

# Dead Letter

## Cenários de Falha

Quando trabalhamos com sistemas baseado na troca de mensagens, vários tipos de erros podem ocorrer, até mesmo mais erros do que em sistemas monolitos devido a natureza de sistemas distribuídos.

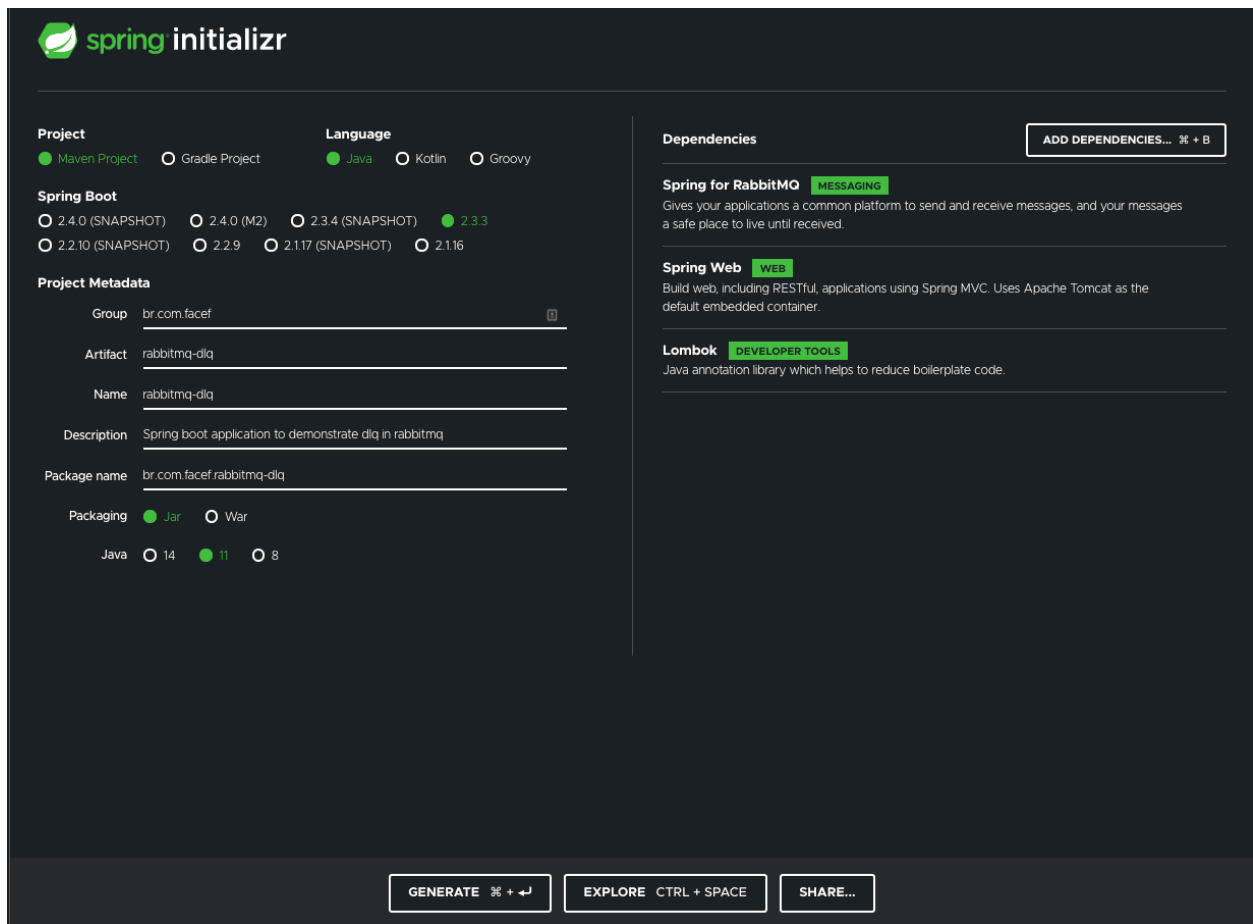
Para ter uma visão dos tipos de erros mais comuns que podem ocorrer quando se utiliza mensageria:

- Falhas de rede ou operação de leitura e escrita.
- Erros por falta ou falha de configuração no sistema de mensageria.
- Falhas nas configurações entre clientes e AMQP Brokers (RabbitMQ), limites, autenticação, policys.
- Exceções que violam alguma regra de negócio ou da aplicação.

Podem existir vários outros tipos de falhas, além das mais comuns são as já citadas.

## Exemplo

Para gerar o projeto podem acessar a [URL](#), nessa url iremos utilizar o **Spring Initializr** para gerar a estrutura padrão do projeto já adicionando a biblioteca do RabbitMQ.



