Spring 5.0 Meets Reactive Programming

sensedia

Cláudio E. de Oliveira

claudio.oliveira@sensedia.com
+55 19 3705-5775

Hello!

Cláudio E. de Oliveira

Book Author

Architect and Software Developer

at *@sensedia*

sensedia

*Launched **at February***

Agenda

- Why Spring Boot
- Reactive Programming
- Project Reactor
- Spring Boot 2
- New Features
- Spring Cloud Projects

Spring Boot

Why Spring Boot became a huge success???

Spring Boot Main Features

- Stand-alone applications
- Starter POMs - *Production Friendly*
- Automatically configurations (a.k.a JPA, Brokers)
- No code generation and *NO XML* anymore
- Production ready features metrics and monitoring

A lot of sub-projects

Spring Framework

Spring Messaging

Spring AMQP

Spring Cloud GCP

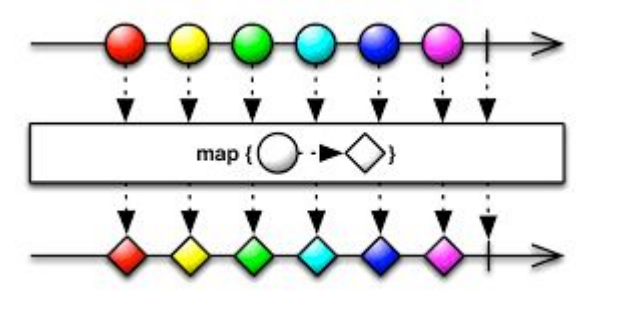
Spring Cloud

Spring Cloud Data Flow

Spring Data

Spring Security

Spring Integration



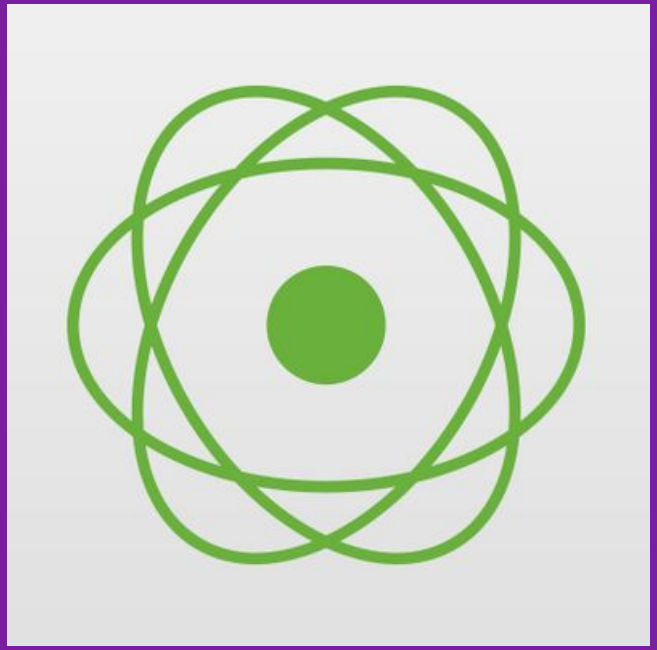
Reactive Programming

The JVM Scenario

sensedia

JVM Players

sensedia



Project Reactor

Reactive library implementation for Reactive Streams Specification

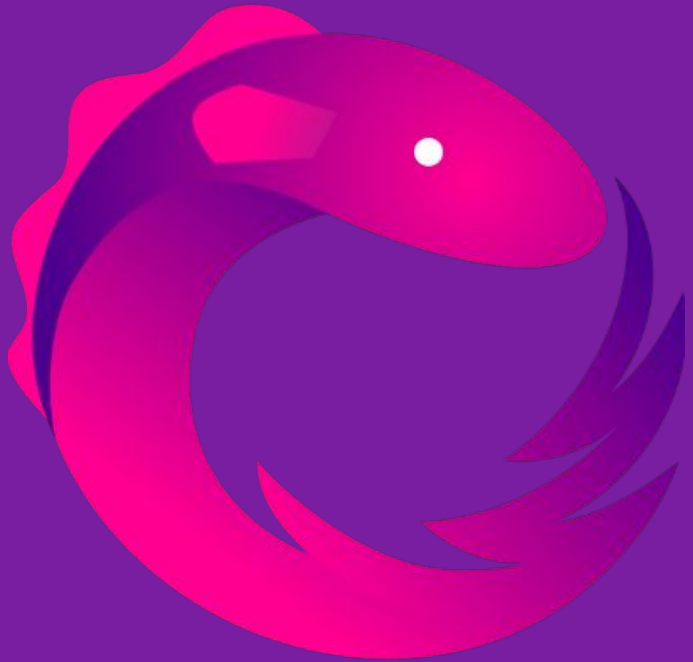
sensedia

Motivation

"Reactor is a fourth-generation Reactive library for building non-blocking applications on the JVM based on the Reactive Streams Specification"

https://projectreactor.io/

Why not RXjava??



sensedia

Let's review some concepts

Reactive Core

Reactor is a fully non-blocking foundation with efficient demand management. It directly interacts with Java 8 functional API, CompletableFuture, Stream and Duration.

Non Blocking IO

Suited for ***Microservices Architecture***, Reactor offers backpressure-ready network engines for HTTP (including Websockets), TCP and UDP. Reactive Encoding/Decoding is fully supported.

Reactor Types

MONO
[0|1]

FLUX
[N]

Samples

```
public Mono<Flight> flight(String id) {
  return this.flightRepository.findById(id);
}

public Flux<Flight> flights() {
  return this.flightRepository.findAll();
}
```

