret、bucketName

5. 编写文件上传逻辑代码

17.事务处理 translational

17.1 概念：指一组原子性的操作序列，这些操作要么全部执行，要么全部不执行

17.2 特性：原子性、一致性、隔离性和持久性

17.3 应用场景：当一组操作需要同时执行时。--金融行业、电商平台、物流行业

17.4 开发流程：在需要事务管理的方法上加注解：@Transactional（）

1. rollbackFor 属性:回滚指定类型的异常：@Transactional(rollbackFor = Exception.class)
2. Propagation 属性 ：

1 事务传播：当一个事务方法被另一个事务方法调用时，这个事务方法应该如何进行事务控制。

1 REQUIRED -- 【默认值】需要事务，有则加入，无则创建新事务

1 REQUIRES_NEW --总是创建新事务

18.Apache Echarts

Apache ECharts

19.Apache POI

poi.apache.org/components/…

19.1 概念：操作 Excel 文件

19.2 应用场景：导出 Excel

19.3 开发流程：

1.导入依赖

2.新建文件对象 new XSSFWorkbook();

3.创建 Sheet 页 excel.createSheet(“”)

4.创建行 sheet.createRow(0)—0 为第一行

5.创建单元格 row.createCell(0) - row 行第一格

6.赋值 setCellValue

7.写出磁盘，关流

20. 排除依赖

通过添加标签来排除某些依赖，从而避免这些依赖传递到子模块中

21. @ConfigurationProperties

21.1 概念：注入外部配置的属性

21.2 注解：

- @EnableConfigurationProperties 开启配置
- @Value 注解只能一个一个的进行外部属性的注入。
- @ConfigurationProperties 可以批量的将外部的属性配置注入到 bean 对象的属性中，且能注入数组结构

21.3 使用：使用在类上，eg: @ConfigurationProperties(prefix = "yam 文件对应的路径")

22. Mysql 嵌套查询

1. 时间格式化：date_format(order_time, '%Y-%m-%d')

```
select date_format(order_time, '%Y-%m-%d')                                                    time,
       (select sum(amount) from orders where date_format(order_time, '%Y-%m-%d') = time) turnover,
       (select count(*) from orders where date_format(order_time, '%Y-%m-%d') = time)     valid_order_count,
       (select ((select count(*) from orders where date_format(order_time, '%Y-%m-%d') = time and status = 5) /
                (select count(*)
                 from orders
                 where date_format(order_time, '%Y-%m-%d') = time)))                        order_completion_rate,
       (select (turnover /
                (select count(distinct user_id)
                 from orders
                 where date_format(order_time, '%Y-%m-%d') = time)))                        unit_price,
       (select count(*) from user where date_format(create_time, '%Y-%m-%d') = time)      new_users
from orders
group by time
having time between #{begin} and #{end}
order by time
```

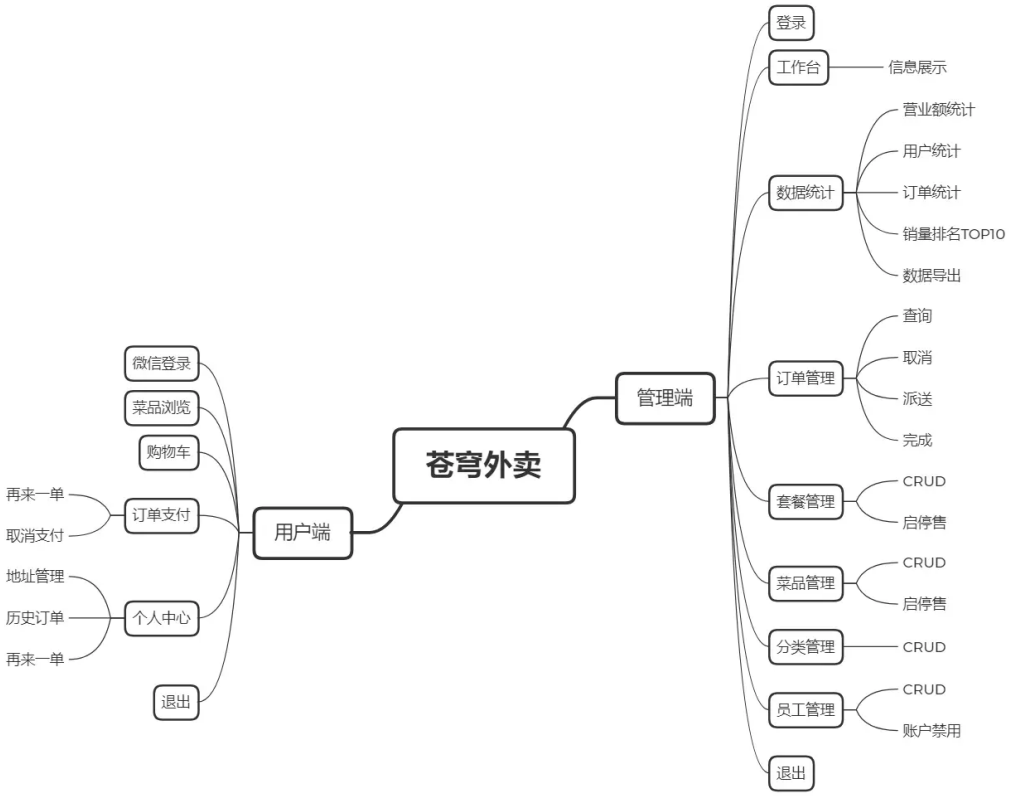
23. 验证码

[概述 | Hutool]