The image shows the Spring Initializr web interface. It is a dark-themed form for generating a Spring project. The 'Project' section has 'Maven Project' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '2.3.3' selected. The 'Project Metadata' section contains fields for Group (br.com.facef), Artifact (rabbitmq-dlq), Name (rabbitmq-dlq), Description (Spring boot application to demonstrate dlq in rabbitmq), and Package name (br.com.facef.rabbitmq-dlq). The 'Packaging' section has 'Jar' selected, and the 'Java' section has '11' selected. The 'Dependencies' section on the right lists 'Spring for RabbitMQ' (Messaging), 'Spring Web' (Web), and 'Lombok' (Developer Tools). At the bottom, there are buttons for 'GENERATE', 'EXPLORE', and 'SHARE...'.

Primeiramente precisamos iniciar o RabbitMQ:

- Iniciando diretamente via docker

```
docker run -it --rm --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3-management
```

- Iniciando via docker-compose

```
docker-compose stop && docker-compose rm -f && docker-compose up -d
```

- Iniciando via helm + microk8s

```
microk8s helm3 install rabbitmq stable/rabbitmq
```

Para acompanhar a inicialização do container pode-se utilizar o comando:

```
docker-compose logs -f
```

Verificando os logs:

```
rabbitmq_1 | 2020-08-25 22:59:22.925 [info] <0.675.0> Ready to start client connection listeners
rabbitmq_1 | 2020-08-25 22:59:22.930 [info] <0.980.0> started TCP listener on [::]:5672
rabbitmq_1 | 2020-08-25 22:59:23.273 [info] <0.675.0> Server startup complete; 4 plugins started.
rabbitmq_1 | * rabbitmq_prometheus
rabbitmq_1 | * rabbitmq_management
rabbitmq_1 | * rabbitmq_web_dispatch
rabbitmq_1 | * rabbitmq_management_agent
rabbitmq_1 | completed with 4 plugins.
```

Após o início do container, podemos acessar a URL do admin através do endereço

<http://localhost:15672/>

**username:** guest

**password:** guest

The image shows the RabbitMQ Admin UI login page. At the top, there is the RabbitMQ logo. Below it, there are two input fields: 'Username:' and 'Password:'. Each field has a small 'x' icon on the right side. Below the password field is a 'Login' button.

Após o login temos acesso a interface de gerenciamento do servidor do rabbitmq, onde temos as visões referentes há:

- **Overview**
- **Connections**
- **Channels**
- **Exchanges**
- **Queues**
- **Admin**

Ao clicar em **exchanges** podemos visualizar as exchanges padrões que o próprio servidor o rabbitmq cria quando iniciamos o serviço. E temos a opção de criar nossas próprias exchanges através do admin.

## Exchanges

▼ All exchanges (7)

Pagination


Page  of 1 - Filter:  ☐ Regex ?

| Name               | Type    | Features | Message rate in | Message rate out | +/- |
|--------------------|---------|----------|-----------------|------------------|-----|
| (AMQP default)     | direct  | D        |                 |                  |     |
| amq.direct         | direct  | D        |                 |                  |     |
| amq.fanout         | fanout  | D        |                 |                  |     |
| amq.headers        | headers | D        |                 |                  |     |
| amq.match          | headers | D        |                 |                  |     |
| amq.rabbitmq.trace | topic   | D I      |                 |                  |     |
| amq.topic          | topic   | D        |                 |                  |     |

► [Add a new exchange](#)

Clicando em **queues** podemos visualizar que por default nenhuma fila é criada no momento da inicialização do servidor.

---

 RabbitMQ™

RabbitMQ 3.8.7Erlang 23.0.3

OverviewConnectionsChannelsExchangesQueuesAdmin

---

## Queues

▼ All queues (0)

Pagination

Page  of 0 - Filter:  ☐ Regex ?

... no queues ...

