Getting Reactive with Spring Framework 5.0’s GA release

Juergen Hoeller
Arjen Poutsma
Rossen Stoyachev
Stephane Maldini
Pieter Humphrey

Spring Framework 5.0!

- **A fully reactive web framework via WebFlux**
- **Native Kotlin extensions**
- **Comprehensive JDK 9, extending Java 8 language & API usage**
- **Integration with popular Java EE 8 APIs**
- **JUnit 5.0 and many further refinements**

Project Reactor

```
Flux.fromIterable(getSomeLongList())
    .mergeWith(Flux.interval(100))
    .doOnNext(serviceA::someObserver)
    .map(d -> d * 2)
    .take(3)
    .onErrorResumeWith(errorHandler::fallback)
    .doAfterTerminate(serviceM::incrementTerminate)
    .subscribe(System.out::println);
```

Synchronous APIs

“elastic”

thread pool

100s, 1000s

waiting blocked threads

Non-blocking code

“parallel”

thread pool

~ per CPU core

busy worker threads

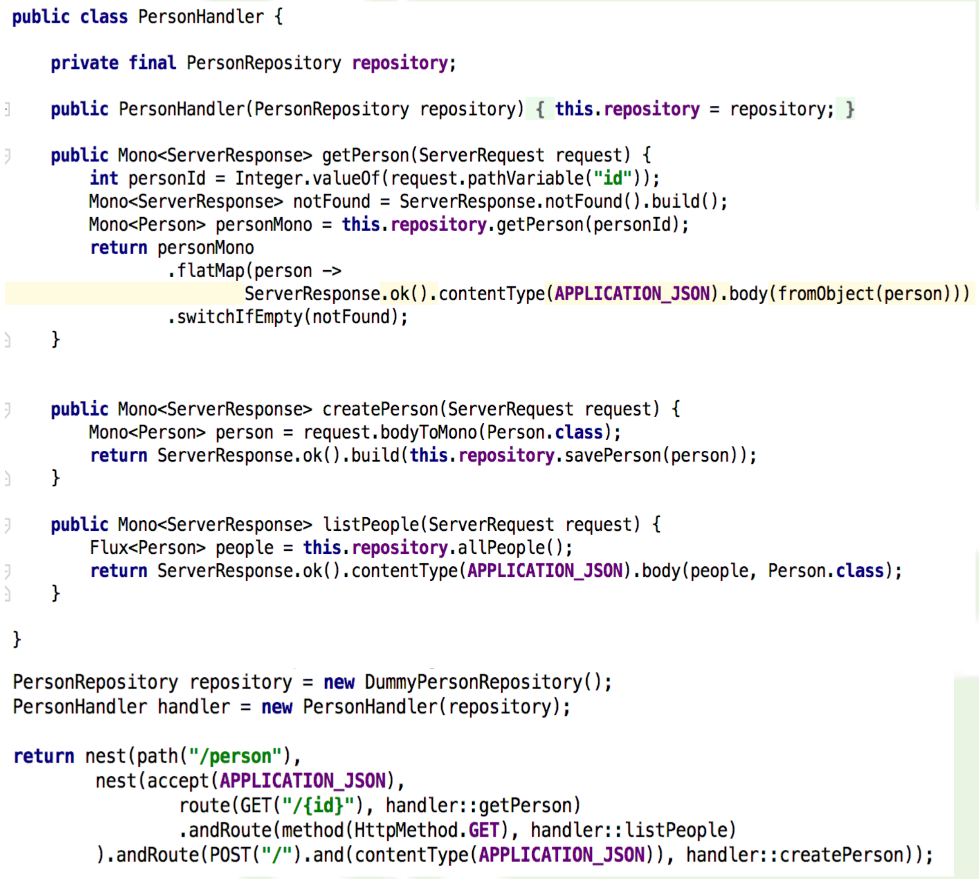
Servlet Stack

- Servlet Container
- Servlet API
- Spring MVC

Reactive Stack

- Netty, Servlet 3.1+, Undertow
- Reactive Streams
- Spring WebFlux

Functional Programming Model



```java
public class PersonHandler {

    private final PersonRepository repository;

    public PersonHandler(PersonRepository repository) { this.repository = repository; }

    public Mono<ServerResponse> getPerson(ServerRequest request) {
        int personId = Integer.valueOf(request.pathVariable("id"));
        Mono<ServerResponse> notFound = ServerResponse.notFound().build();
        Mono<Person> personMono = this.repository.getPerson(personId);
        return personMono
                .flatMap(person ->
                        ServerResponse.ok().contentType(APPLICATION_JSON).body(fromObject(person)))
                .switchIfEmpty(notFound);
    }


    public Mono<ServerResponse> createPerson(ServerRequest request) {
        Mono<Person> person = request.bodyToMono(Person.class);
        return ServerResponse.ok().build(this.repository.savePerson(person));
    }

    public Mono<ServerResponse> listPeople(ServerRequest request) {
        Flux<Person> people = this.repository.allPeople();
        return ServerResponse.ok().contentType(APPLICATION_JSON).body(people, Person.class);
    }

}
```

```java
PersonRepository repository = new DummyPersonRepository();
PersonHandler handler = new PersonHandler(repository);

return nest(path("/person"),
        nest(accept(APPLICATION_JSON),
                route(GET("/{id}"), handler::getPerson)
                .andRoute(method(HttpMethod.GET), handler::listPeople)
        ).andRoute(POST("/").and(contentType(APPLICATION_JSON)), handler::createPerson));
```