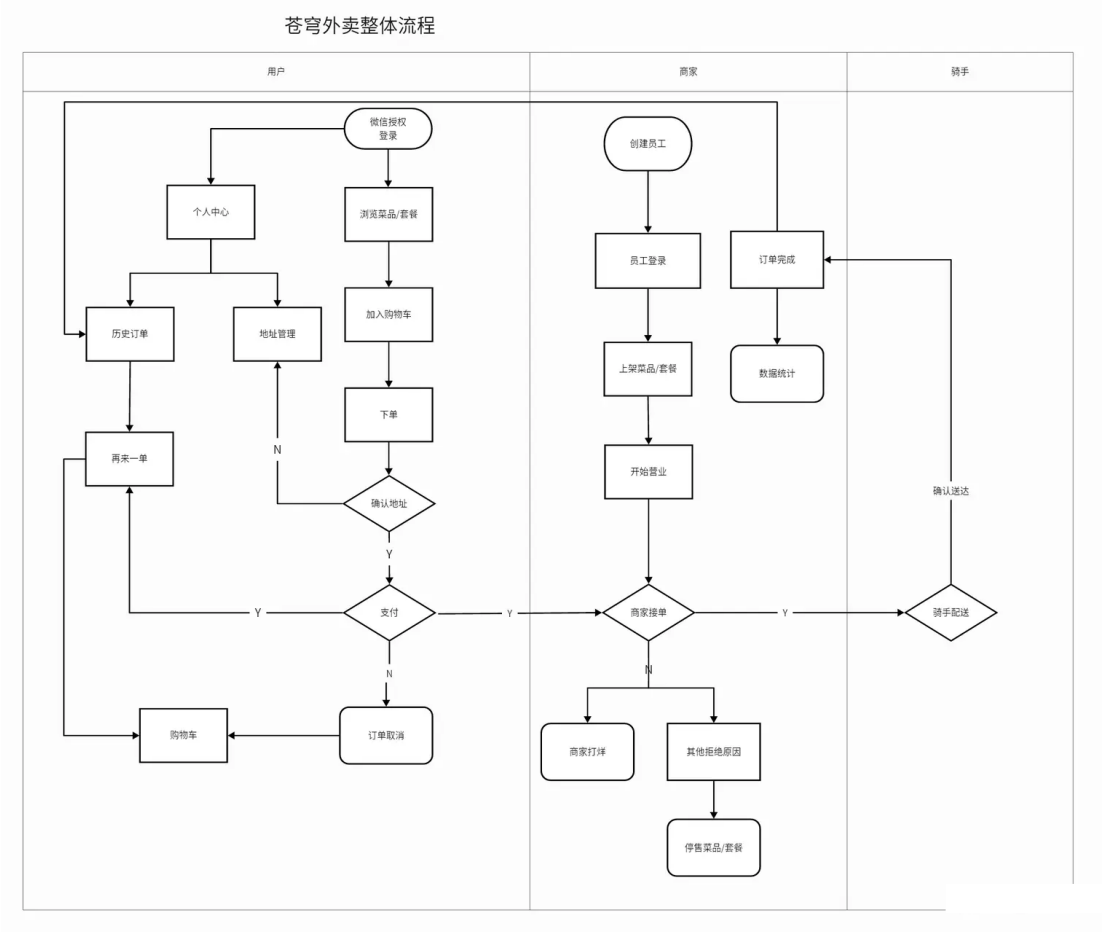
利用 Hutool 工具生成验证码,并保存验证码至数据库,方便验证

二、工作流程图

1.业务功能图

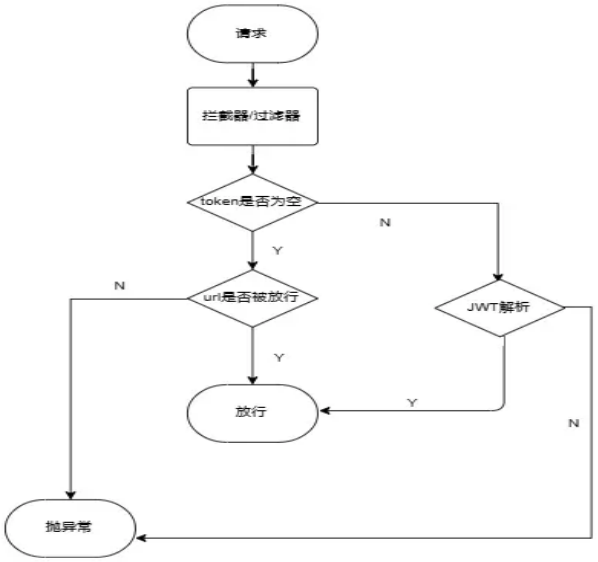


2.业务功能图

