Spring响应式编程概述

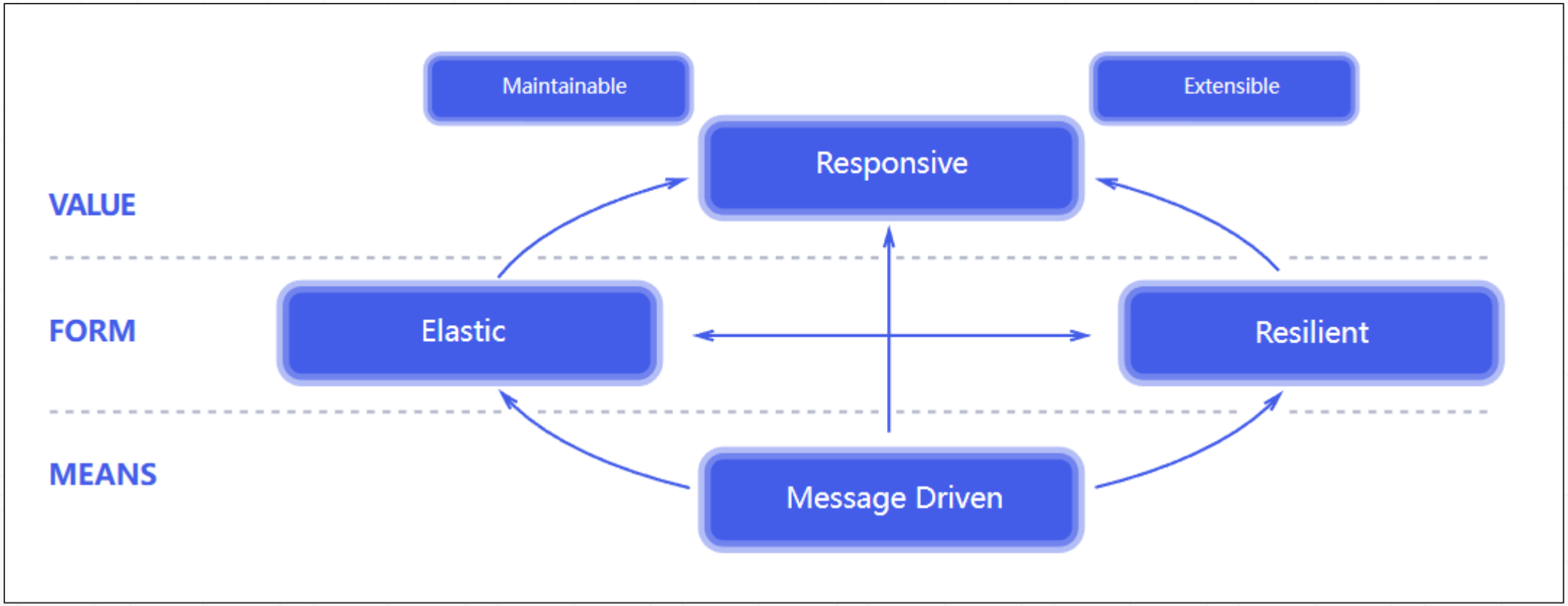
北京理工大学计算机学院
金旭亮

什么是响应式编程?