► Add a new queue

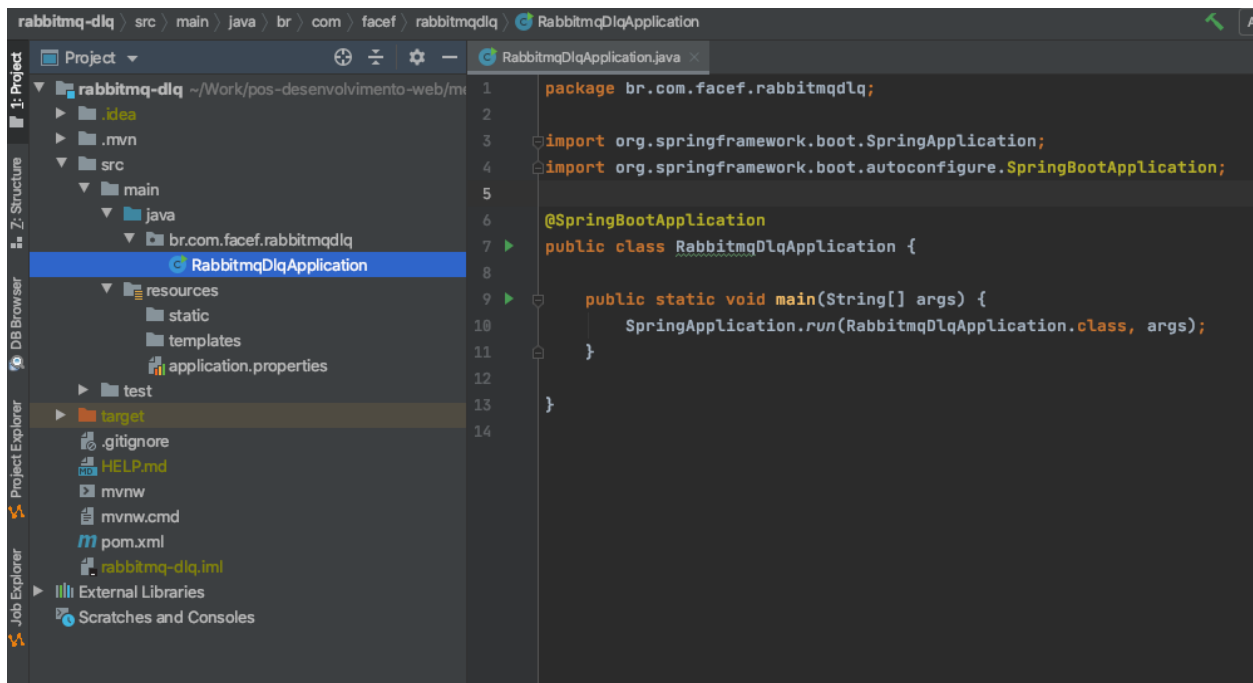
---

HTTP APIServer DocsTutorialsCommunity SupportCommunity SlackCommuni

Para demonstrar o funcionamento, irei fazer um passo a passo com um exemplo funcional, implementando um método que será executado durante a inicialização do projeto para enviar uma mensagem fake para a Fila e um consumer que irá processar a mensagem e lançar uma exception para simular um erro de regra de negócio.

Passo a passo:

- **Initial project**



- **Include docker-compose**

```
version: '3'
services:
  rabbitmq:
    image: rabbitmq:3-management
    ports:
      - "5672:5672"
      - "15672:15672"
```

- **Include rabbitmq configuration to DirectExchange**

```
package br.com.facef.rabbitmqdlq.configuration;

import org.springframework.amqp.core.Binding;
import org.springframework.amqp.core.BindingBuilder;
import org.springframework.amqp.core.DirectExchange;
import org.springframework.amqp.core.ExchangeBuilder;
import org.springframework.amqp.core.Queue;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class DirectExchangeConfiguration {

    public static final String DIRECT_EXCHANGE_NAME = "order-exchange";
    public static final String ORDER_MESSAGES_QUEUE_NAME = "order-messages-queue";

    @Bean
    Queue orderMessagesQueue() {
        return new Queue(ORDER_MESSAGES_QUEUE_NAME);
    }
}
```

```

@Bean
DirectExchange exchange() {
    return ExchangeBuilder.directExchange(DIRECT_EXCHANGE_NAME).durable(true).build();
}

@Bean
Binding bindingOrderMessagesQueue(
    @Qualifier("orderMessagesQueue") Queue queue, DirectExchange exchange) {
    return BindingBuilder.bind(queue).to(exchange).with(ORDER_MESSAGES_QUEUE_NAME);
}
}

```

- **Include a class to send message to a exchange**

```

package br.com.facef.rabbitmqdlq.producer;

import static br.com.facef.rabbitmqdlq.configuration.DirectExchangeConfiguration.DIRECT_EXCHANGE_NAME;
import static br.com.facef.rabbitmqdlq.configuration.DirectExchangeConfiguration.ORDER_MESSAGES_QUEUE_NAME;

import java.time.LocalDateTime;
import java.util.UUID;
import lombok.extern.slf4j.Slf4j;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
@Slf4j
public class MessageProducer {

    @Autowired private RabbitTemplate rabbitTemplate;

    public void sendFakeMessage() {
        log.info("Sending a fake message...");
        this.rabbitTemplate.convertAndSend(
            DIRECT_EXCHANGE_NAME,
            ORDER_MESSAGES_QUEUE_NAME,
            "FAKE-MESSAGE-"
                .concat(LocalDateTime.now().toString())
                .concat(UUID.randomUUID().toString()));
    }
}