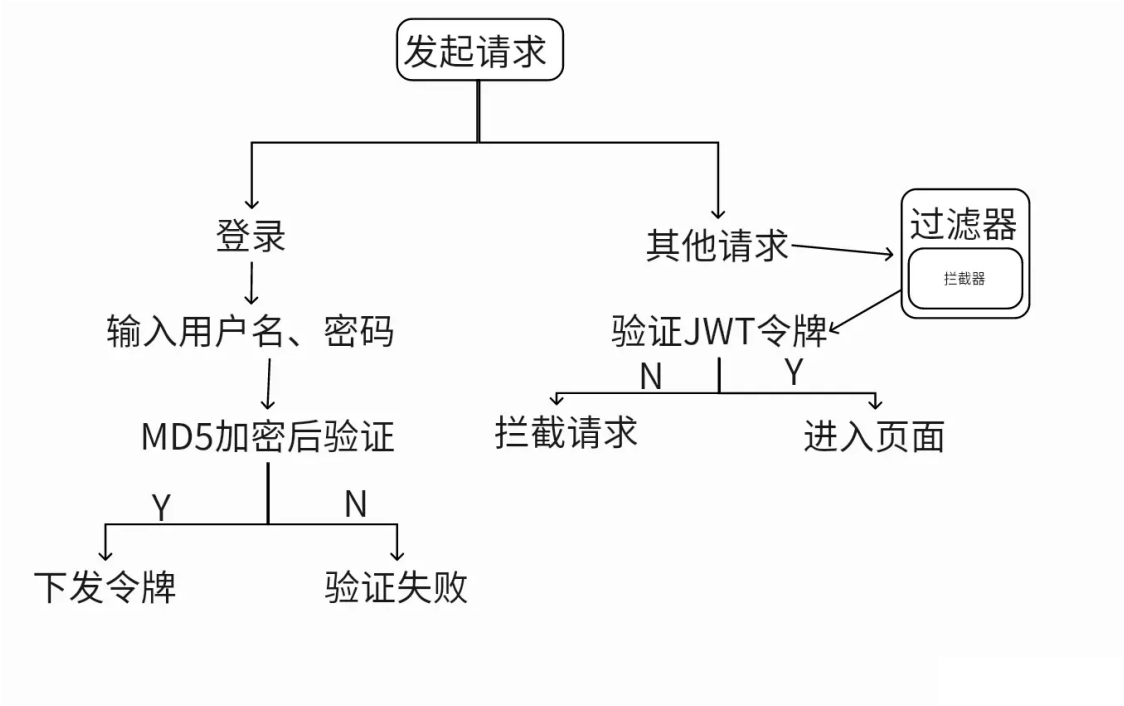


三、具体业务流程图

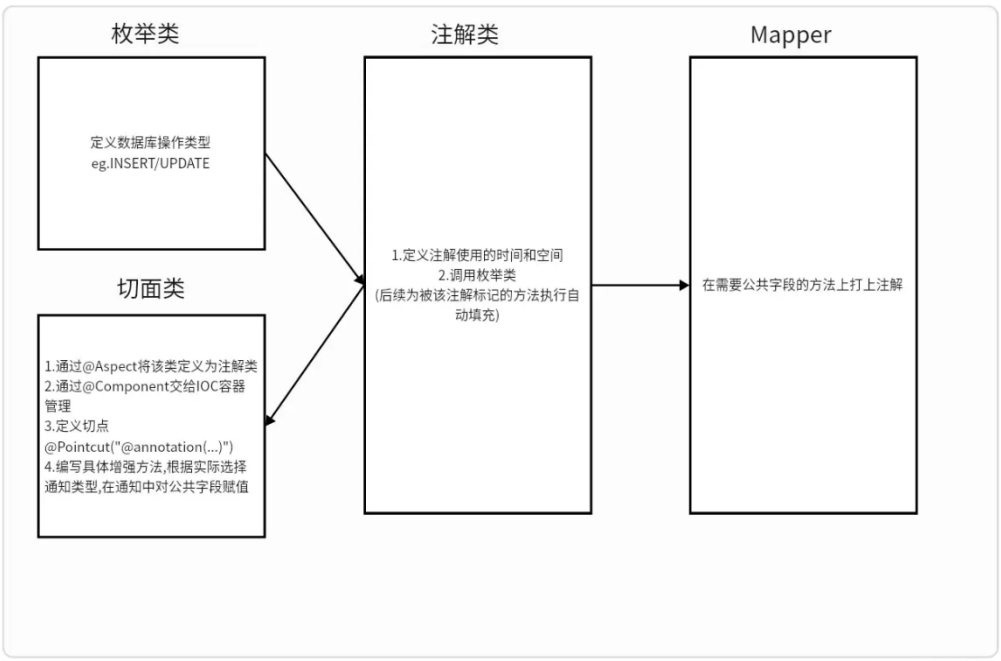
1. 拦截器



2. 登录 -- JWT

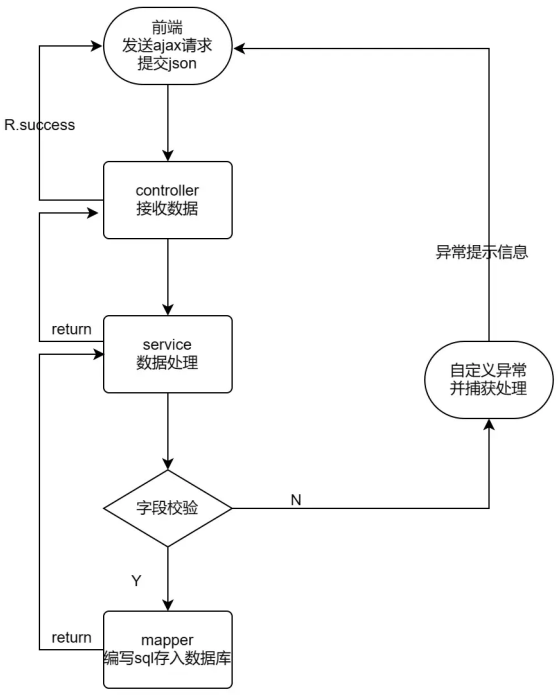


3. 公共字段填充—AOP 技术