- 一种新的编程范式
- 异步非阻塞的运行模式
- 基于事件响应和消息驱动
- 采用函数式编程风格
- 支持背压（Back Pressure）

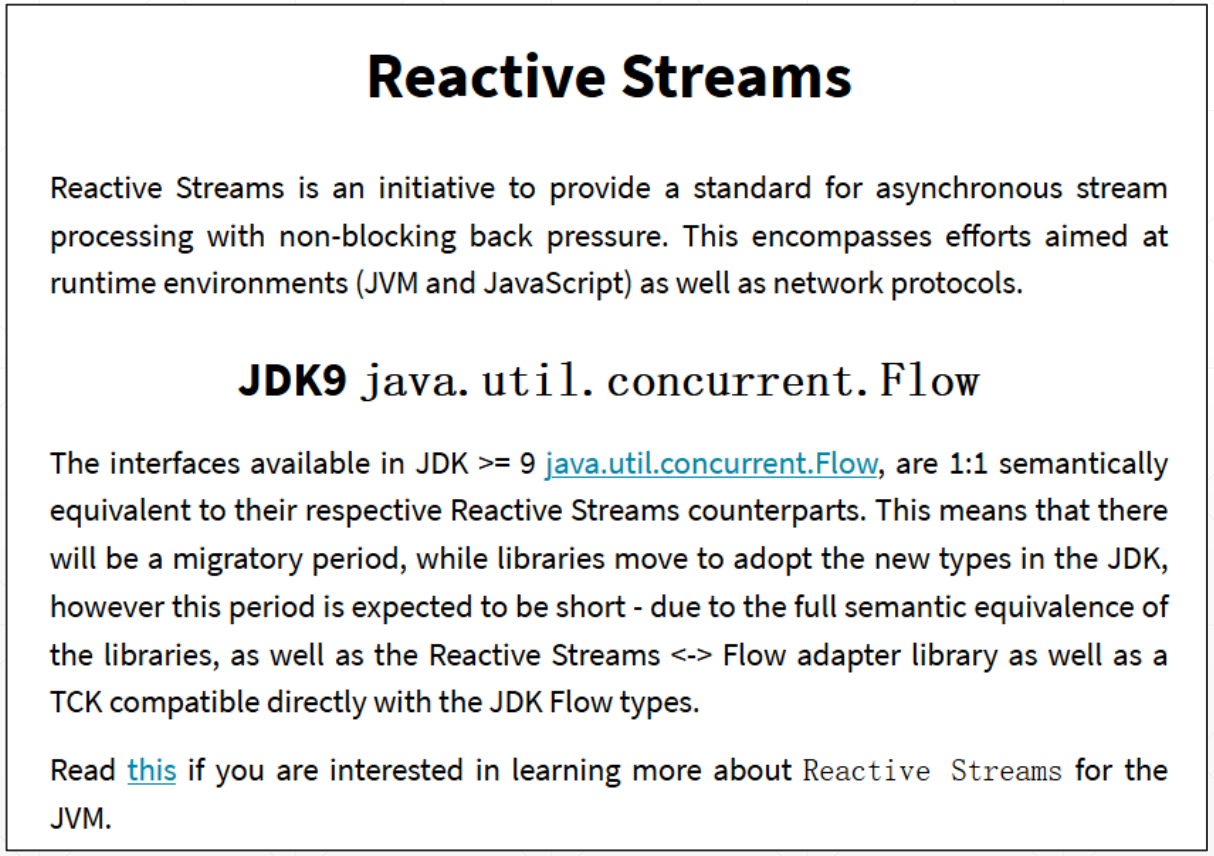
响应式编程宣言

https://www.reactivemanifesto.org/



响应式流规范及具体实现

http://www.reactive-streams.org/



Reactive Streams

Reactive Streams is an initiative to provide a standard for asynchronous stream processing with non-blocking back pressure. This encompasses efforts aimed at runtime environments (JVM and JavaScript) as well as network protocols.

JDK9 java.util.concurrent.Flow

The interfaces available in JDK >= 9 java.util.concurrent.Flow, are 1:1 semantically equivalent to their respective Reactive Streams counterparts. This means that there will be a migratory period, while libraries move to adopt the new types in the JDK, however this period is expected to be short - due to the full semantic equivalence of the libraries, as well as the Reactive Streams <-> Flow adapter library as well as a TCK compatible directly with the JDK Flow types.

Read this if you are interested in learning more about Reactive Streams for the JVM.

响应式编程是一种编程范式，或者说是一种编程思想，它需要统一的规范，并且提供“开箱即用”的具体实现。这个规范，就是“响应式流（Reactive Streams）”。