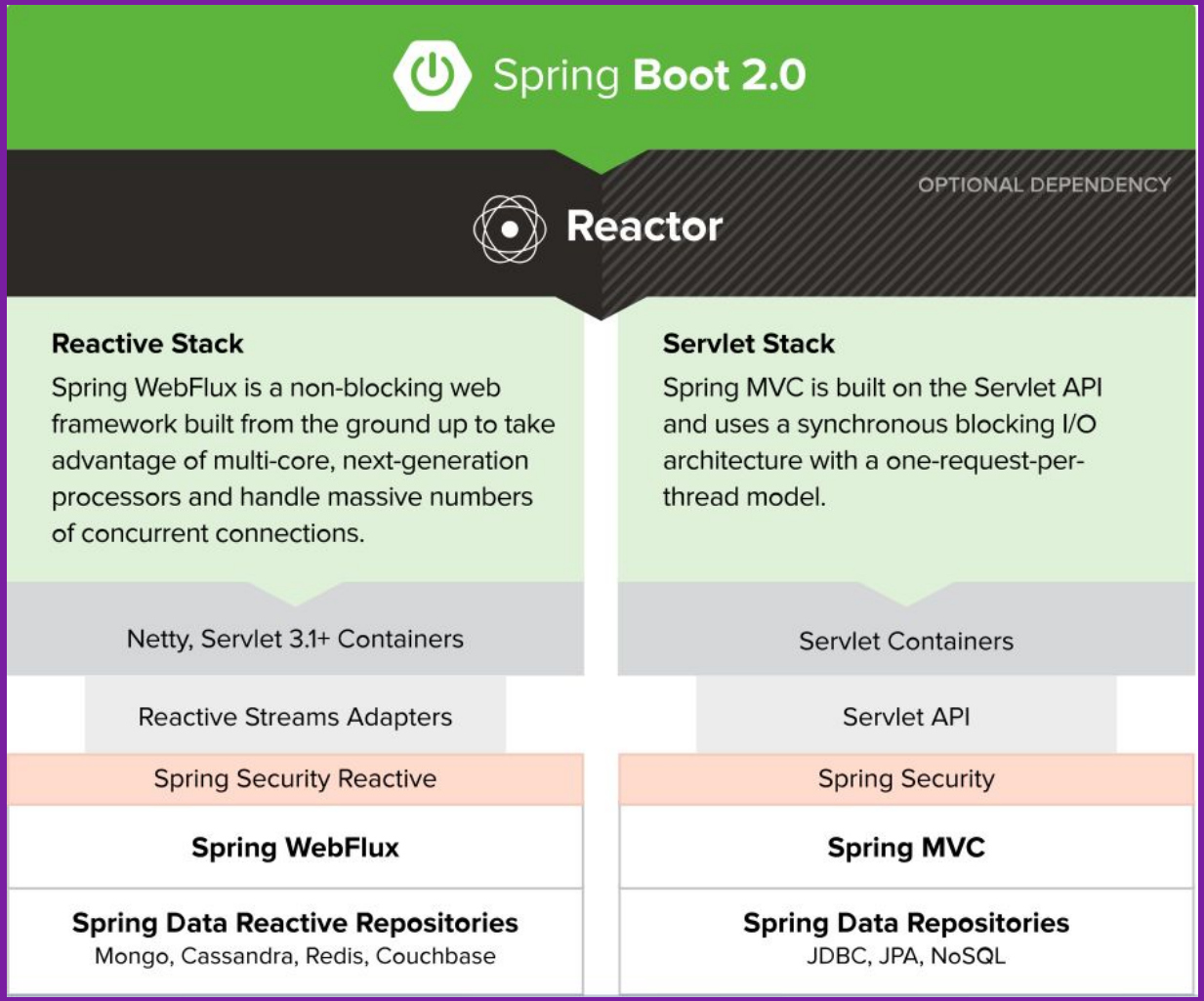
Spring Boot 2

What is coming???


Reactive Programming

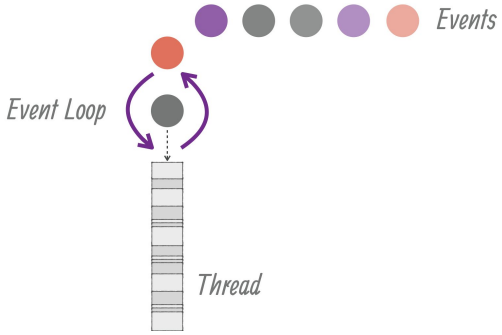
Reactive

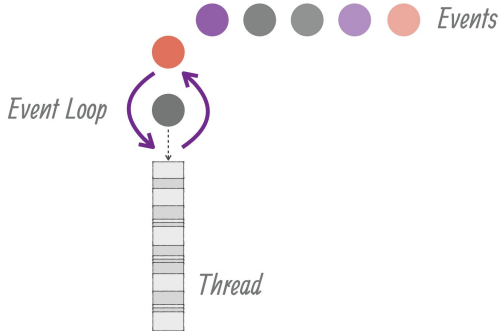
- Simpler Code, more readable
- Focus on Business logic
- Stream processing implies memory efficient
- Flow API Java 9

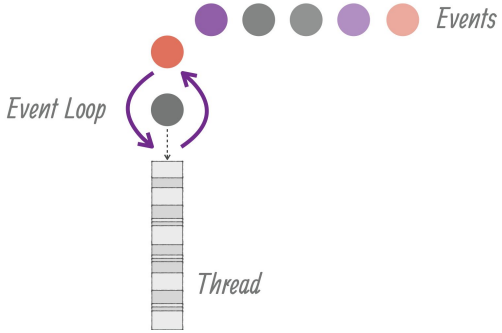

Spring Boot 2.0
Reactor
OPTIONAL DEPENDENCY
Reactive Stack
Spring WebFlux is a non-blocking web framework built from the ground up to take advantage of multi-core, next-generation processors and handle massive numbers of concurrent connections.
Servlet Stack
Spring MVC is built on the Servlet API and uses a synchronous blocking I/O architecture with a one-request-per-thread model.
Netty, Servlet 3.1+ Containers
Servlet Containers
Reactive Streams Adapters
Servlet API
Spring Security Reactive
Spring Security
Spring WebFlux
Spring MVC
Spring Data Reactive Repositories
Mongo, Cassandra, Redis, Couchbase
Spring Data Repositories
JDBC, JPA, NoSQL

