

apache-camel-spring-boot-1.x (0.0.1)

Maksim Kostromin

Version 0.0.1, 2018-06-14 23:47:22 UTC

Table of Contents

1. Introduction	2
2. Implementation	3
2.1. camel spring configuration	3
2.2. move all files from directory	3
2.3. move files via jms	3
2.4. change filename by using camel header	5
2.5. process exchange	5
2.6. easier exchange	6
2.7. choice	7
3. BOM	9
4. Links	10

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

Chapter 1. Introduction

[Reference documentation](#)

[See Travis CI integration tests for details](#)

generated by [generator-jvm](#) yeoman generator (java-spring-boot_1.x)

Chapter 2. Implementation

2.1. camel spring configuration

src/main/resources/application.properties

```
camel.springboot.main-run-controller=true
```

2.2. move all files from directory

source directory: */tmp/camel-in/* destination directory: */tmp/camel-out/*

src/main/java/daggerok/app/MoveFilesCamelConfig.java

```
@Configuration
@RequiredArgsConstructor
public class MoveFilesCamelConfig {

    @Bean
    public RouteBuilder moveFilesRoute() {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                from("file:///tmp/camel-in")
                    .routeId("dir-to-dir")
                    .to("file:///tmp/camel-out");
            }
        };
    }
}
```

2.3. move files via jms

source directory: */tmp/camel-jms-in/* process queue: *files* destination directory: */tmp/camel-jms-out/*

move files via jms queue `jms:queue:files`

```
@Slf4j
@Configuration
@RequiredArgsConstructor
public class MoveFilesViaJmsCamelConfig {

    @Bean
    public RouteBuilder moveFilesViaJmsRoute() {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {

                from("file:///tmp/camel-jms-in")
                    .routeId("in-dir-to-jms")
                    .transform()
                    .body(GenericFile.class, genericFile -> {
                        final File file = File.class.cast(genericFile.getFile());
                        try (final FileInputStream inputStream = new FileInputStream(file)) {
                            final InputStreamReader streamReader = new InputStreamReader
(inputStream);
                            final BufferedReader in = new BufferedReader(streamReader);
                            return in.lines().collect(joining());
                        } catch (IOException e) {
                            log.error(e.getLocalizedMessage());
                            throw new RuntimeException(e);
                        }
                    })
                    .to("jms:queue:files")
                ;

                from("jms:queue:files")
                    .routeId("jms-to-out-dir")
                    .to("file:///tmp/camel-jms-out")
                ;
            }
        };
    }
}
```

```
@Component
@RequiredArgsConstructor
class ApplicationJmsComponentCustomizer implements ComponentCustomizer<JmsComponent> {

    final ConnectionFactory connectionFactory;

    @Override
    public void customize(JmsComponent component) {
        component.setConnectionFactory(connectionFactory);
    }
}
```

2.4. change filename by using camel header

source directory: `/tmp/camel-filename-in/` destination directory: `/tmp/camel-filename-out/`

setup camel header to change output filename in pattern: `uuid-changed.txt`

```
@Configuration
@RequiredArgsConstructor
public class ChangeFilenameHeaderCamelConfig {

    @Bean
    public RouteBuilder filenameHeaderRoute() {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                from("file:///tmp/camel-filename-in")
                    .routeId("change-filename-header")
                    .setHeader("CamelFileName", () -> format("%s-changed.txt", UUID.
randomUUID().toString()))
                    .to("file:///tmp/camel-filename-out");
            }
        };
    }
}
```

2.5. process exchange

source directory: `/tmp/camel-exchange-in/` destination directory: `/tmp/camel-exchange-out/`

man in a middle: do additional stuff with message during camel pipeline (for instance, log message body):

```
@Configuration
@RequiredArgsConstructor
public class ExchangeProcessorCamelConfig {

    @Bean
    public RouteBuilder processExchangeRoute() {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                from("file:///tmp/camel-exchange-in")
                    .routeId("exchange-processor")
                    .process().exchange(exchange -> {
                        final Message inputMessage = exchange.getIn();
                        final Object body = inputMessage.getBody(String.class);
                        log.info("handling message body in exchange consumer: {}", body);
                    })
                    .to("file:///tmp/camel-exchange-out");
            }
        };
    }
}
```

2.6. easier exchange

source directory: `/tmp/camel-easier-exchange-in/` destination directory: `/tmp/camel-easier-exchange-out/`

same, but easier:

```
@Configuration
@RequiredArgsConstructor
public class EasierExchangeProcessorCamelConfig {

    @Bean
    public RouteBuilder easierExchangeRoute() {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                from("file:///tmp/camel-easier-exchange-in")
                    .routeId("easier-exchange")
                    .process()
                    .message(message -> {
                        final String body = message.getBody(String.class);
                        final Map<String, Object> headers = message.getHeaders();
                        log.info("easy body: {}", body);
                        headers.entrySet().parallelStream()
                            .filter(e -> "CamelFileLength".contains(e.getKey()))
                            .forEach(e -> log.info("header({}): {}", e.getKey(), e.getValue()));
                    })
                /* // absolutely same, even more easier!
                .body(String.class, (body, headers) -> {
                    log.info("easiest: {}", body);
                    headers.entrySet().parallelStream()
                        .filter(e -> "CamelFileLength".contains(e.getKey()))
                        .forEach(e -> log.info("header({}): {}", e.getKey(), e.getValue()));
                })
                */
                .to("file:///tmp/camel-easier-exchange-out");
            }
        };
    }
}
```

2.7. choice

source directory: `/tmp/camel-choice-in/` destination directory: `/tmp/camel-choice-out/`

```
@Configuration
@RequiredArgsConstructor
public class ChoiceCamelConfig {

    @Bean
    public RouteBuilder choiceRoute() {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                from("file:///tmp/camel-choice-in")
                    .routeId("choice")
                    .choice()
                    .when(exchange -> {
                        final String body = exchange.getMessage().getBody(String.class);
                        return !body.toLowerCase().contains("err");
                    })
                    .to("file:///tmp/camel-choice-out")
                    .otherwise()
                    .to("file:///tmp/camel-choice-error")
                    .endChoice()
            }
        };
    }
}
```

Chapter 3. BOM

maven

```
<dependencyManagement>
  <dependencies>
    <!-- now we can easily use compatible apache camel components -->
    <dependency>
      <groupId>org.apache.camel</groupId>
      <artifactId>camel-spring-boot-dependencies</artifactId>
      <version>${apache.camel.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

gradle

```
dependencyManagement {
  imports {
    mavenBom "org.apache.camel:camel-spring-boot-dependencies:${apacheCamelVersion}"
  }
}
```

Chapter 4. Links

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