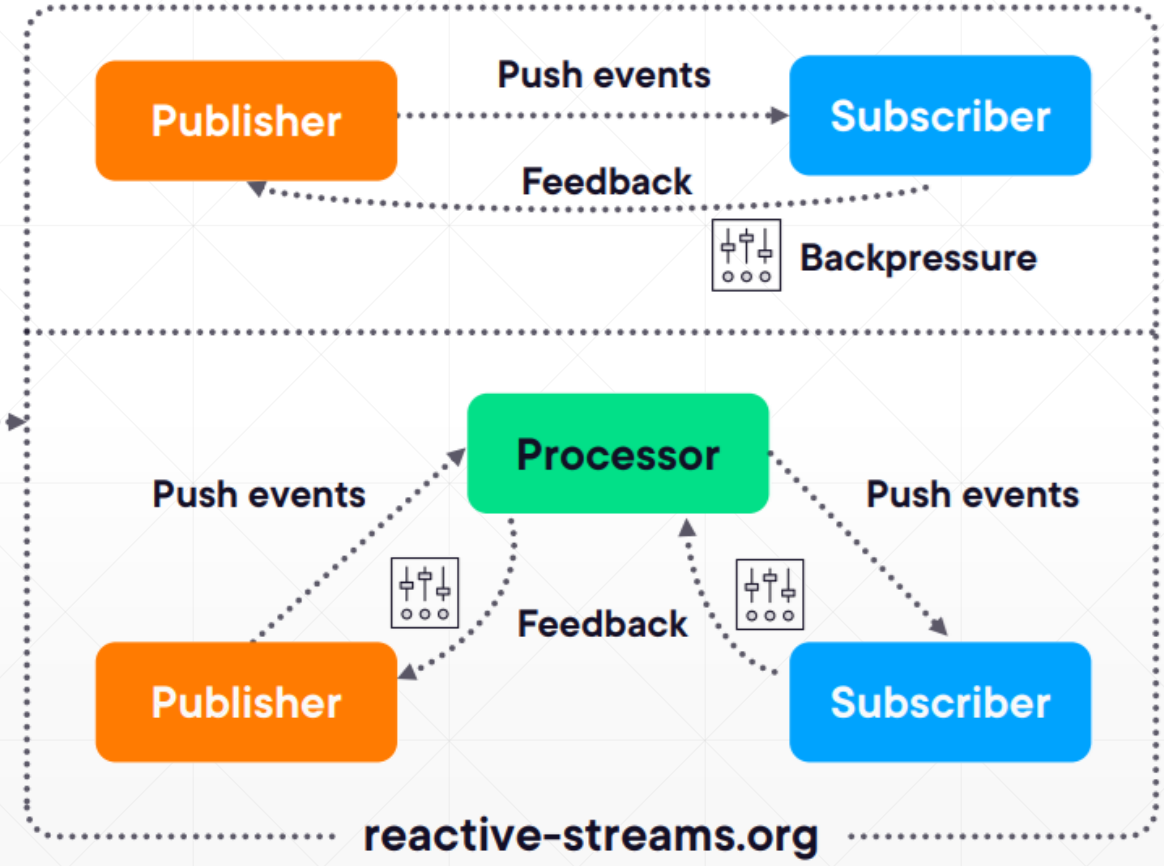
Java 9中对此宣言进行了响应，在JDK中引入了相关的接口和类型，开源社区中也积极地响应号召，推出了各种响应式编程框架，其中，RxJava（Netflix）、Vert.x（ReadHat）和Project Reactor（Pivotal）就是非常著名的几个响应式开发框架。

响应式编程与响应式流

响应式编程仅仅只是一种理念和原则，
响应式流则是一种规范和技术实现方案，
它主要定义了四个核心组件：
Publisher、Subscriber、Processer，
还有一个Subscription对象。