```

- **Change to on the startup create a fake message on rabbitmq**

```

package br.com.facef.rabbitmqdlq;

import br.com.facef.rabbitmqdlq.producer.MessageProducer;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.event.ApplicationReadyEvent;
import org.springframework.context.event.EventListener;

@SpringBootApplication
@Slf4j
public class RabbitmqDlqApplication {

    @Autowired private MessageProducer messageProducer;
}

```

```

public static void main(String[] args) {
    SpringApplication.run(RabbitmqDlqApplication.class, args);
}

@EventListener(ApplicationReadyEvent.class)
public void runningAfterStartup() {
    log.info("Running method after startup to send messages!");
    messageProducer.sendFakeMessage();
}
}

```

- **Create a consumer to simulate BusinessException on processing message**

```

package br.com.facef.rabbitmqdlq.consumer;

import br.com.facef.rabbitmqdlq.configuration.DirectExchangeConfiguration;
import lombok.extern.slf4j.Slf4j;
import org.springframework.amqp.AmqpRejectAndDontRequeueException;
import org.springframework.amqp.core.Message;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.context.annotation.Configuration;

@Configuration
@Slf4j
public class MessageConsumer {

    @RabbitListener(queues = DirectExchangeConfiguration.ORDER_MESSAGES_QUEUE_NAME)
    public void processOrderMessage(Message message) {
        log.info("Processing message: {}", message.toString());
        // By default the messages will be requeued
        throw new RuntimeException("Business Rule Exception");
        // To dont requeue message can throw AmqpRejectAndDontRequeueException
        // throw new AmqpRejectAndDontRequeueException("Business Rule Exception");
    }
}

```

```

# To disable requeue enable property as false
#spring.rabbitmq.listener.simple.default-requeue-rejected=false

```

Após a finalização da implementação do projeto iremos executar o projeto utilizando o próprio plugin do spring.

```

./mvnw clean spring-boot:run

```

Após o startup da aplicação conseguimos ver nos **logs o envio da mensagem**.



```

Spring
:: Spring Boot :: (v2.3.3.RELEASE)

2020-08-27 22:11:41.363 INFO 25291 --- [ main] b.c.f.r.RabbitmqDlqApplication : Starting RabbitmqDlqApplication on Cassios-MacBook-Air.local with PID 25291 (/Users/cassiothadeu/work/pos-desenvolvimento/thadeu/work/pos-desenvolvimento-web/messengeria-streams/rabbitmq-dlq)
2020-08-27 22:11:41.369 INFO 25291 --- [ main] b.c.f.r.RabbitmqDlqApplication : No active profile set, falling back to default profiles: default
2020-08-27 22:11:43.249 INFO 25291 --- [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-08-27 22:11:43.284 INFO 25291 --- [ main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-08-27 22:11:43.285 INFO 25291 --- [ main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.37]
2020-08-27 22:11:43.467 INFO 25291 --- [ main] o.s.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-08-27 22:11:43.467 INFO 25291 --- [ main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2015 ms
2020-08-27 22:11:44.536 INFO 25291 --- [ main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-08-27 22:11:45.033 INFO 25291 --- [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2020-08-27 22:11:45.036 INFO 25291 --- [ main] o.s.a.r.c.CachingConnectionFactory : Attempting to connect to: [localhost:5672]
2020-08-27 22:11:45.223 INFO 25291 --- [ main] o.s.a.r.c.CachingConnectionFactory : Created new connection: rabbitConnectionFactory#4defd42:0/SimpleConnectionFactory#7593ea79 [delegate=amqp://guest@127.0.0.1:5672]
2020-08-27 22:11:45.379 INFO 25291 --- [IntContainer#0-1] b.c.f.r.consumer.MessageConsumer : Processing message: (Body: 'FAKE-MESSAGE-2020-08-27T22:11:03.778843f297d90a-e3f9-4177-81d2-1d20701b9612' MessageProperties { ryMode=PERSISTENT, priority=0, redelivered=true, receivedExchange=order-exchange, receivedRoutingKey=order-messages-queue, deliveryTag=1, consumerTag=amq.ctag-FTBUALpTXXtWm4_0TRTPJw, consumerQueue=order-messages-queue})
2020-08-27 22:11:45.387 INFO 25291 --- [ main] b.c.f.r.RabbitmqDlqApplication : Started RabbitmqDlqApplication in 9.027 seconds (JVM running for 10.543)
2020-08-27 22:11:45.392 INFO 25291 --- [ main] b.c.f.r.producer.MessageProducer : Running method after startup to send messages
2020-08-27 22:11:45.394 WARN 25291 --- [IntContainer#0-1] s.a.r.l.ConditionalRejectingErrorHandler : Sending a fake message...
2020-08-27 22:11:45.394 WARN 25291 --- [IntContainer#0-1] s.a.r.l.ConditionalRejectingErrorHandler : Execution of Rabbit message listener failed.
```