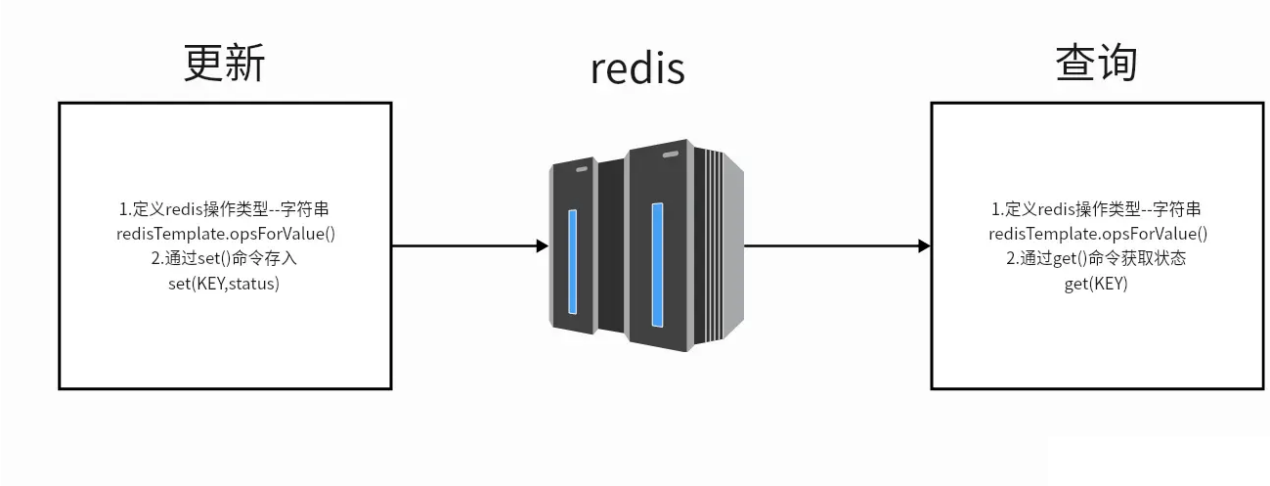


4. 新增

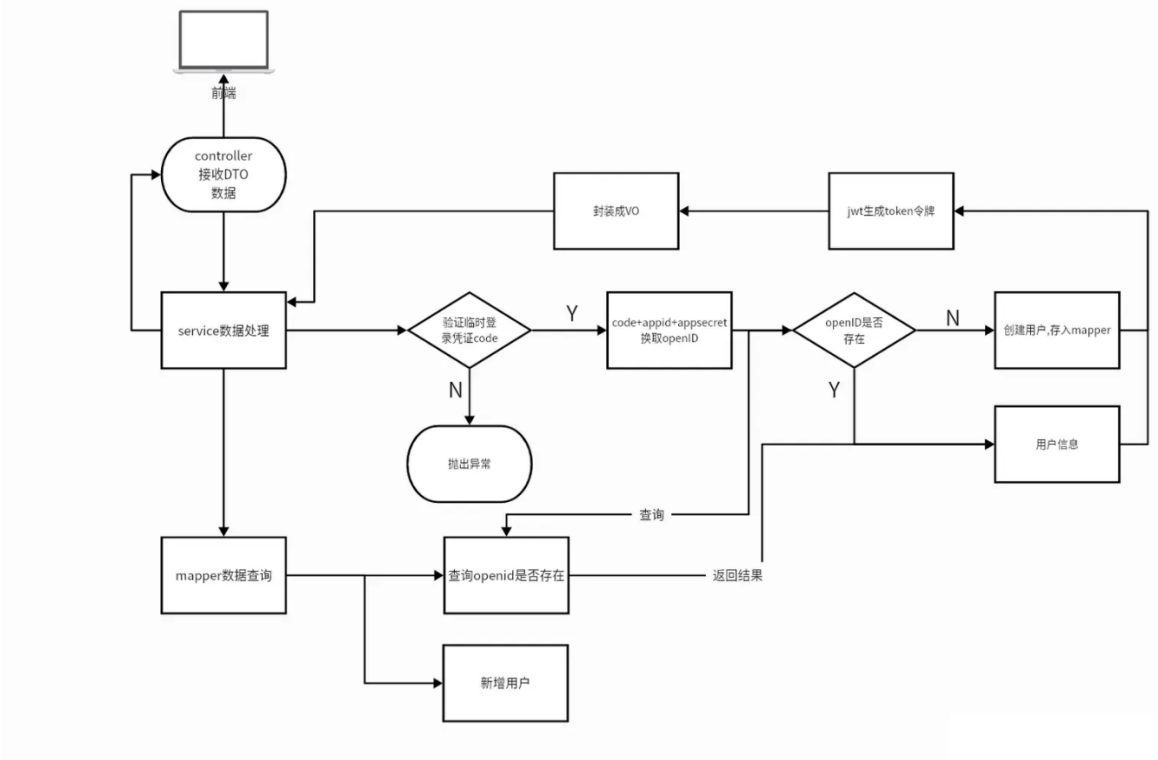
新增



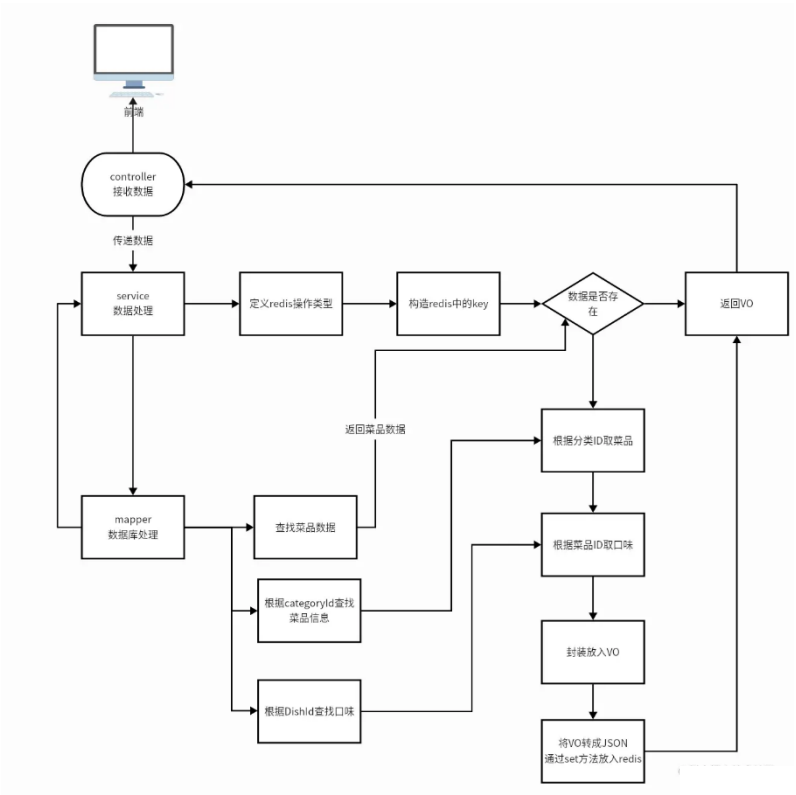
5.店铺营业状态—redis 缓存