Reactive Streams

Reactive Programming



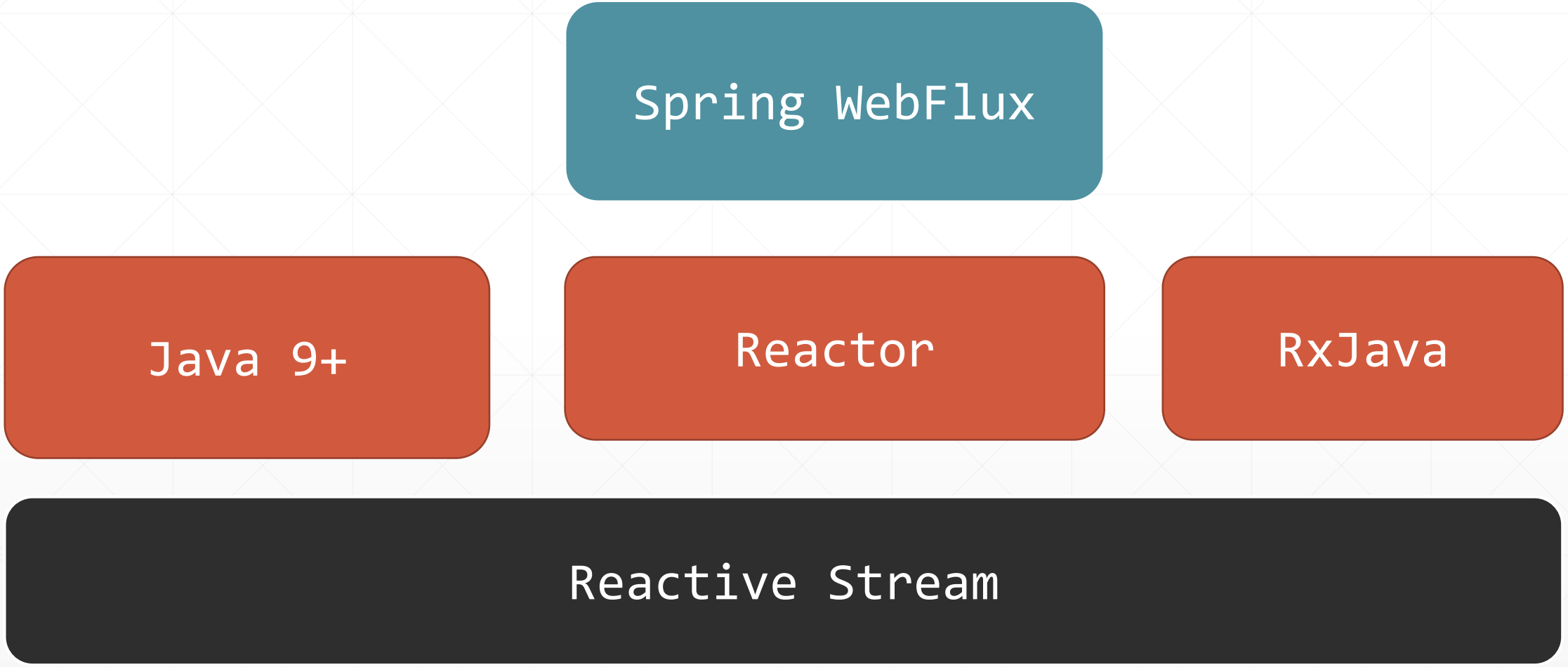
Reactive Stream的三个特性

asynchronous
（异步）

non-blocking
（非阻塞）

event-driven
（事件驱动）

Java领域主要的Reactive Library

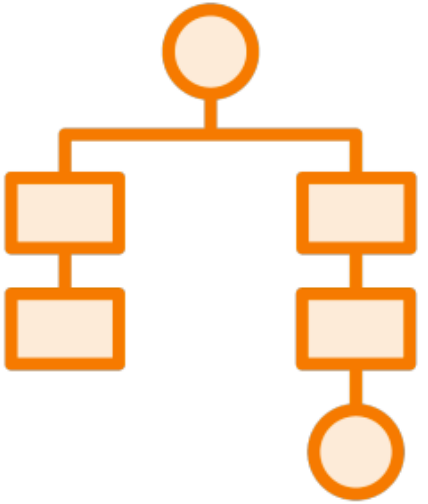


同步阻塞式的代码

响应式风格的代码

数据离散到达，数据处理方利用可组合的运算符函数，动态构建出一条异步非阻塞的“数据管线”，数据“即到即处理”。

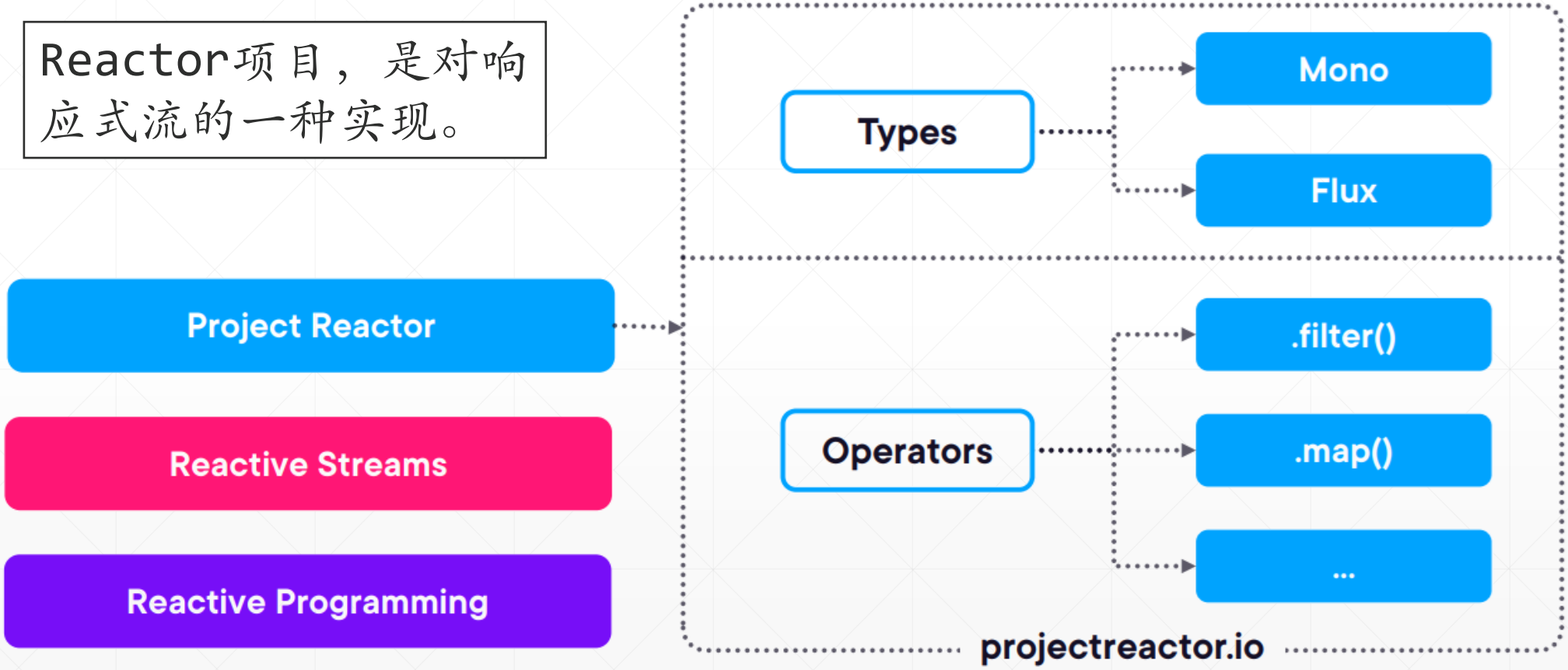
```
getData() {
    [          ]
    .filter( [          ] )
    .process( [          ] )
    .return( [          ] )
}
```