Como estamos lançando uma exception fake e não estamos fazendo a tratativa estoura erros no console e o processo entra em loop devido ao processo de **requeue que é o comportamento padrão**.

```

org.springframework.amqp.rabbit.support.ListenerExecutionFailedException: Listener method 'public void br.com.facef.rabbitmqdlq.consumer.MessageConsumer.processOrderMessage(org.springframework.amqp.core.Message)' threw exception
    at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.invokeHandler(MessagingMessageListenerAdapter.java:228) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.invokeHandlerAndProcessResult(MessagingMessageListenerAdapter.java:140) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.doInvokeListener(AbstractMessageListenerContainer.java:1591) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.actualInvokeListener(AbstractMessageListenerContainer.java:1510) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.doExecuteListener(AbstractMessageListenerContainer.java:1489) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.executeListener(AbstractMessageListenerContainer.java:1433) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.doReceiveAndExecute(SimpleMessageListenerContainer.java:970) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.receiveAndExecute(SimpleMessageListenerContainer.java:916) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.access$1600(SimpleMessageListenerContainer.java:83) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer$AsyncMessageProcessingConsumer.mainLoop(SimpleMessageListenerContainer.java:1291) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer$AsyncMessageProcessingConsumer.run(SimpleMessageListenerContainer.java:1197) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at java.base/java.lang.Thread.run(Thread.java:830) ~[na:na]
Caused by: java.lang.RuntimeException: Business Rule Exception
    at br.com.facef.rabbitmqdlq.consumer.MessageConsumer.processOrderMessage(MessageConsumer.java:18) ~[classes/na]
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(Native Method) ~[na:na]
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) ~[na:na]
    at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:na]
    at java.base/java.lang.reflect.Method.invoke(Method.java:567) ~[na:na]
    at org.springframework.messaging.handler.invocation.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:171) ~[spring-messaging-5.2.8.RELEASE;jar:5.2.8.RELEASE]
    at org.springframework.messaging.handler.invocation.InvocableHandlerMethod.invoke(InvocableHandlerMethod.java:120) ~[spring-messaging-5.2.8.RELEASE;jar:5.2.8.RELEASE]
    at org.springframework.amqp.rabbit.listener.adapter.HandlerAdapter.invokeHandlerAndProcessResult(HandlerAdapter.java:53) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.invokeHandler(MessagingMessageListenerAdapter.java:228) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    ... 13 common frames omitted
2020-08-27 22:11:45.414 INFO 25291 --- [IntContainer#0-1] b.c.f.r.consumer.MessageConsumer : Processing message: (Body: 'FAKE-MESSAGE-2020-08-27T22:11:45.39948014050aaa-d1ca-4cbb-b573-159465bd1789' MessageProperties { headers={}, ryMode=PERSISTENT, priority=0, redelivered=false, receivedExchange=order-exchange, receivedRoutingKey=order-messages-queue, deliveryTag=2, consumerTag=amq.ctag-FTBUALpTXXtWm4_0TRTPJw, consumerQueue=order-messages-queue})
2020-08-27 22:11:45.415 WARN 25291 --- [IntContainer#0-1] s.a.r.l.ConditionalRejectingErrorHandler : Execution of Rabbit message listener failed.

org.springframework.amqp.rabbit.support.ListenerExecutionFailedException: Listener method 'public void br.com.facef.rabbitmqdlq.consumer.MessageConsumer.processOrderMessage(org.springframework.amqp.core.Message)' threw exception
    at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.invokeHandler(MessagingMessageListenerAdapter.java:228) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.invokeHandlerAndProcessResult(MessagingMessageListenerAdapter.java:140) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.doInvokeListener(AbstractMessageListenerContainer.java:1591) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.actualInvokeListener(AbstractMessageListenerContainer.java:1510) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.doExecuteListener(AbstractMessageListenerContainer.java:1489) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.executeListener(AbstractMessageListenerContainer.java:1433) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.doReceiveAndExecute(SimpleMessageListenerContainer.java:970) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.receiveAndExecute(SimpleMessageListenerContainer.java:916) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.access$1600(SimpleMessageListenerContainer.java:83) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer$AsyncMessageProcessingConsumer.mainLoop(SimpleMessageListenerContainer.java:1291) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer$AsyncMessageProcessingConsumer.run(SimpleMessageListenerContainer.java:1197) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at java.base/java.lang.Thread.run(Thread.java:830) ~[na:na]
Caused by: java.lang.RuntimeException: Business Rule Exception
    at br.com.facef.rabbitmqdlq.consumer.MessageConsumer.processOrderMessage(MessageConsumer.java:18) ~[classes/na]
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(Native Method) ~[na:na]
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) ~[na:na]
    at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:na]
    at java.base/java.lang.reflect.Method.invoke(Method.java:567) ~[na:na]
    at org.springframework.messaging.handler.invocation.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:171) ~[spring-messaging-5.2.8.RELEASE;jar:5.2.8.RELEASE]
    at org.springframework.messaging.handler.invocation.InvocableHandlerMethod.invoke(InvocableHandlerMethod.java:120) ~[spring-messaging-5.2.8.RELEASE;jar:5.2.8.RELEASE]
    at org.springframework.amqp.rabbit.listener.adapter.HandlerAdapter.invokeHandlerAndProcessResult(HandlerAdapter.java:53) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
    at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.invokeHandler(MessagingMessageListenerAdapter.java:228) ~[spring-rabbit-2.2.10.RELEASE;jar:2.2.10.RELEASE]
```

