Spring Cloud Stream 原理与扩展

编程语言发展趋势

提示：偏个人主观

机器语言（低级）

01010 = 二进制语言

汇编语言（低级）

Intel ex mov

AMD

举例：

- 内存屏障 `volatile`
- 线程实现
- 协程实现

面向过程编程（中级）

C、Pascal、Fortran

以函数为计算单元，关注输入参数，返回结果

默认情况是串行

优势：高度静态语言，编译器优化居多

C 语言类型大致两种：

- 简单类型 - int、char、void*
- 复杂类型 - [], struct、union、enum

C - 结构（`struct`）

- 只有成员
- 没有函数

面向对象编程（高级）

核心技术：多态

Smalltalk、C++、Java 等

所有运行实体都对象，均继承 java.lang.Object

实体单元组织形式类，以 java.lang.Class

- 成员
  - 构造器
  - 方法
  - 字段

设计原则：一切都是对象

基本特性：来自于生物学

- 封装（组织）
- 继承（遗传）
- 多态（变异）

优势：便于理解

不足：对象树的方式，嵌套太深，多态不易于理解

设计模式：GoF23

- 组合：既保留特性，不想存在乘次
- 代理：别人来实现
- 装饰器：别人来主导，我来打擦边球
- 策略模式：利用多态，一种契约，多种路径
- 适配器：它方的数据，是适应现在的格式（不同层次关系）

Java 面向对象：除了执行单元（类、接口），异常也是如此

面向切面编程（高级）

核心特性：拦截

AOP 两种类型：

- 静态 - 通过适配器模式、代理
- 动态 - 调整字节码动态植入一段代码，形成回调
    - Java 动态代理
    - Java 字节码提升
        - Java ASM
        - CGLib
        - AspectJ

面向声明式编程（高级）

核心特性：简化理解（实现）

实现来源：第三方框架、容器

JavaDoc、Java 注解（Annotation）、C# 属性（Attribute）

JavaDoc

- Java 文档 HTML
- Doclet
- Maven 2.x 插件属性设置

Java 注解（Annotation）

- 源码时运用
  - Java 文档 HTML
- 编译时运用
  - Java 6+ `@Override`
  - `@SuppressWarnings`
- 运行时
  - 反射

面向函数编程（现代）

核心特性：简化语法

实现来源：编程语言、扩展语言（Reactive、Groovy）

Java 8 Lambda、ES JS、NodeJS、TypeScript

主要趋势

- 面向过程编程（思维固化、始祖）
- 面向对象编程（维护状态、基石）
- 面向函数编程（未来趋势）
- 面向 Reactive Streams 编程（未来大趋势）

- 面向分布式编程
  - Streaming
  - Batch Processing
  - Distributed Function

2014年说的微服务服务，号称兼顾 Dev（开发）和 Ops（运维），但实际大多数人都在讲 Dev，包括框架 Spring Boot / Spring Cloud，方法论 DDD，包括容器化 Docker，强调开发

2016年 CNCF 鼓吹云原生（Cloud Native），推广 K8S 等运维平台或工具，偏向运维

微服务开发模式 = Spring Cloud + DDD + Docker + K8S + 其他...

Spring Cloud Stream 与 Kafka 整合

> 重要注意事项：
>
> 1. 尽管 Spring Cloud Stream Binder 中存在 Kafka 的整合，然而 Spring Kafka 和 Spring Cloud Stream Kafka 在处理数据生产和消费是存在差异，因此不要混用
> 2. 当 Spring Cloud Stream 发送消息包含头信息时， Kafka Deserializer 实现方法回调时不会予以处理
> 3. 通常业务逻辑可以使用 @StreamListener 来监听数据（主体、载体），如果是需要更多头信息，需要 SubscribableChannel 来实现
> 4. @StreamListener 与 SubscribableChannel 实现是存在差异的，以 Kafka 为例，SubscribableChannel 会触发 Kafka 自定义反序列化

> 5. 如果同一个应用同时使用 @StreamListener 和 SubscribableChannel 时，两者会轮流处理
> 6. Spring Cloud Stream Kafka 是将对象序列化成 JSON，通过 JSON 反序列化成对象（不经过自定义 Kafka 序列化/反序列化实现）

编程模型

1. 原生（扩展）Message 实现 - Kafka、RabbitMQ、RocketMQ
2. 基于注解实现 - `@StreamListener`
3. 基于 API 实现 - `MessageChannel`、`SubscribableChannel`

两个重要注解

输入注解 @Input

@Input 注解对应 SubscribableChannel

输出注解 @Output

@Output 注解对应 MessageChannel

相同点

- 两种注解均屏蔽了具体 Stream 实现（Binder），比如看不出是 Kafka 还是 RabbitMQ，开发人员更关注于自己的业务实现
- `@Input` 和 `@Output` 中的 value() 属性均为 Channel（管道）名称，不允许重复名称（因为不允许重复Bean），通过定义它的 destination 来指定对应 topic

Spring Cloud Stream 对微服务影响

Spring Cloud Stream 作为 Spring Cloud Data Flow 基石

Spring Cloud Stream 也是 Spring Cloud Bus 核心依赖

Spring Cloud Bus 是 Spring 分布式事件

Spring Cloud Stream 是高度统一的流式处理分布式框架