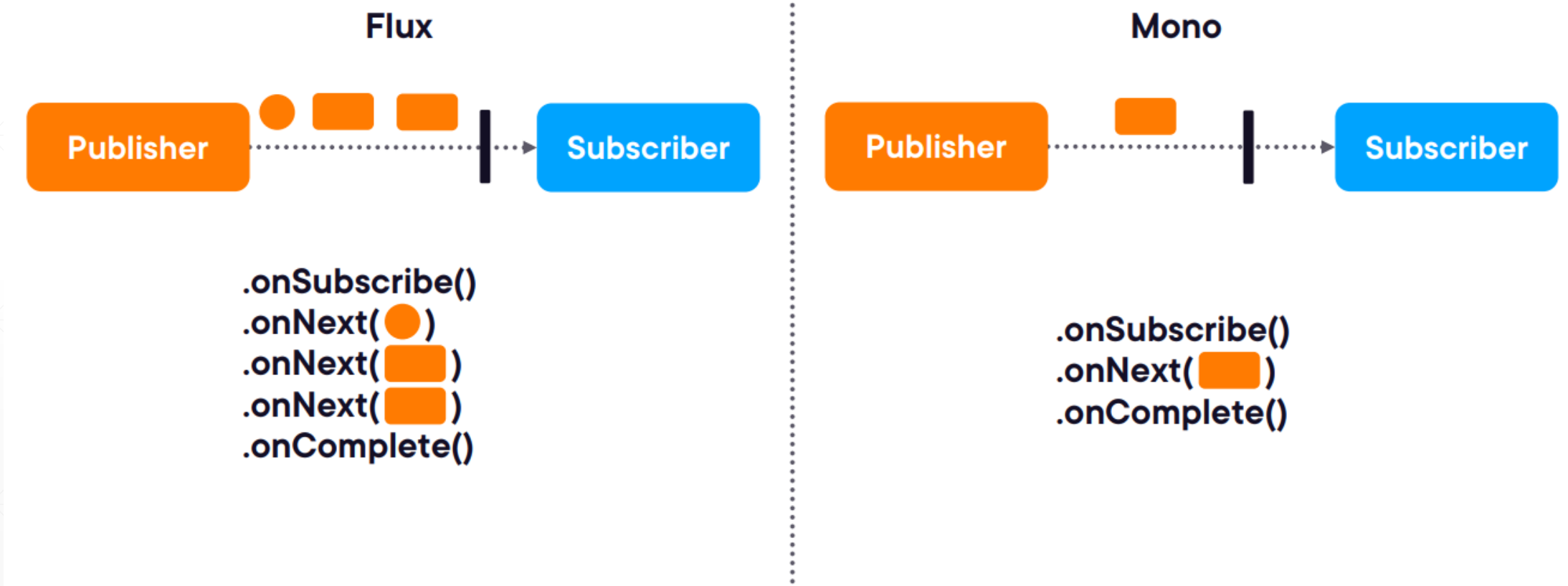
采用声明式的编程风格，代码简洁易读

Project Reactor

Reactor项目，是对响应式流的一种实现。



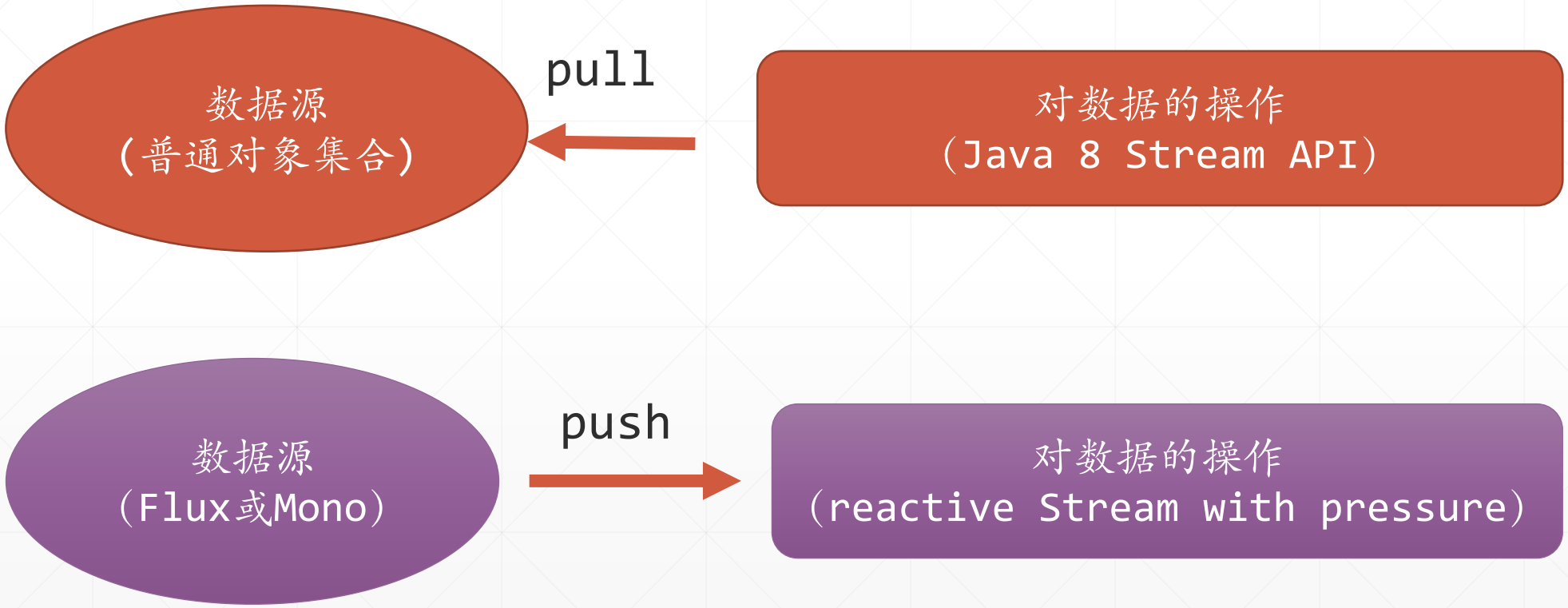