Para evitarmos o loop-infinito de requeue, podemos desabilitar a configuração via properties ou lançando uma exception padrão para evitar requeue.

```
# To disable requeue enable property as false
spring.rabbitmq.listener.simple.default-requeue-rejected=false
```

```
package br.com.facef.rabbitmqdlq.consumer;

import br.com.facef.rabbitmqdlq.configuration.DirectExchangeConfiguration;
import lombok.extern.slf4j.Slf4j;
import org.springframework.amqp.AmqpRejectAndDontRequeueException;
import org.springframework.amqp.core.Message;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
```

```


import org.springframework.context.annotation.Configuration;

@Configuration
@Slf4j
public class MessageConsumer {

    @RabbitListener(queues = DirectExchangeConfiguration.ORDER_MESSAGES_QUEUE_NAME)
    public void processOrderMessage(Message message) {
        log.info("Processing message: {}", message.toString());
        // By default the messages will be requeued
        // throw new RuntimeException("Business Rule Exception");
        // To dont requeue message can throw AmqpRejectAndDontRequeueException
        throw new AmqpRejectAndDontRequeueException("Business Rule Exception");
    }
}

```

Após o startup da aplicação conseguimos ver a exchange e a fila criada.


RabbitMQ 3.8.7
Erlang 23.0.3

Overview
Connections
Channels
**Exchanges**
Queues
Admin

## Exchanges

▼ All exchanges (8)

Pagination

Page 1 of 1 - Filter:  ☐ Regex ?

| Name               | Type    | Features | Message rate in | Message rate out | +/- |
|--------------------|---------|----------|-----------------|------------------|-----|
| (AMQP default)     | direct  | D        |                 |                  |     |
| amq.direct         | direct  | D        |                 |                  |     |
| amq.fanout         | fanout  | D        |                 |                  |     |
| amq.headers        | headers | D        |                 |                  |     |
| amq.match          | headers | D        |                 |                  |     |
| amq.rabbitmq.trace | topic   | D I      |                 |                  |     |
| amq.topic          | topic   | D        |                 |                  |     |
| order-exchange     | direct  | D        | 0.00/s          | 0.00/s           |     |

► Add a new exchange

## Queues

▼ All queues (1)

Pagination

Page  of 1 - Filter:  ☐ Regexp ?

| Overview             |         |  |   | Messages |         |       | Message rates |               |        | +/- |
|----------------------|---------|--|---|----------|---------|-------|---------------|---------------|--------|-----|
| Name                 | Type    | Features   | State   | Ready    | Unacked | Total | incoming      | deliver / get | ack    |     |
| order-messages-queue | classic | <span style="background-color: #00FFFF; border: 1px solid #00FFFF; padding: 2px;">D</span> | <span style="background-color: #D3D3D3; border: 1px solid #D3D3D3; padding: 2px;">idle</span> | 0        | 0       | 0     | 0.00/s        | 0.00/s        | 0.00/s |     |

► Add a new queue

## Dead Letter Queue

Uma **Dead Letter Queue (DLQ)**, é uma fila que armazena mensagens não entregues ou que tiveram falha durante o processamento, uma DLQ permite lidar com mensagens ruins, monitorar falhas, permitir que o sistema se recupere após falhas ou exceções e o mais importante ajuda a prevenir **loop infinito** devido ao processo de re-enfileiramento.

Existem **dois** principais conceitos quando estudamos DLQ utilizando RabbitMQ: **Dead Letter Exchange(DLX) e Dead Letter Queue(DLQ)**.

DLX é uma exchange normal que podemos criar com os tipos padrão: **topic, direct ou fanout**.