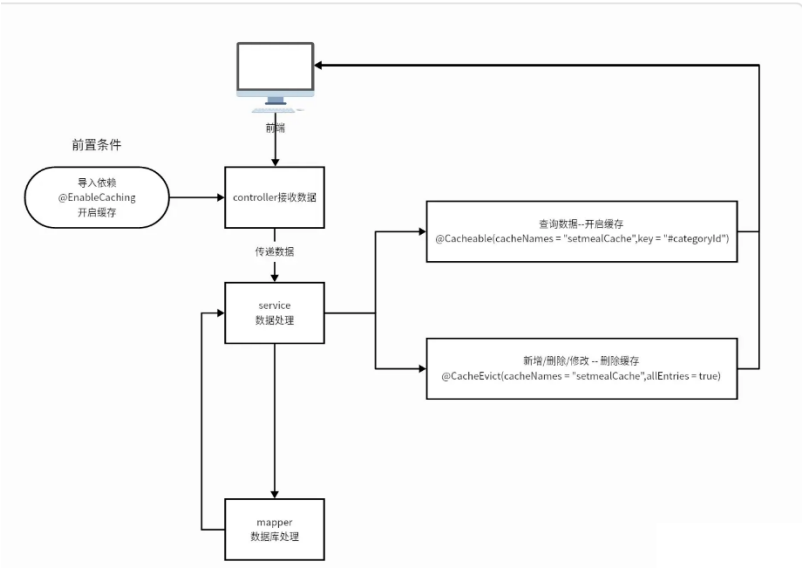
6.微信小程序登录



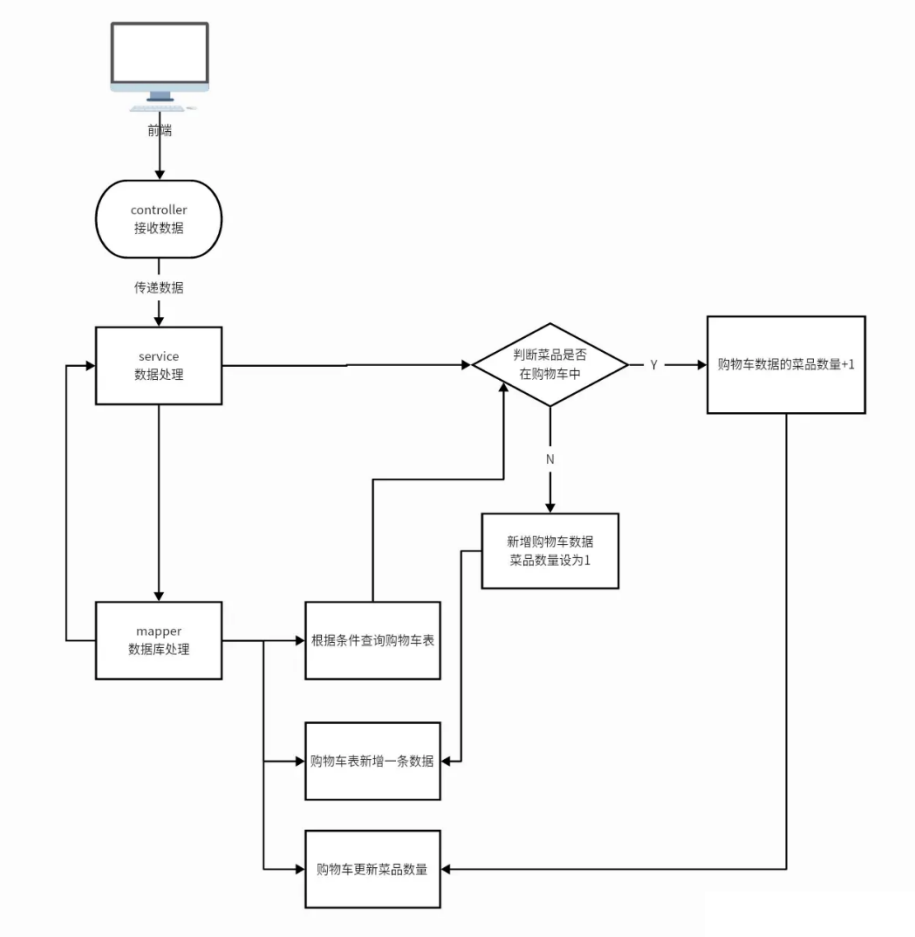
7.用户端查看菜品—redis 手动缓存



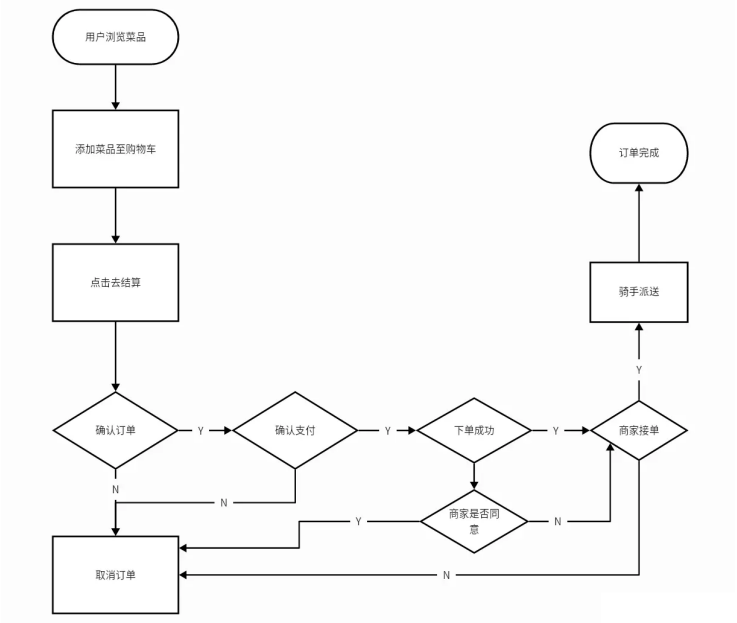
8.用户查看套餐—spring Cache 注解缓存



