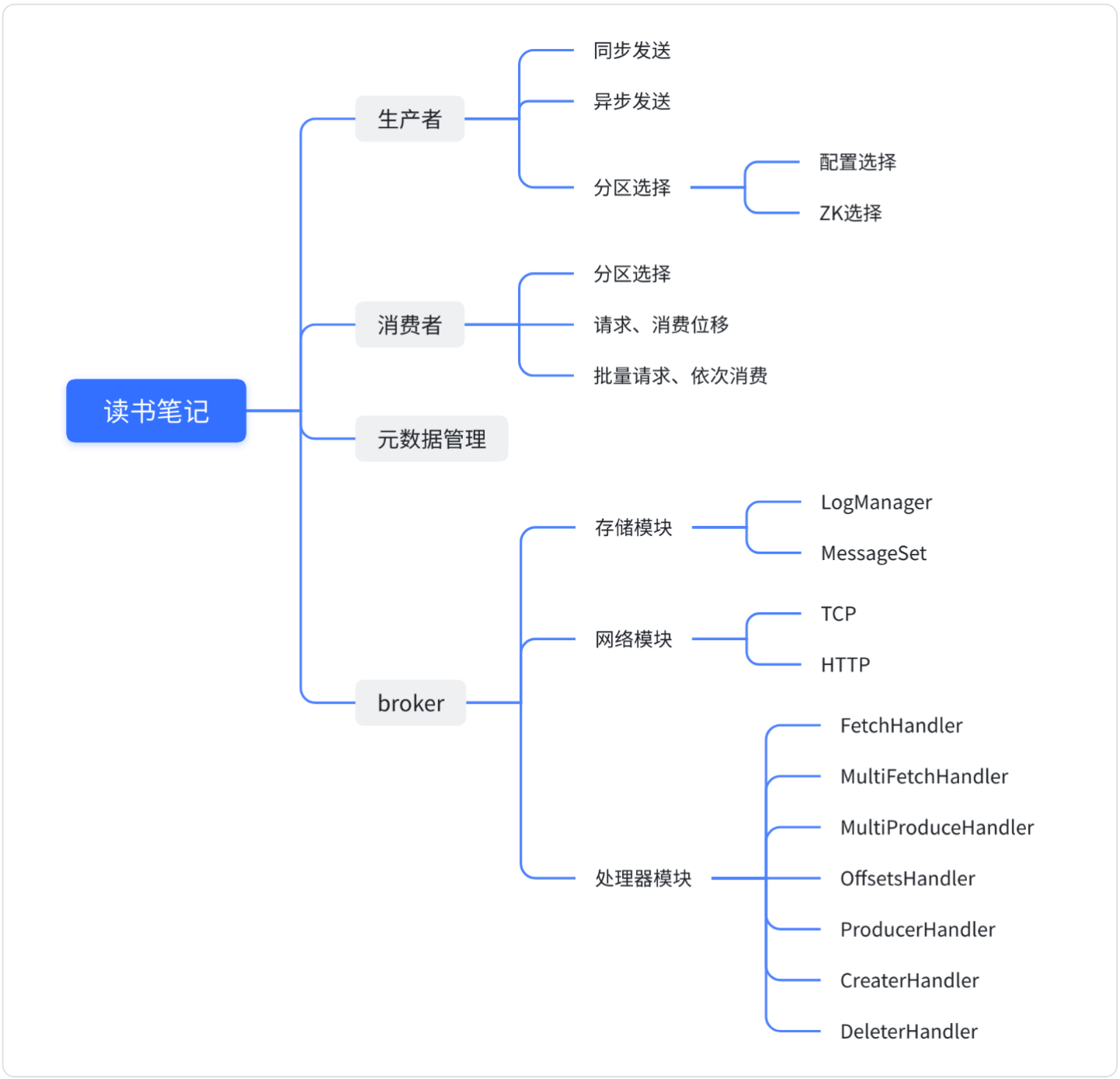
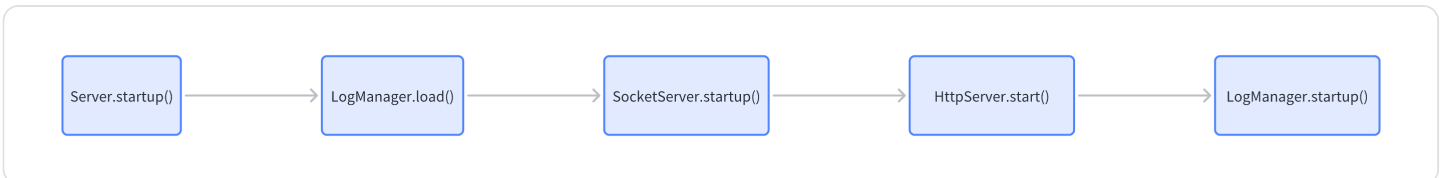
# jafka学习笔记