Project Reactor的核心类型



出错时……

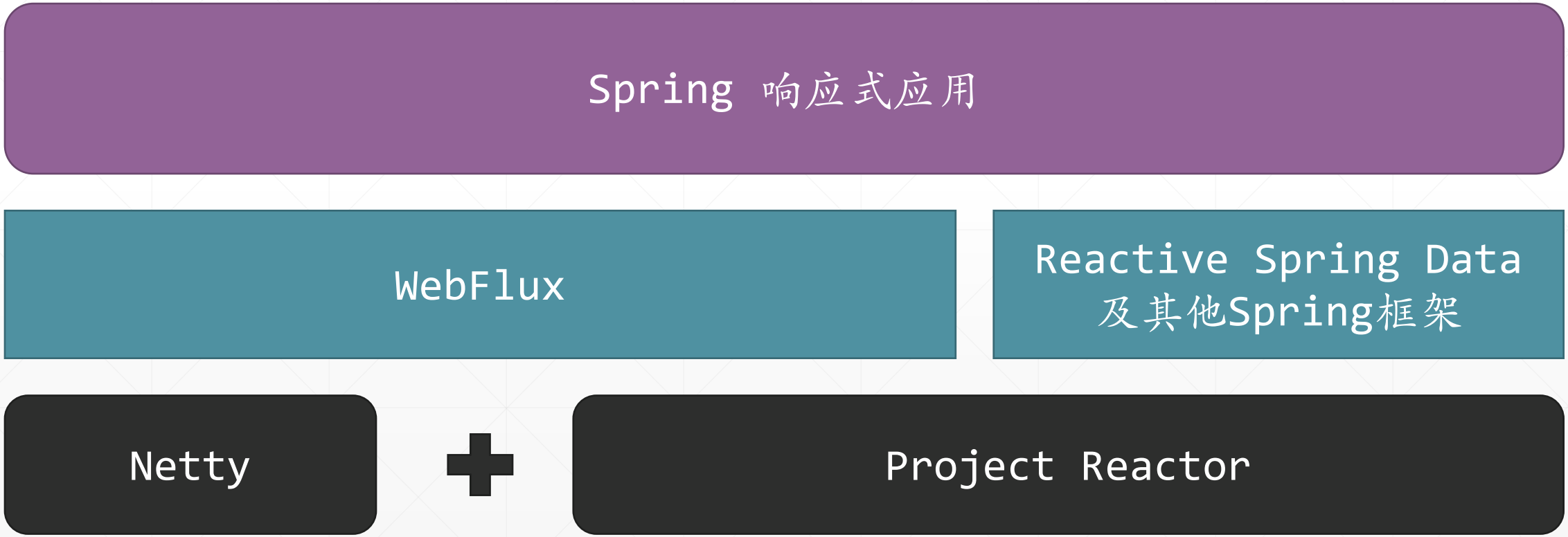
.onSubscribe()
.onNext( )
.onNext( )
.onError()