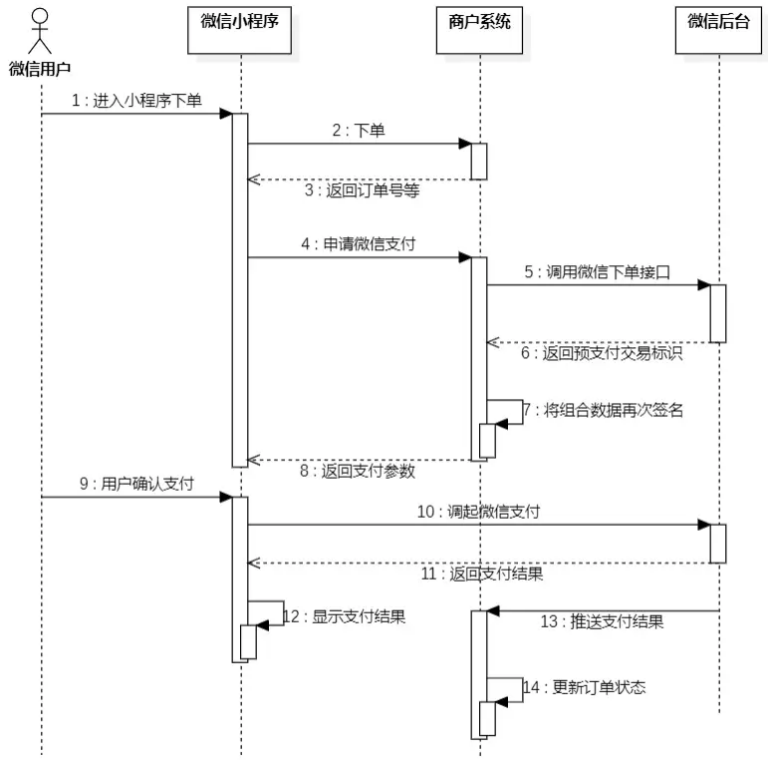
9.添加菜品/套餐至购物车


前端
controller
接收数据
传递数据
service
数据处理
判断菜品是否
在购物车中
Y
购物车数据的菜品数量+1
N
新增购物车数据
菜品数量设为1
mapper
数据库处理