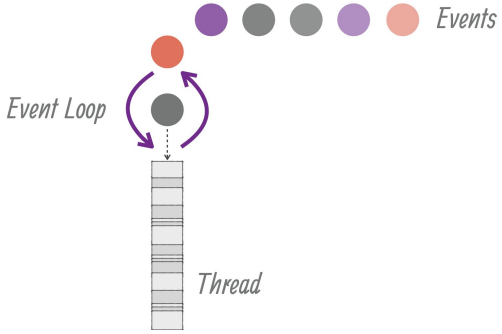
sensedia

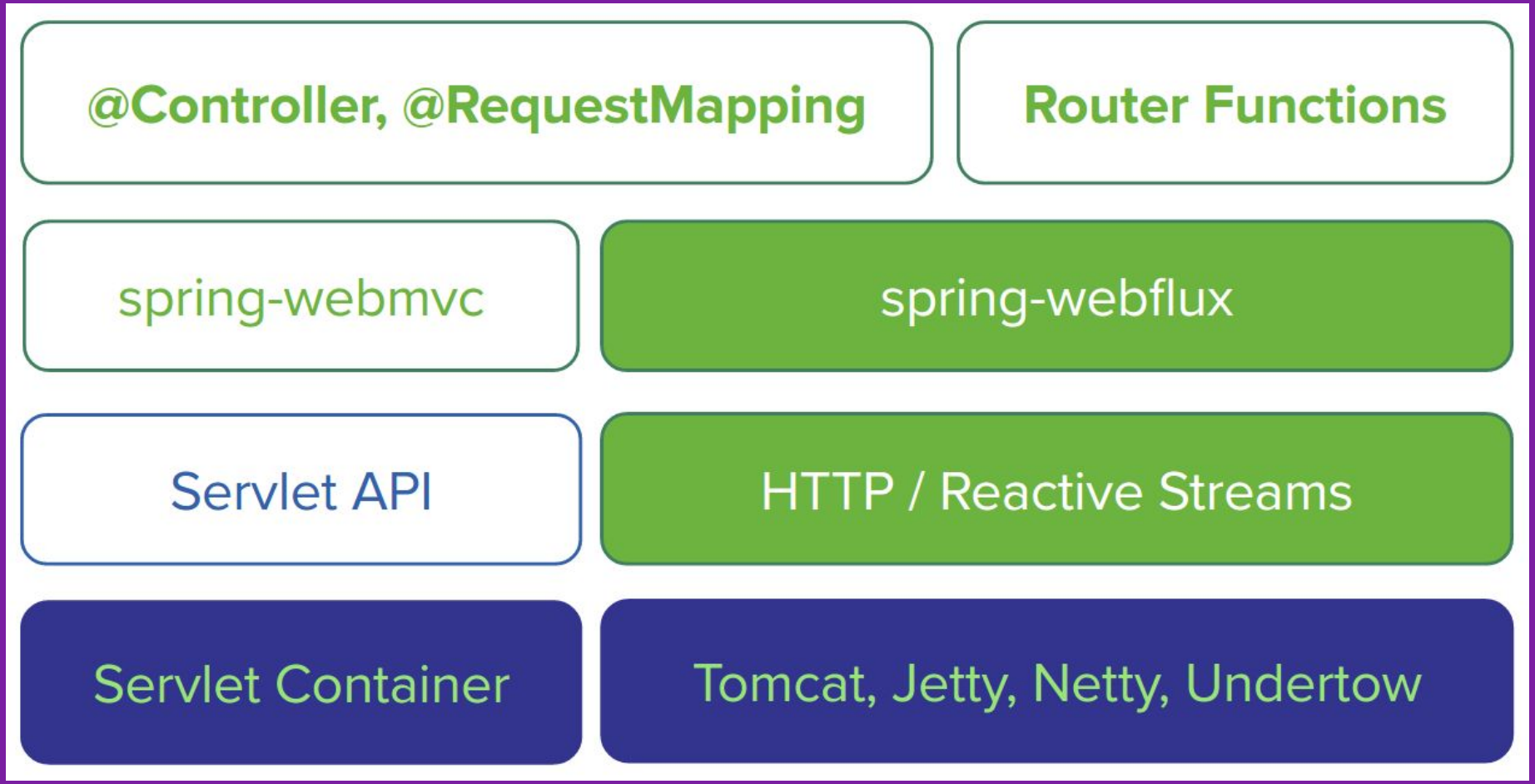
Event-Loop


Events
Event Loop
Thread


Events
Event Loop
Thread


Events
Event Loop
Thread


Events
Event Loop
Thread


@Controller, @RequestMapping
Router Functions
spring-webmvc
spring-webflux
Servlet API
HTTP / Reactive Streams
Servlet Container
Tomcat, Jetty, Netty, Undertow

Talk is cheap

Show me the

CODE

Simple Reactive Controller

```
// omitted imports


@Bean
  public RouterFunction<ServerResponse> routes() {
    return route(GET("/api/goodbye"), serverRequest ->
ok().body(fromPublisher(goodbye(),String.class)))
        .andRoute(GET("/api/nap"), request ->
ok().body(fromPublisher(nap(),String.class)));
}
```

Reactive WebClient

```
// omitted imports

public Mono<Flight> flight(String id) {
  return  discoveryService.serviceAddressFor(this.flightService).next().flatMap(
           address -> this.webClient.mutate()
           .baseUrl(address + "/" + this.flightServiceApiPath + "/" +
id).build().get().exchange()
           .flatMap(clientResponse -> clientResponse.bodyToMono(Flight.class)));
}
```

Spring Data Reactive

redis

mongoDB®

Couchbase

cassandra

sensedia

Reactive

Kotlin Support

sensedia

Simple REST Controller

```
// omitted imports

@RestController
@RequestMapping("/api/temperature")
class TemperatureResource(val temperatureService: TemperatureService) {