Reactive Stream vs. Java 8 Stream



Spring对响应式编程的支持

Spring WebFlux构建于Project Reactor之上，可用于开发异步非阻塞的微服务。



Spring WebFlux的两种编程模式

注解模式

```
@RestController
@RequestMapping(value = "/products")
class ProductController {
    private final ProductRepository repository;
    ...
    @GetMapping(value = "/")
    public Flux<Product> listProducts() {
        return repository.findAll();
    }
    ...
}
```

与传统编程模式高度类似，只是函数返回类型为Mono或Flux

函数式模式

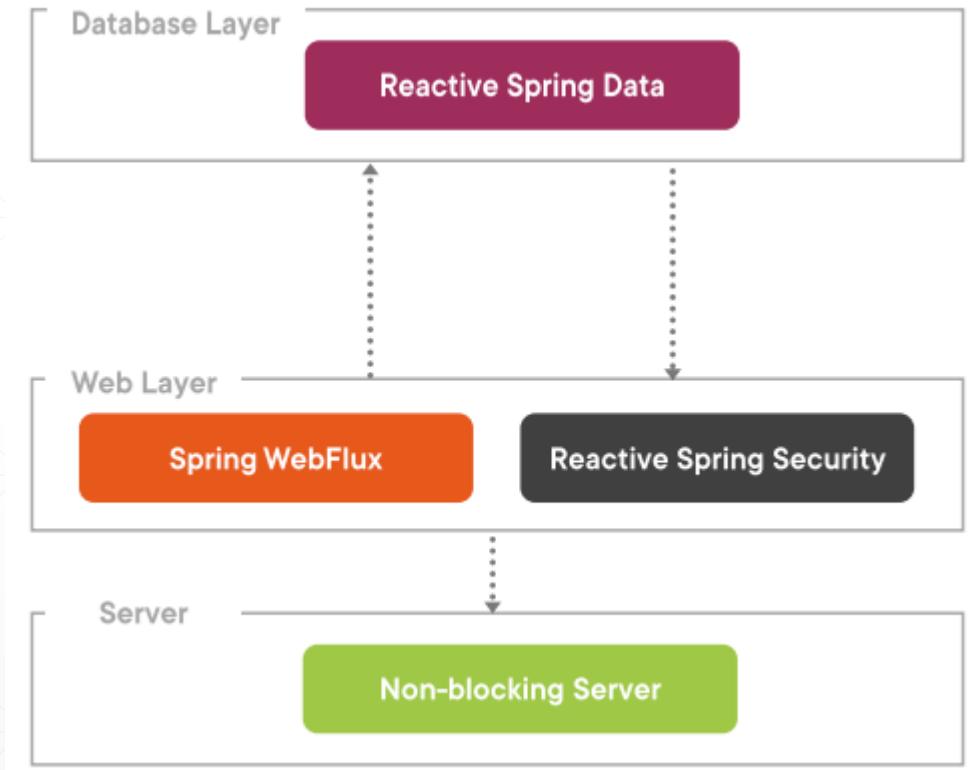
```
RouterFunction<ServerResponse> productRoute =
    route(GET("/product").and(
                                accept(APPLICATION_JSON)),
                                handler::listProducts
                               );

...

public Mono<ServerResponse> listProducts(ServerRequest request) {
  Flux<Product> products = repository.findAll();
  return ServerResponse.ok().contentType(APPLICATION_JSON).body(
            products, Product.class);
}
```

学习曲线陡峭一些，需要熟悉相关的组件用法。

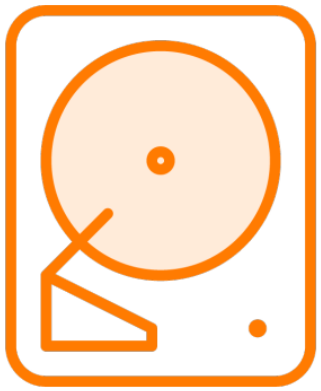
Spring响应式技术栈



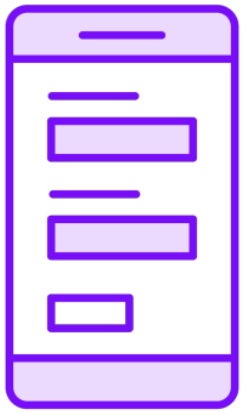
JDBC演化为R2DBC（Reactive Relation Database Connection）

