

# 应用系统体系架构 — 作业 2

## 程序设计方案

### 消息格式

除去 Kafka 自身提供的header，我实现的消息格式为<String, String>型的 record

### Topic 配置项

Topic的配置采取Kafka默认配置

第一个为 topic1, 接收创建订单请求的消息，第二个为topic2，接受创建订单结果的消息（暂未实现将结果返回给前端），两个topic设置成相同的group\_id

### 订单处理结果呈现方式

1. 在前端工程中，使用 JavaScript 监听订单处理结果消息发送到的 Topic，然后刷新页面
2. 在前端发送 Ajax 请求获取订单的最新状态，后端接收到请求后将订单状态返回给前端去显示；
3. 采用 WebSocket 方式，后端的消息监听器类监听到消息处理结果 Topic 中的消息后，通过 WebSocket 发送给前端

我采用的处理方式：

在前端发送 Ajax 请求获取订单的最新状态，后端接收到请求后将订单状态返回给前端去显示；

实现效果如图：

欢迎来到EBook

退出登录

搜索你想要的订单

搜索

购物车

查看所有订单

Start date ~ End date

订单创建时间: 2022/10/8 20:22:17

总价 281.00

书名: 《Java编程思想》	作者: Bruce Eckel	单价: 91.2	数量: 2
书名: 《老人与海》	作者: 海明威	单价: 27.8	数量: 1
书名: 《高考英语 五年高考三年模拟》	作者: 曲一线	单价: 70.8	数量: 1

Wave's GitHub

2022-10-08 20:22:17.714 INFO 19108 --- [ntainer#0-0-C-1] c.w.b.service.impl.OrderServiceImpl : created order successfully  
userId: 21  
topic2 get message, create\_order\_response : CreateOrderResponse(status=ORDER\_ALL\_OK, id=65)

## 三种实现方式的优缺点：

### 1. 第一种

- 优点
  - 可以持续监听后端处理订单的情况并告知用户
- 缺点
  - 暴露后端的实现细节

### 2. 第二种

- 优点
  - 实现简单，前后端隔离较好
  - 在请求量小的时候可以做到即时交互
- 缺点
  - 每次请求都存在大量重复的请求头，存在较多I/O浪费
  - 每次请求需要重新建立连接，请求与回应花费较大

### 3. 第三种

- 优点
  - 浏览器和服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输。
  - 支持服务器推送消息
- 缺点
  - 服务器长期维护长连接需要一定的成本
  - 各个浏览器支持程度不一