

event-sourcing-payment-system (0.0.1)

Maksim Kostromin

Version 0.0.1, 2018-08-12 19:12:13 UTC

Table of Contents

1. Introduction	2
2. Components	3
3. Prepare kafka	4
4. CQRS and Event Sourcing API (commands, queries, errors)	5
5. Commands application	6
6. Links	8

Travis CI status: [\[Build Status\]](#)

Chapter 1. Introduction

In progress: event-sourcing app using spring-boot + spring-cloud-kafka + kafka-streams

Chapter 2. Components

- `api` - module contains all commands, events and errors defined in application
- `commands-app` - REST API application for sending payment system commands through kafka

TODO:

- `queries-app` - application for querying payment system

Chapter 3. Prepare kafka

install spring boot cloud CLI

```
brew tap pivotal/tap
brew search spring
brew install springboot

spring install org.springframework.cloud:spring-cloud-cli:2.0.0.RELEASE

spring cloud kafka

# check if zookeeper is running:
lsof -i:2181|awk '{print $2}'

# check if kafka is running:
lsof -i:9092|awk '{print $2}'
```

Chapter 4. CQRS and Event Sourcing API (commands, queries, errors)

Commands REST API

```
sealed class Error(override val message: String) : RuntimeException(message)
sealed class Command
sealed class Event
```

Chapter 5. Commands application

Including api project as included build using gradle composit builds approach

```
// file: settings.gradle
rootProject.name = 'commands-app'
enableFeaturePreview 'STABLE_PUBLISHING'

includeBuild('../api') {
    dependencySubstitution {
        substitute module('com.github.daggerok:api') with project(':')
    }
}

// file: gradle/java.gradle
allprojects {
    apply plugin: 'java'

    apply plugin: 'io.franzbecker.gradle-lombok'
    lombok.version = project.lombokVersion

    version = '0.0.1'
    group = 'com.github.daggerok'
    sourceCompatibility = targetCompatibility = "$javaVersion"

    defaultTasks 'clean', 'build'

    dependencies {
        implementation 'com.github.daggerok:api:0.0.1'
        // In java we trust...
        implementation "io.vavr:vavr:$vavrVersion"
    }
}
```



```
@Configuration
class Rest(val producer: Producer) {

    companion object {
        val ref = mutableMapOf<String, String>():class.java
    }

    @Bean
    fun routes() = router {
        ("/").nest {
            contentType(MediaType.APPLICATION_JSON_UTF8)

            POST("/") {
                ok().body(it.bodyToMono(ref)
                    .map { it["message"].orEmpty() }
                    .map { MessageBuilder.withPayload(it).build() }
                    .map { producer.send(it) }
                    .subscribeOn(Schedulers.elastic())
                    .flatMap { Mono.just("sending message...") }
                )
            }

            GET("/**") {
                val map = mapOf(
                    "errors" to listOf(
                        Error::class.java.name
                    ),
                    "commands" to listOf(
                        Command::class.java.name
                    )
                )
                ok().body(
                    Mono.just(map), map.javaClass
                )
            }
        }
    }
}
```

Chapter 6. Links

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