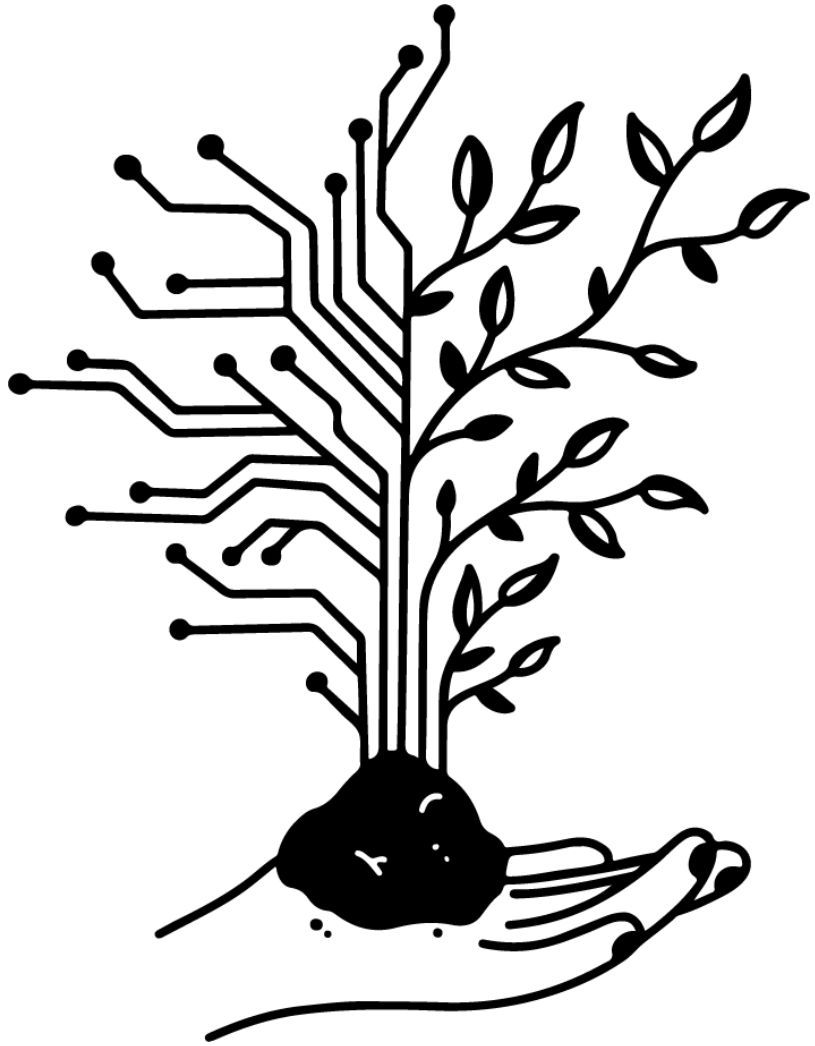
CodeBuzz 2025

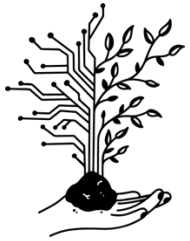
Follow-Along Guide

05.06.2025

Steffen Börner



stockprice-monitor > StockPriceMonitorApplication

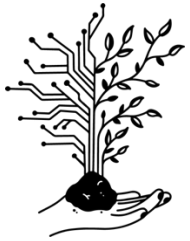


1

`StockPriceMonitor` **starten**

```java
@SpringBootApplication
@EnableScheduling
public class StockpriceMonitorApplication {

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


}
```

trading-platform > TradingPlatformApplication

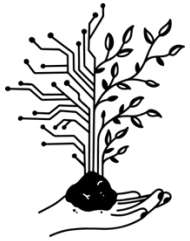


2

TradingPlatform **starten**

```java
@SpringBootApplication
// @EnableRetry
public class TradingPlatformApplication {

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

}
```

stockprice-monitor > StockPriceUpdateEvent

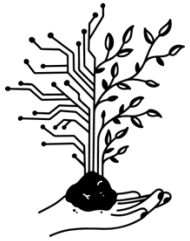


3

```java
record StockPriceUpdateEvent(String symbol, double priceInCents) {

}

// record StockPriceUpdateEvent(String symbol, double priceInCents, String name) {

// }
```

```java
// record StockPriceUpdateEvent(String symbol, double priceInCents) {

// }

record StockPriceUpdateEvent(String symbol, double priceInCents, String name) {

}
```