支持响应式存取的NoSQL数据库

响应式编程适用的场景



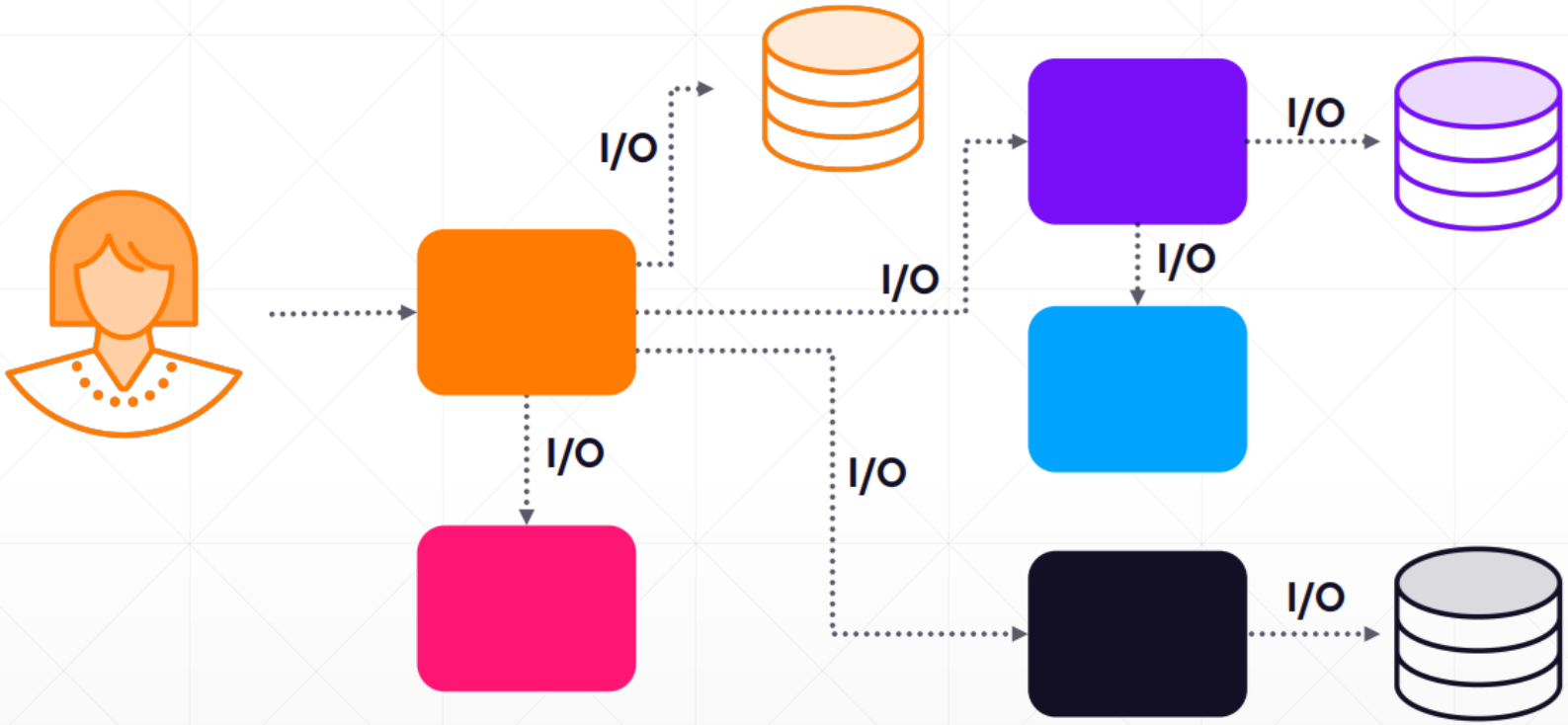
有大量IO操作的场景，比如采用微服务架构的Web后端应用



有大量需要移入后台执行的数据提取或处理任务，拥有可视化界面的手机或桌面应用开发

需要不断发送、接收与处理数据的物联网应用

响应式编程在微服务时代……



采用微服务架构的Web应用，由于存在大量的网络IO操作，使之成为了响应式编程技术极适合的应用场景。

响应式编程典型微服务应用场景示例

跨越机器边界的
异步事件驱动

客户端与服务端之间
的双向通讯

跨越机器边界的流式
数据处理

在后面的课程中，我们将介绍上述场景具体的技术方案及示例.....