## Broker

启动流程：



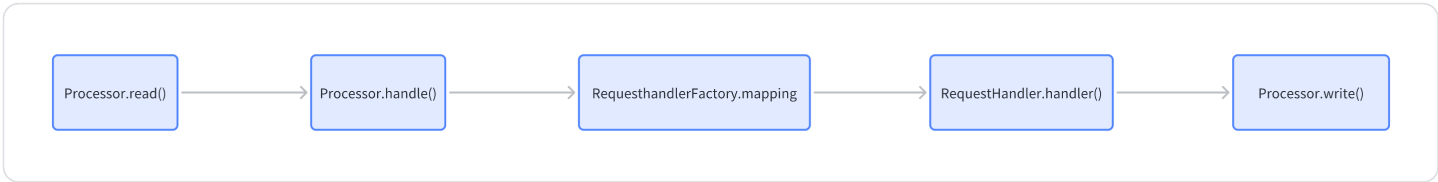
## 网络-接入层

### TCP处理模块

基于原生 NIO Reactor模式实现，分为：

- Acceptor：监听 TCP 连接，并将 accept 事件派发给Processor；
- Processor：处理TCP accept、read、write、close 事件；

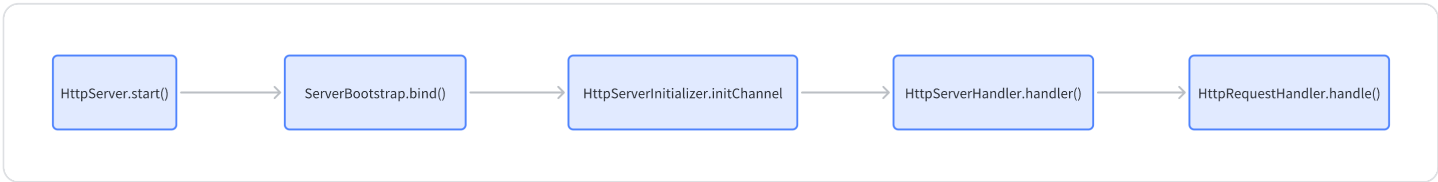
Processor 读取客户端数据后，解析请求包，得到请求类型、数据后调用 RequestHandlerFactory.mapping 方法得到 handler，然后执行 handler 方法，获取请求结果，并返回。




handler处理器见下。

### HTTP处理模块

基于 Netty 实现 HTTP 服务器，只支持消息 Produce 一个接口（参数在HTTP Header中）。




### 处理器

不同于HTTP模块只处理Produce一种请求，TCP模块处理多种请求：

```
@Override
public RequestHandler mapping(RequestKeys id, Receive request) {
    switch (id) {
        case FETCH:
            return fetchHandler;
        case PRODUCE:
            return producerHandler;
        case MULTIFETCH:
            return multiFetchHandler;
        case MULTIPRODUCE:
            return multiProduceHandler;
```

```
        case OFFSETS:
            return offsetsHandler;
        case CREATE:
            return createrHandler;
        case DELETE:
            return deleterHandler;
    }
    return null;
}
```

包括：

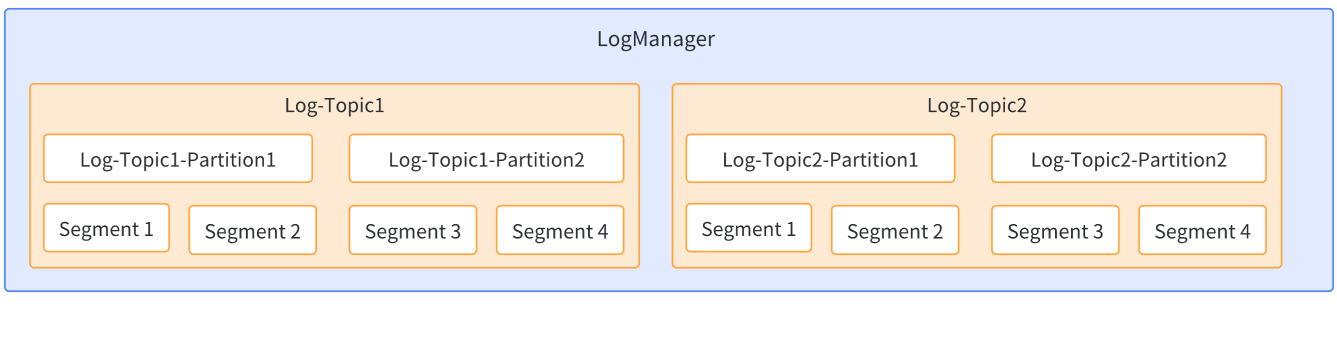
- fetch：请求消息；
- produce：生产消息；
- multiFetch：请求多个topic消息；
- multiProduce：生产多个topic消息；
- offsets：请求 offset 偏移；
- create：新建 topic；
- delete：删除 topic；

