

# spring-kafka-quickstart (0.0.1)

Maksim Kostromin

Version 0.0.1, 2018-06-28 13:54:50 UTC

# Table of Contents

1. Introduction . . . . .	2
2. Implementation . . . . .	3
3. Links . . . . .	6

Travis CI status: [Build Status](#)

# Chapter 1. Introduction

This repo contains rapid development guide, how to quick bootstrap with spring and kafka.

Read [project reference documentation](#) on github pages

generated by [generator-jvm](#) yeoman generator (java-spring-boot)

# Chapter 2. Implementation

*create project*

```
brew install node
npm i -g yo generator-jvm
yo jvm -n spring-kafka-quickstart -t java-spring-boot
idea spring-kafka-quickstart/pom.xml
```

*bootstrap kafka using [spring boot \(cloud\) CLI](#)*

```
spring cloud kafka
```

*add dependencies [pom.xml](#) file:*

```
<dependencies>
  <dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

*add dependencies [build.gradle](#) file:*

```
dependencies {
  implementation('org.springframework.kafka:spring-kafka')
  testImplementation('org.springframework.kafka:spring-kafka-test')
}
```

*add kafka listener:*

```
@Log4j2
@Service
public class MessageListener {

    @KafkaListener(topics = "messages")
    public void on(final ConsumerRecord<Object, Object> message) {
        log.info("received message: {}", message.value());
        Mono.just(new Message().setAt(now())
                               .setBody(message.value().toString()))
            .subscribe(msg -> log.info("saving message object: '{}'", msg));
    }
}
```

*add kafka sender functionality:*

```
@Configuration
@RequiredArgsConstructor
class WebfluxRoutesConfig {

    static final ParameterizedTypeReference<Map<String, String>> sendMessageRequestType
        = new ParameterizedTypeReference<Map<String, String>>() {};

    final KafkaTemplate<Object, Object> kafka;

    @Bean
    HandlerFunction<ServerResponse> sendMessageHandler() {
        return request ->
            ok().body(request.bodyToMono(sendMessageRequestType)
                .map(it -> it.getOrDefault("message", ""))
                .filter(it -> !it.trim().isEmpty())
                .doOnNext(message -> kafka.send("messages", message))
                .map(s -> "message sent.")
                .flatMap(Mono::just), Object.class);
    }

    @Bean
    RouterFunction routes(final HandlerFunction<ServerResponse> fallbackHandler) {
        return
            route(
                POST("/"),
                sendMessageHandler()
            ).andOther(
                route(
                    GET("/**"),
                    fallbackHandler
                )
            )
        ;
    }
}
```

*build run and test (gradle)*

```
./gradlew
bash -jar build/libs/*.jar
http :8080 message=lolo
http :8080 message=trololo
```

*build run and test (maven)*

```
./mvnw
bash target/*.jar
```

# Chapter 3. Links

- [GitHub repo](#)
- [GitHub pages](#)