stockprice-monitor > StockPriceMonitor

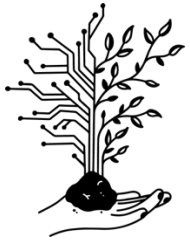


4

```
39  @Scheduled(fixedRate = 5000)
40  void monitorStockPrice() throws UnknownHostException, JsonProcessingException, NumberFormatException {
41      String stockPrice = restTemplate.getForObject(url, String.class);
42      LOG.info("Stock price received: " + stockPrice);
43
44      StockPriceUpdateEvent event = new StockPriceUpdateEvent("A0DJ5J",
45              Double.parseDouble(stockPrice));
46      rabbitTemplate.convertAndSend("stockPriceUpdate", "",
47              event);
48      LOG.info("Event published: {}", event);
```

```
39  @Scheduled(fixedRate = 5000)
40  void monitorStockPrice() throws UnknownHostException, JsonProcessingException, NumberFormatException {
41      String stockPrice = restTemplate.getForObject(url, String.class);
42      LOG.info("Stock price received: " + stockPrice);
43
44      StockPriceUpdateEvent event = new StockPriceUpdateEvent("A0DJ5J",
45              Double.parseDouble(stockPrice), "Some stock");
46      rabbitTemplate.convertAndSend("stockPriceUpdate", "",
47              event);
48      LOG.info("Event published: {}", event);
```

stockprice-monitor > RabbitMQConfig

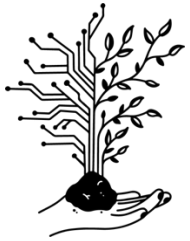


5

```
39  // @Bean
40  // FanoutExchange fanoutV2() {
41  //     return new FanoutExchange("stockPriceUpdateV2");
42  // }
```

```
39  @Bean
40  FanoutExchange fanoutV2() {
41      return new FanoutExchange("stockPriceUpdateV2");
42  }
```

stockprice-monitor > StockPriceMonitor

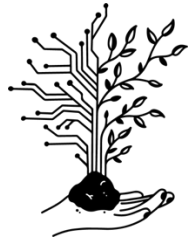


6

```
44  StockPriceUpdateEvent event = new StockPriceUpdateEvent("A0DJ5J",
45          Double.parseDouble(stockPrice), "Some stock");
46  rabbitTemplate.convertAndSend("stockPriceUpdate", "",
47          event);
48  LOG.info("Event published: {}", event);
49
50  // StockPriceUpdateEventV2 eventV2 = new StockPriceUpdateEventV2(new
51  // Symbol(UUID.randomUUID().toString(), "A0DJ5J"),
52  // Double.parseDouble(stockPrice), "Some stock");
53  // rabbitTemplate.convertAndSend("stockPriceUpdateV2", "",
54  // eventV2);
55  // LOG.info("Event V2 published: {}", eventV2);
```

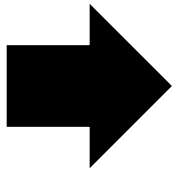
```
44  StockPriceUpdateEvent event = new StockPriceUpdateEvent("A0DJ5J",
45          Double.parseDouble(stockPrice), "Some stock");
46  rabbitTemplate.convertAndSend("stockPriceUpdate", "",
47          event);
48  LOG.info("Event published: {}", event);
49
50  StockPriceUpdateEventV2 eventV2 = new StockPriceUpdateEventV2(
51          new Symbol(UUID.randomUUID().toString(), "A0DJ5J"),
52          Double.parseDouble(stockPrice), "Some stock");
53  rabbitTemplate.convertAndSend("stockPriceUpdateV2", "",
54          eventV2);
55  LOG.info("Event V2 published: {}", eventV2);
```

trading-platform > RabbitMQConfig



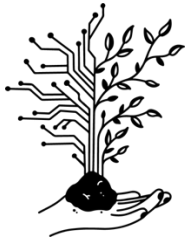
7

```java
@Bean
public FanoutExchange fanoutExchange() {
    return new FanoutExchange("stockPriceUpdate");
}

@Bean
public Queue stockPriceQueue() {
    return new Queue("tradingPlattformStockPriceUpdate", false);
}

// @Bean
// public FanoutExchange fanoutExchange() {
// return new FanoutExchange("stockPriceUpdateV2");
// }

// @Bean
// public Queue stockPriceQueue() {
// return new Queue("tradingPlattformStockPriceUpdateV2", false);
// }