整个 handlers 体系最核心的是 LogManager 组件，负责消息查询、存储等核心功能。

Zookeeper是可选的，可直接在内存中注册，而生产者、消费者则从配置文件中读取。

## 存储-持久层

Log表示一种存储方式，即**追加写**，而不是说将消息当做日志存储。




- LogManager：消息存储管理器，与 handlers 直接关联，用于 topic 创建、删除，消息生产、消费；
- Log：某个 topic 下的一个分区，比如 Log-Topic1-Partition1，可以理解为一个目录；

- LogSegment：分区下消息按照段来存储，最后一个段是可写的，其它的段都是可读的，每个段可以理解为一个文件；
- topic、分区元数据都注册到ZK中，异步任务线程会不断的刷新ZK元数据；
- 刷盘调度器按照时间策略来将日志数据刷盘；
- 清理调度器按照过期时间来清理日志数据；

Segment 下通过 MessageSet 来存储消息数据，查找 Message 时，先通过 offset 来找到对应的 LogSegment，然后再从 MessageSet 中读取 Message：

```
1  public MessageSet read(long offset, int length) throws IOException {
2      List<LogSegment> views = segments.getView();
3      LogSegment found = findRange(views, offset, views.size());
4      if (found == null) {
5          if (logger.isTraceEnabled()) {
6              logger.trace(format("NOT FOUND MessageSet from Log[%s], offset=%d, l
7          }
8          return MessageSet.Empty;
9      }
10     // read message set from the found segment
11     return found.getMessageSet().read(offset - found.start(), length);
12 }
```

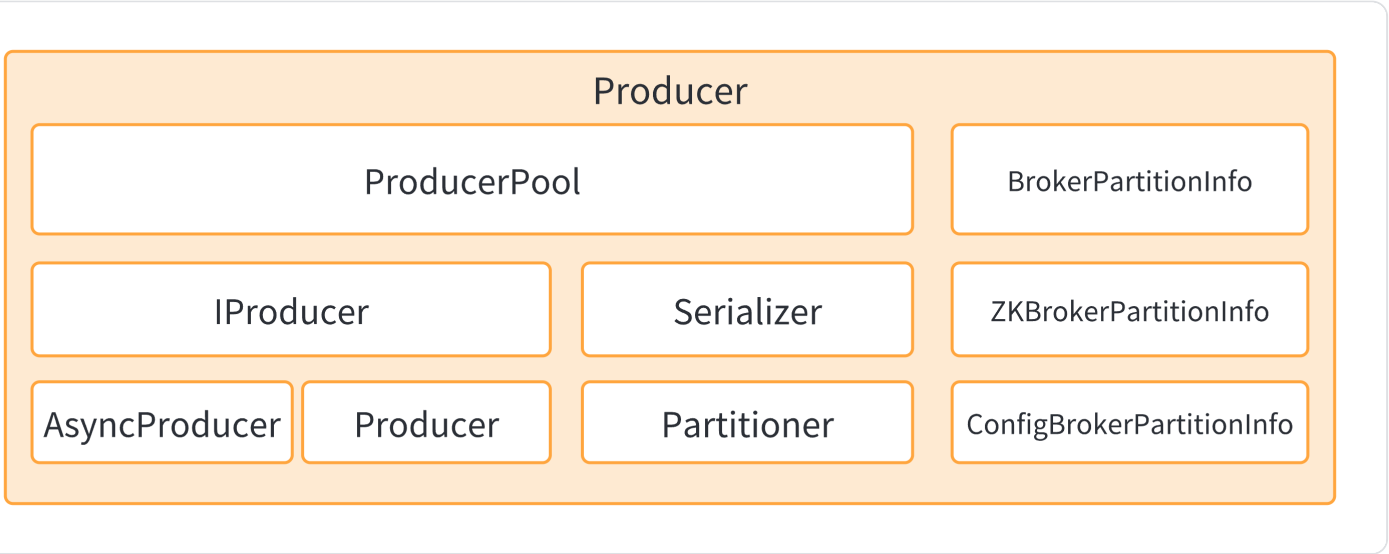
MessageSet 是 Message 和 Offset 的集合：

- Message：生产、消费消息数据；
- Offset：消息偏移；

Offset 可以视为 Message 在存储模块的唯一标识，也是客户端请求、消费 Message 的标识。

## 生产者

生产者可选择从ZK中获取主题、分区注册信息，或者是从配置中获取。




- ProducerPool：生产者池，比如每个topic一个生产者；
- IProducer：生产者抽象，同步、异步生产者；
- Serializer：消息序列化、反序列化；
- Partitioner：分区选择抽象，比如随机分区、哈希分区等；
- BrokerPartitionInfo：主题分区信息，支持从配置文件、ZK中读取；

## 消费者

消费者只支持从 ZK 中获取 broker 信息，ZookeeperConsumerConnector 主要做如下三件事：

1. 连接 ZK，并获取可消费主题分区元信息；
2. 新建 Fetcher 请求主题消息；
3. 开启后台偏移提交任务，offset 存储在 ZK 中；