根据条件查询购物车表
购物车表新增一条数据
购物车更新菜品数量

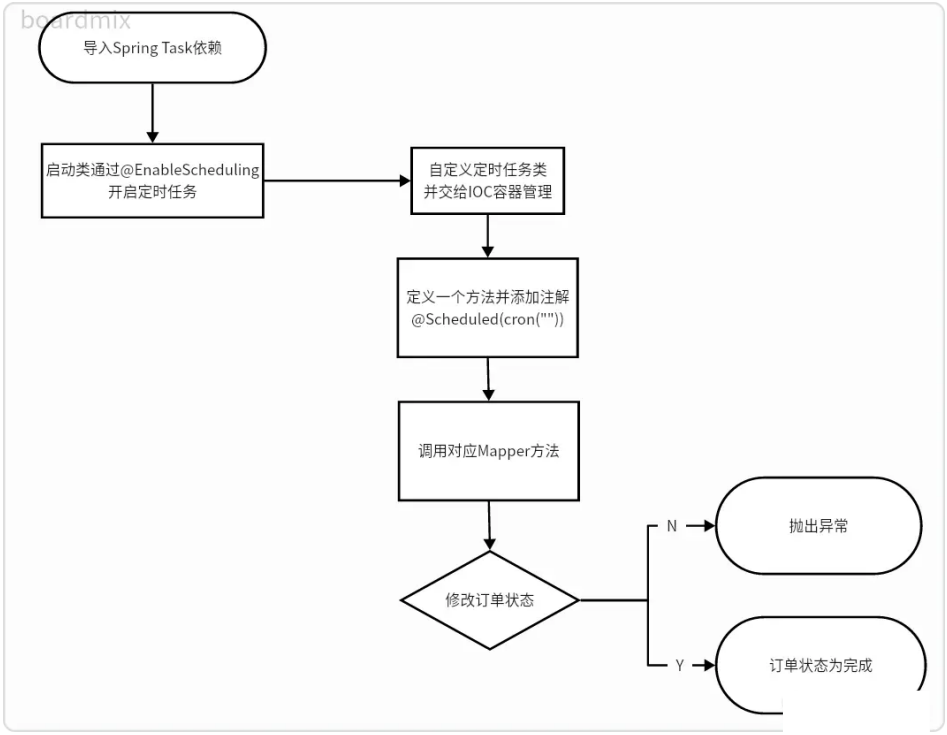
10. 用户下单


用户浏览菜品
添加菜品至购物车
点击去结算
确认订单
Y
确认支付
Y
下单成功
Y
商家接单
订单完成
骑手派送
Y
N
N
Y
商家是否同
意
N
取消订单
N

