应用系统体系架构 — 作业1

# 学号： 522031910213 姓名： 朱涵 得分：

# 本次作业回答如下：

## 

## 在Controller和Service层都使用了session作用域的scope。

## Service层使用session的原因：

## - 用户会话保持的时间是一个状态 需要有状态的服务 使用默认的单例的话 每个用户获取到的service都是同一个实例 无法为每一个用户维护不同的状态 所以显然不用单例模式 同理的 如果选用prototype 同一个用户每一次调用会获取到不同的服务 每次都会获得一个新的计时器 自然也不符合要求

## - 而既然要求一个会话维护一个状态 那么session作用域是最好的选择 即一个用户如果有新的会话 调用的service就会新创建一个实例 做到一次会话对应一个计时器对应一个状态 实现为每一次会话记录持续时间

## Controller层使用session的原因：

## - 如果服务层使用了session作用域 而控制层却使用单例作用域 因为控制层只被创造一次 也就只进行一次依赖注入 也就是说控制层实际上只会调用同一个服务层实例 那么服务层的session作用域也就无效了 同理如果是prototype作用域 会出现多个controller实例注入了同一个服务层实例的问题 而且也没必要每次请求都创建新的控制层实例 所以也不妥

## - 和服务层一样使用session作用域是最好的选择 同一个会话的所有请求对应控制层 这个控制层又会对应注入一个服务层

## - 当然配置代理也是可以的 比如让单例的控制层不断重新注入新的服务层实例 这就需要权衡利弊了

## 下面是在两个浏览器内登录登出的结果 一个浏览器立马登出 只有2s 另一个稍等一会 41s

## 

## 如上图是控制台输出的结果 上半部分是下订单成功的信息 第一行是控制层接收到前端发送的请求 第二行是AddOrder监听器接受到控制层发送的消息 第三行则是FinishOrder监听器监听到AddOrder监听器完成订单添加操作后发送的消息

## 同理 下半部分的最后一行是AddOrder监听器进行订单添加操作中途发生异常情况下 FinishOrder监听器接受到的异常消息

## 简单讲一下代码部分 主要分为kafka配置 控制层 监听器三块

## - Kafka在spring的配置主要是服务url和topic的新建 在这里建立两个topic：AddOrder和FinishOrder 分别用于存储下订单的消息和处理订单完成的消息

## - 控制层需要注入kafkaTemplate进行消息的发送 因为监听器接受到消息后调用服务层时 上下文中存储的session已经没有了 所以需要在发送消息前进行鉴权获取用户uid 然后把uid也封装到消息里 表示这个订单是由这个uid的用户所下的

## - 监听器需要配置两组消费者 一组负责监听AddOrder 收到消息后调用服务层的addOrder 把处理结果（成功还是失败）封装为消息后转发给FinishOrder这个topi；另一组监听FinishOrder 输出处理订单的结果

## 后续使用websocket改装 只需要修改addorder监听器内的操作即可 很方便