    @GetMapping
    fun all() = this.temperatureService.all()

    @PostMapping
    fun register(request: TemperatureRequest) =
            this.temperatureService.register(request)

}
```

Repository

```
// omitted imports

interface TemperatureRepository:ReactiveCrudRepository<Temperature,String>{

    @Tailable
    fun findByDeviceId(deviceId:String):Flux<Temperature>

}
```

Continuous Queries

Makes queries on DB and get notification when data changes.

@Tailable for MongoDB

SSE

Server-Sent Events

SSE REST Controller

```
// omitted imports

@RestController
@RequestMapping("/api/device")
class DeviceResource(val temperatureService: TemperatureService) {

    @GetMapping(path = ["/{id}/real-time"],produces =
[MediaType.APPLICATION_STREAM_JSON_VALUE])
    fun byDeviceRealTime(@PathVariable("id")deviceId:String) =
this.temperatureService.byDevice(deviceId)

}
```

Demo SSE

Java 9

Full Support for Java 9 Module System

Others changes

Actuator Endpoints

Reactive Thymeleaf

jUnit 5 Support

Spring Boot Actuator

How About Spring Cloud Projects???

GitHub

https://github.com/claudioed/kotlin-mongodb-demo

https://github.com/claudioed/goodbye

https://github.com/PacktPublishing/Spring-5.0-By-Example

Contacts

@claudioed

https://www.linkedin.com/in/claudioedoliveira/

claudioed.oliveira@gmail.com

sensedia

Obrigado!

Cláudio E. de Oliveira

claudio.oliveira@sensedia.com
+55 19 3705-5775