Functional Programming Model - Kotlin

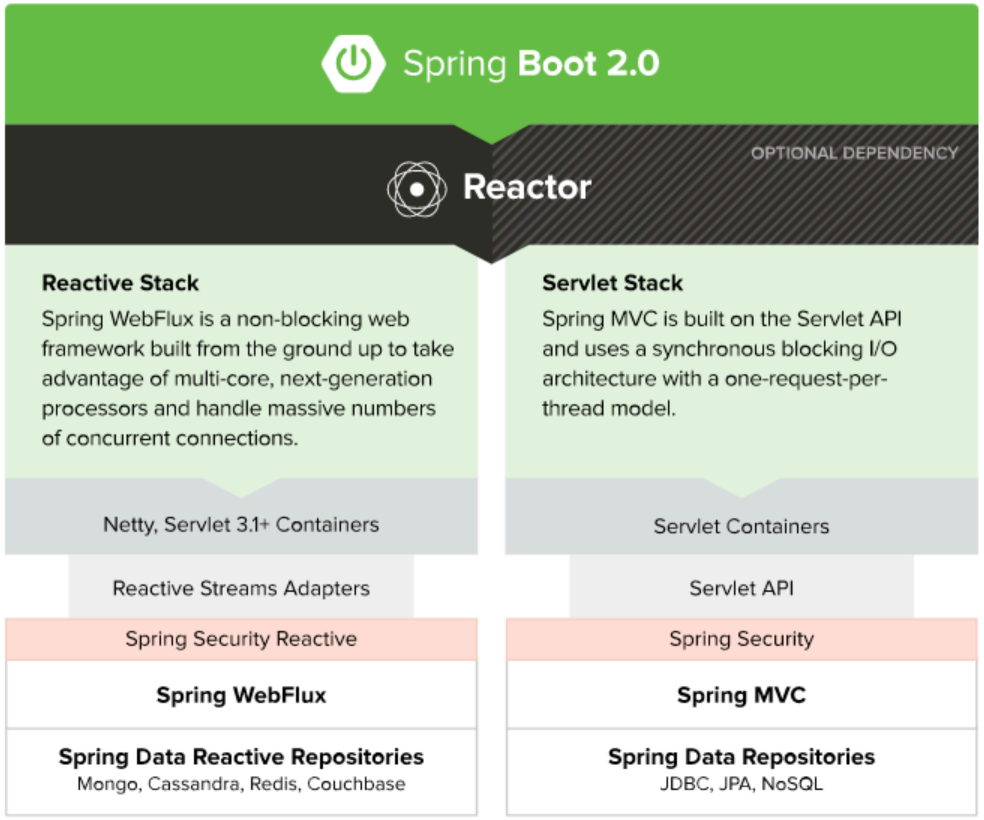
```kotlin
router {
  accept(TEXT_HTML).nest {
    GET("/") { ok().render("index") }
    GET("/sse") { ok().render("sse") }
    GET("/users", userHandler::findAllView)
  }
  "/api".nest {
    accept(APPLICATION_JSON).nest {
      GET("/users", userHandler::findAll)
    }
    accept(TEXT_EVENT_STREAM).nest {
      GET("/users", userHandler::stream)
    }
  }
  resources("/**", ClassPathResource("static/"))
}
```

How should Spring MVC think about this?

```
@PostMapping("/booking")
public Mono<ResponseEntity<Void>> book() {

    return locationClient.get()
            .uri("/cars")
            .retrieve()
            .bodyToFlux(Car.class)
            .take(5)
            .flatMap(car -> bookingClient.post()
                    .uri("/cars/{id}/booking", car.getId())
                    .exchange()
                    .map(this::toBookingResponseEntity))
            .next();
}
```

Looking ahead



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.

Reactive Stack	Servlet Stack
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

Learn More. Stay Connected.

- Poll question
- Click on attachments tab, bookmark!
- Rate this webinar
- Upcoming DDD Webinar Series!

Twitter: twitter.com/springcentral

YouTube: spring.io/video

LinkedIn: spring.io/linkedin

Google Plus: spring.io/gplus

SpringOnePlatform.io

SpringOne Platform

REGISTER NOW.

December 4–7, 2017
Moscone West, San Francisco

Pivotal