Reactive Spring

An Exercise-Driven Approach

Contact Info

Ken Kousen

Kousen IT, Inc.

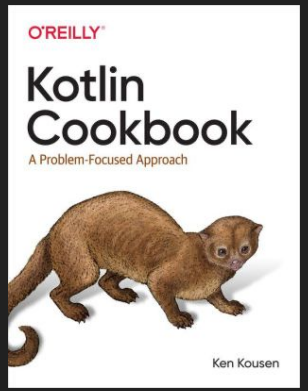
ken.kousen@kousenit.com

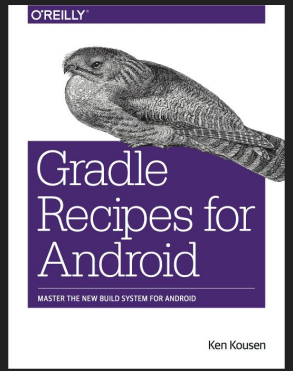
http://www.kousenit.com

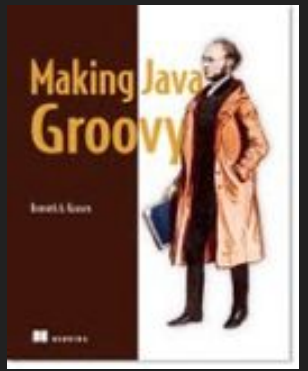
http://kousenit.org (blog)

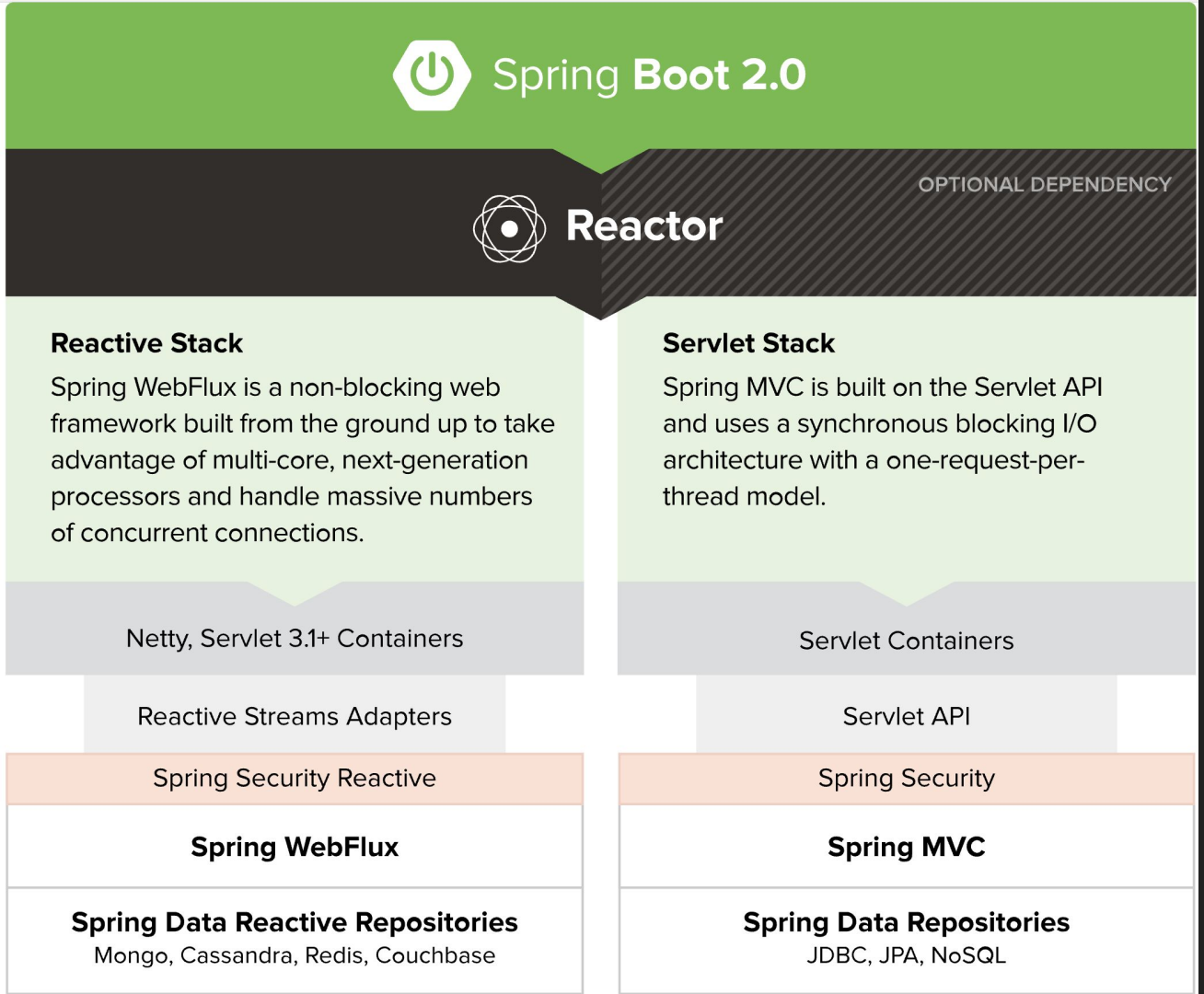
@kenkousen (twitter)

https://kenkousen.substack.com (newsletter)








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

Spring

Project infrastructure

Spring

- Lifecycle management of "beans"
  - Any POJO with getters/setters

Spring

- Provides "services"
  - transactions, security, persistence, …

Spring

- Library of beans available
  - transaction managers
  - rest client
  - DB connection pools
  - testing mechanisms