Um ponto importante de entender é que o **producer não deve saber sobre as filas, ele somente deve conhecer as exchanges e todas as mensagens enviadas para a exchange serão roteadas de acordo com a configuração da mensagem e routing-key**.

Vamos adicionar o código referente a configuração da **DLQ**

- **Adjust DLQ name**

```
# To disable requeue enable property as false
spring.rabbitmq.listener.simple.default-requeue-rejected=false
```

```
package br.com.facef.rabbitmqdlq.configuration;

import org.springframework.amqp.core.Binding;
import org.springframework.amqp.core.BindingBuilder;
import org.springframework.amqp.core.DirectExchange;
import org.springframework.amqp.core.ExchangeBuilder;
import org.springframework.amqp.core.Queue;
import org.springframework.amqp.core.QueueBuilder;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
```

```

public class DirectExchangeConfiguration {

    public static final String DIRECT_EXCHANGE_NAME = "order-exchange";
    public static final String ORDER_MESSAGES_QUEUE_NAME = "order-messages-queue";
    public static final String ORDER_MESSAGES_QUEUE_DLQ_NAME =
        ORDER_MESSAGES_QUEUE_NAME.concat(".dlq");

    @Bean
    Queue orderMessagesQueue() {
        return QueueBuilder.durable(ORDER_MESSAGES_QUEUE_NAME)
            .withArgument("x-dead-letter-exchange", "")
            .withArgument("x-dead-letter-routing-key", ORDER_MESSAGES_QUEUE_DLQ_NAME)
            .build();
    }

    @Bean
    Queue orderMessagesDeadLetterQueue() {
        return QueueBuilder.durable(ORDER_MESSAGES_QUEUE_DLQ_NAME).build();
    }

    @Bean
    DirectExchange exchange() {
        return ExchangeBuilder.directExchange(DIRECT_EXCHANGE_NAME).durable(true).build();
    }

    @Bean
    Binding bindingOrderMessagesQueue(
        @Qualifier("orderMessagesQueue") Queue queue, DirectExchange exchange) {
        return BindingBuilder.bind(queue).to(exchange).with(ORDER_MESSAGES_QUEUE_NAME);
    }
}

```

Para executarmos o exemplo iremos **apagar e recriar o container do rabbitmq**.

```

docker-compose stop && docker-compose rm -f && docker-compose up -d