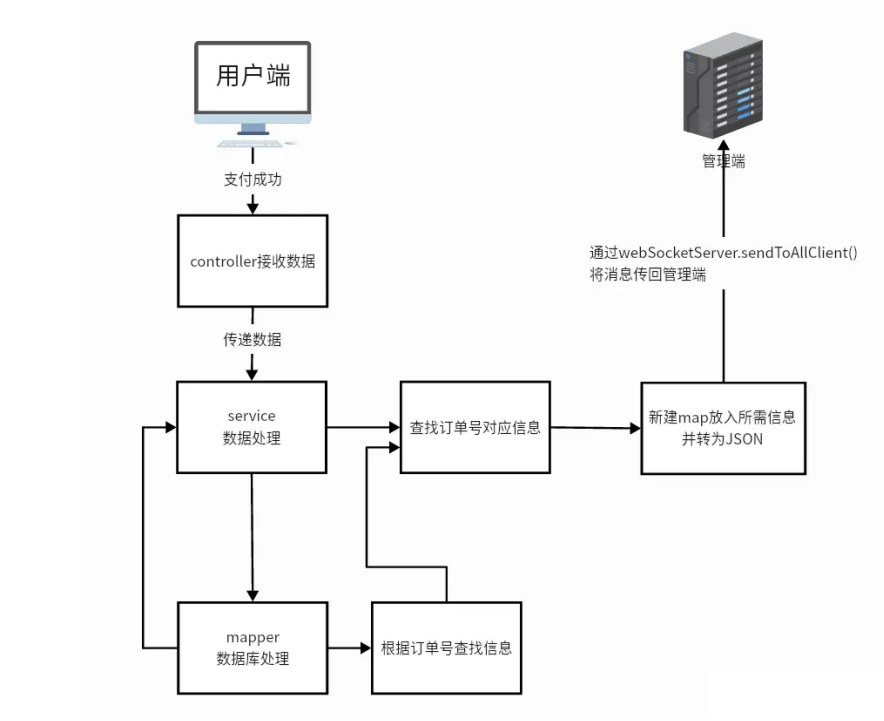
11.用户支付



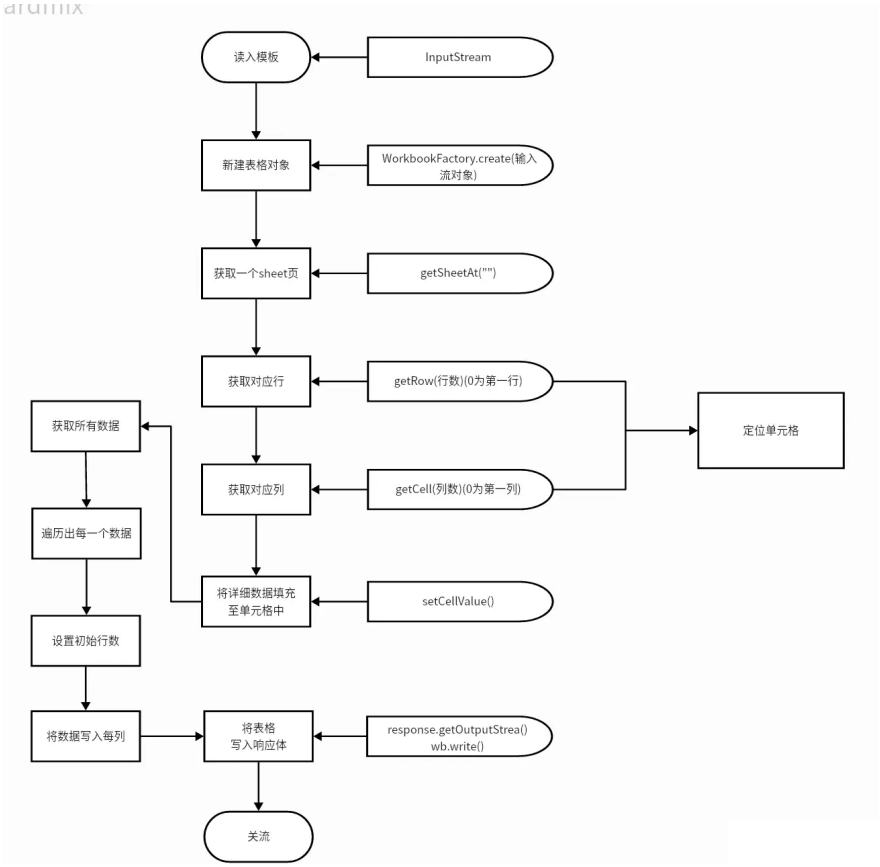
12.订单状态定时修改—spring task



13.来单/催单 - web socket



14.导出报表 - POI


读入模板
InputStream
新建表格对象
WorkbookFactory.create(输入流对象)
获取一个sheet页
getSheetAt("")
获取对应行
getRow(行数)(0为第一行)
定位单元格
获取所有数据
获取对应列
getCell(列数)(0为第一列)
遍历出每一个数据
将详细数据填充至单元格中
setCellValue()
设置初始行数
将数据写入每列
将表格写入响应体
response.getOutputStrea()
wb.write()
关流