Spring

- Need "metadata"
  - Tells Spring what to instantiate and configure
    - XML → old style
    - Annotations → better
    - JavaConfig → preferred
  - All still supported

Spring

Application Context

- Collection of managed beans
- the "lightweight" Spring container

Spring Boot

- Easy creation and configuration for Spring apps
  - Many "starters"
  - Gradle or Maven based
- Automatic configuration based on classpath
  - If you add JDBC driver, it adds DataSource bean

Dependency Injection

- Spring adds dependencies on request
  - Annotate field, or setter, or constructor
  - `@Autowired` → autowiring by type
  - `@Resource` (from Java EE) → autowiring by (bean) name, then by type if necessary

Spring Initializr

Website for creating new Spring (Boot) apps

http://start.spring.io

Incorporated into major IDEs

Select features you want

Download zip containing build file

Spring Boot

Application with main method created automatically

Annotated with `@SpringBootApplication`

Gradle or Maven build produces executable jar in build/libs folder

```
$ java -jar appname.jar
```

Or use gradle task `bootRun`

Spring MVC

Annotation based MVC framework

`@Controller` → controllers

`@GetMapping` → annotations for HTTP methods

`@RequestParam` and more for model parameters

Rest Client

Spring includes a class called `RestTemplate`

- Access RESTful web services
- Set HTTP methods, headers, query string, templates
- Use `RestTemplateBuilder` to create one
- Use content negotiation to return JSON or XML
- Convenient `getForObject(url, class)` method

Rest Client

Spring 5 includes a new class called `WebClient`

- Can do async processing
- Understands Flux and Mono

Testing

Spring tests include the JUnit 5 extension

- `@ExtendWith(SpringExtension.class)`
- Part of `@SpringBootTest`
- Annotate tests with `@Test`
- Use normal asserts as usual, but with JUnit 5 additions

Testing

Special annotations for web integration tests

`@WebMvcTest(... controller class ...)`

`MockMvc` package

`MockMvcRequestBuilders`

`MockMvcRequestMatchers`

Parsing JSON

- Several options, but one is the `Jackson JSON 2` library
  - Create classes that map to JSON response
  - `restTemplate.getForObject(url, … your class …)`
- Maps JSON to Java objects

Component Scan

Spring detects annotated classes in the expected folders

`@Component` → Spring bean

`@Controller`, `@Service`, `@Repository` → based on `@Component`

Application properties

Two options for file name

Default folder is `src/main/resources`

- `application.properties` → standard Java properties file
- `application.yml` → YAML format

Transactions

- Spring transactions configured with `@Transactional`
- Spring uses `TransactionManager` to talk to resource
  - usually a relational DB, but other options available

Reactive Spring

Spring 5 → requires Java SE8

WebFlux module

Web on Reactive Stack

Spring Boot 2

Spring WebFlux Framework

WebFlux

Two approaches:

- Annotation-based → Similar to MVC
- Functional → Uses a "routing configuration"

Reactive Streams

Industry specification with wide adoption

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

Supported in Java 9

java.util.concurrent.Flow

Reactive Streams

- Four interfaces
  - Publisher
  - Subscriber
  - Subscription
  - Processor

Publisher

Provides a sequence of elements

Signals emitted:

- onSubscribe → always signaled
- onNext → possibly unbounded number of signals
- onError → only if there is a failure
- onComplete → no more elements available

Publisher

```
public interface Publisher<T> {
    void subscribe(Subscriber<? super T> s);
}
```

Subscriber

Receives signals

```
public interface Subscriber<T> {
    void onSubscribe(Subscription s);
    void onNext(T t);
    void onError(Throwable t);
    void onComplete();
}
```

Subscription

Sent from Publisher to Subscriber

```
public interface Subscription {
    void request(long n);
    void cancel();
}
```

Processor

A combination Publisher/Subscriber

```
public interface Processor<T,R>
        extends Subscriber<T>, Publisher<R> {
}
```

No additional methods

Project Reactor

Reactive library for the JVM based on Reactive Streams

Reactive Core → fully non-blocking

Typed sequences → Flux, Mono

Non-blocking IO with backpressure

WebFlux

- Annotated Controllers
  - Support reactive return types
    - Reactor, RxJava 2
- Functional Endpoints
  - Lambda-based, functional programming model
  - Library of utilities to route and handle requests

WebFlux

- Uses Netty by default
  - Others: Undertow, Tomcat (Servlet 3.1+), Jetty (Servlet 3.1+)

WebClient

Reactive, non-blocking client for HTTP requests

Functional API with Java 8 lambdas

Both synchronous and asynchronous

Use WebTestClient for testing → mock request and response objects

WebFlux

- Functional Endpoints
  - HandlerFunction
  - RouterFunction

Exercises

HTML docs: http://www.kousenit.com/reactivespring/

Solutions:

https://github.com/kousen/reactive-spring

https://github.com/kousen/spring-and-spring-boot (MVC, non-reactive)

Docs

- Spring API JavaDocs:
  - http://docs.spring.io/spring/docs/current/javadoc-api/
- Spring Reference Guide:
  - https://spring.io/docs/reference
- Spring Boot Reference Guide
  - http://docs.spring.io/spring-boot/docs/1.5.1.RELEASE/reference/htmlsingle/
- Spring Initializr
  - https://start.spring.io/

Docs

Wait, you're all on Safari, so…

- Learning Path: Learn Spring and Spring Boot
  - Includes Spring Framework Essentials
- Spring in Action, 5th Edition
- Spring Boot in Action
- … lots and lots more …