```

Após executar a aplicação novamente com **./mvnw clean spring-boot:run**, iremos perceber que a aplicação ao iniciar envia a mensagem fake para a fila, o consumer ao processar lança a exception de validação de negócio e com a propriedade de **requeue** desabilitada a mensagem vai para a fila de **DLQ**.

```
11 Spring Boot 11 (v2.3.3.RELEASE)
2020-08-27 22:41:02.959 INFO 27112 [main] b.c.f.r.RabbitmqApplication : Starting RabbitmqApplication on Cassios-MacBook-Air.local with PID 27112 (/Users/cassiothadeu/Work/pos-desenvolvimento-web/messengeria-streams/rabbitmq-
thadeu/Work/pos-desenvolvimento-web/messengeria-streams/rabbitmq-dlq)
2020-08-27 22:41:02.952 INFO 27112 [main] b.c.f.r.RabbitmqApplication : No active profile set, falling back to default profiles: default
2020-08-27 22:41:04.759 INFO 27112 [main] o.s.b.w.e.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-08-27 22:41:04.792 INFO 27112 [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-08-27 22:41:04.793 INFO 27112 [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.37]
2020-08-27 22:41:04.998 INFO 27112 [main] o.s.c.g.c.tomcat.Localhost1/r/ : Initializing Spring embedded WebApplicationContext
2020-08-27 22:41:04.998 INFO 27112 [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1972 ms
2020-08-27 22:41:05.949 INFO 27112 [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-08-27 22:41:06.348 INFO 27112 [main] o.s.b.w.e.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2020-08-27 22:41:06.342 INFO 27112 [main] o.s.a.r.c.CachingConnectionFactory : Attempting to connect to: [localhost:5672]
2020-08-27 22:41:06.492 INFO 27112 [main] o.s.a.r.c.CachingConnectionFactory : Created new connection: rabbitConnectionFactory#2330a3e8/0/SingleConnection@11eed657 [delegate=amp://guest@127.0.0.1:5672/, localPort= 57134]
2020-08-27 22:41:06.632 INFO 27112 [main] b.c.f.r.RabbitmqApplication : Started RabbitmqApplication in 9.627 seconds (JVM running for 18.219)
2020-08-27 22:41:06.638 INFO 27112 [main] b.c.f.r.RabbitmqApplication : Running method after startup to send messages!
2020-08-27 22:41:06.638 INFO 27112 [main] b.c.f.r.producer.MessageProducer : Sending 9 fake message...
2020-08-27 22:41:06.672 INFO 27112 [main] b.c.f.r.consumer.MessageConsumer : Processing message: [Body:FAKE-MESSAGE-2020-08-27T22:41:06.64124442a652e-df9d-411d-9f7a-4d889d4f9bb5 MessageProperties {headers={}, contentType=text/pla
ryMode=PERSISTENT, priority=, redelivered=false, receivedExchange=order-exchange, receivedRoutingKey=order-messages-queue, deliveryTag=1, consumerTag=amq.ctag-JCYdJngc74oQNCihmeUa, consumerQueue=order-messages-queue}]
2020-08-27 22:41:06.682 INFO 27112 [main] b.c.f.r.c.ConditionalRejectingErrorHandler : Execution of Rabbit message listener failed.
org.springframework.amqp.rabbit.support.ListenerExecutionFailedException: Listener method 'public void br.com.facef.rabbitmqdlq.consumer.MessageConsumer.processOrderMessage(org.springframework.amqp.core.Message)' threw exception
at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.invokeHandler(MessagingMessageListenerAdapter.java:228) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.invokeHandlerAndProcessResult(MessagingMessageListenerAdapter.java:148) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.onMessage(MessagingMessageListenerAdapter.java:133) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.doInvokeListener(AbstractMessageListenerContainer.java:1593) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.actualInvokeListener(AbstractMessageListenerContainer.java:1518) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.invokeListener(AbstractMessageListenerContainer.java:1498) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.executeListener(AbstractMessageListenerContainer.java:1469) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.doInvokeListener(AbstractMessageListenerContainer.java:1433) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.doReceiveAndExecute(SimpleMessageListenerContainer.java:978) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.receiveAndExecute(SimpleMessageListenerContainer.java:951) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.access$1000(SimpleMessageListenerContainer.java:83) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer$AsyncMessageProcessingConsumer.mainLoop(SimpleMessageListenerContainer.java:1291) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer$AsyncMessageProcessingConsumer.run(SimpleMessageListenerContainer.java:1197) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at java.base/java.lang.Thread.run(Thread.java:830) ~[na:na]
Caused by: java.lang.RuntimeException: Business Rule Exception
at br.com.facef.rabbitmqdlq.consumer.MessageConsumer.processOrderMessage(MessageConsumer.java:18) ~[classes/na]
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(Native Method) ~[na:na]
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) ~[na:na]
at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:na]
at java.base/java.lang.reflect.Method.invoke(Method.java:567) ~[na:na]
at org.springframework.messaging.handler.invocation.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:171) ~[spring-messaging-5.2.8.RELEASE.jar:5.2.8.RELEASE]
at org.springframework.messaging.handler.invocation.InvocableHandlerMethod.invoke(InvocableHandlerMethod.java:120) ~[spring-messaging-5.2.8.RELEASE.jar:5.2.8.RELEASE]
at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.invokeHandlerAdapter(MessagingMessageListenerAdapter.java:133) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
at org.springframework.amqp.rabbit.listener.adapter.MessagingMessageListenerAdapter.invokeHandler(MessagingMessageListenerAdapter.java:220) ~[spring-rabbit-2.2.10.RELEASE.jar:2.2.10.RELEASE]
... 33 common frames omitted
```



RabbitMQ 3.8.7 Erlang 23.0.3

Overview Connections Channels Exchanges **Queues** Admin

## Queues

► All queues (2)

| Overview                 |         | Messages  |       |       |         |       | Message rates |               |        | +/- |
|--------------------------|---------|-----------|-------|-------|---------|-------|---------------|---------------|--------|-----|
| Name                     | Type    | Features  | State | Ready | Unacked | Total | incoming      | deliver / get | ack    |     |
| order-messages-queue     | classic | D DLX DLK | idle  | 0     | 0       | 0     | 0.00/s        | 0.00/s        | 0.00/s |     |
| order-messages-queue.dlq | classic | D         | idle  | 1     | 0       | 1     |               |               |        |     |

► Add a new queue

## Parking Lot Queue

Podemos ter um cenário onde não podemos descartar uma mensagem, pois a mensagem pode ser de uma transação de pagamento ou um pedido de um cliente, em alguns casos as mensagens irão precisar de algum tipo de intervenção manual ou até mesmo armazenar as mensagens depois de terem ocorrido erros por **X vezes**.

Para esses cenários dentro do RabbitMQ existe o conceito de **ParkingLot queue**, um cenários comum é processar as mensagens das filas **DLQ** e após terem atingindo um limite de falhas enviar as mensagens para as filas **parking lot** para efetuar um processamento posterior.