```



```java
// @Bean
// public FanoutExchange fanoutExchange() {
// return new FanoutExchange("stockPriceUpdate");
// }

// @Bean
// public Queue stockPriceQueue() {
// return new Queue("tradingPlattformStockPriceUpdate", false);
// }

@Bean
public FanoutExchange fanoutExchange() {
    return new FanoutExchange("stockPriceUpdateV2");
}

@Bean
public Queue stockPriceQueue() {
    return new Queue("tradingPlattformStockPriceUpdateV2", false);
}
```

trading-platform > StockUpdateEventListener

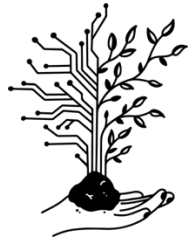


8

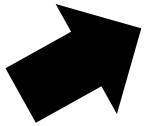
```java
@RabbitListener(queues = "tradingPlattformStockPriceUpdate")
public void handleStockPriceUpdate(StockPriceUpdateEvent message) {
    // Process the stock price update message
    LOG.info("Received stock price update: {}", message);
}

// @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
// public void handleStockPriceUpdate(StockPriceUpdateEventV2 message) {
// LOG.info("Received stock price update v2: {}", message);
// }
```

```java
// @RabbitListener(queues = "tradingPlattformStockPriceUpdate")
// public void handleStockPriceUpdate(StockPriceUpdateEvent message) {
// // Process the stock price update message
// LOG.info("Received stock price update: {}", message);
// }

@RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
public void handleStockPriceUpdate(StockPriceUpdateEventV2 message) {
    LOG.info("Received stock price update v2: {}", message);
}
```

stockprice-monitor > RabbitMQConfig



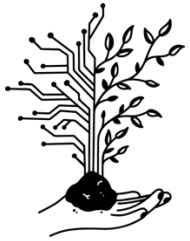
9



```java
// @Bean
// RabbitTemplate rabbitTemplate(final CachingConnectionFactory connectionFactory) {
//     RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
//     rabbitTemplate.setMessageConverter(messageConverter());
//     return rabbitTemplate;
// }

@Bean
FanoutExchange fanoutV2() {
    return new FanoutExchange("stockPriceUpdateV2");
}

@Bean
RabbitTemplate rabbitTemplate(final CachingConnectionFactory connectionFactory) {
    RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
    rabbitTemplate.setMessageConverter(messageConverter());
    connectionFactory.setPublisherConfirmType(ConfirmType.CORRELATED);
    rabbitTemplate.setConfirmCallback((correlationData, ack, cause) -> {
        if (ack) {
            System.out.println("Message published successfully: " + (correlationData != null ? correlationData.getId() : "null"));
        } else {
            System.err.println("Message publish failed: " + (correlationData != null ? correlationData.getId() : "null") + ", cause: " + cause);
        }
    });
    return rabbitTemplate;
}
```

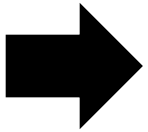

```java
@Bean
RabbitTemplate rabbitTemplate(final CachingConnectionFactory connectionFactory) {
    RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
    rabbitTemplate.setMessageConverter(messageConverter());
    return rabbitTemplate;
}

@Bean
FanoutExchange fanoutV2() {
    return new FanoutExchange("stockPriceUpdateV2");
}

// @Bean
// RabbitTemplate rabbitTemplate(final CachingConnectionFactory connectionFactory) {
//     RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
//     rabbitTemplate.setMessageConverter(messageConverter());
//     connectionFactory.setPublisherConfirmType(ConfirmType.CORRELATED);
//     rabbitTemplate.setConfirmCallback((correlationData, ack, cause) -> {
//         if (ack) {
//             System.out.println("Message published successfully: " + (correlationData != null ? correlationData.getId() : "null"));
//         } else {
//             System.err.println("Message publish failed: " + (correlationData != null ? correlationData.getId() : "null") + ", cause: " + cause);
//         }
//     });
//     return rabbitTemplate;
// }
```

stockprice-monitor > StockPriceMonitor



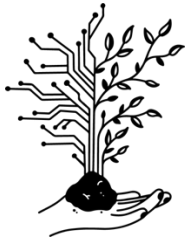
10

```
50  StockPriceUpdateEventV2 eventV2 = new StockPriceUpdateEventV2(
51          new Symbol(UUID.randomUUID().toString(), "A0DJ5J"),
52          Double.parseDouble(stockPrice), "Some stock");
53  rabbitTemplate.convertAndSend("stockPriceUpdateV2", "",
54          eventV2);
55  LOG.info("Event V2 published: {}", eventV2);
56
57  // StockPriceUpdateEventV2 eventV2 = new StockPriceUpdateEventV2(
58  // new Symbol(UUID.randomUUID().toString(), "A0DJ5J"),
59  // Double.parseDouble(stockPrice), "Some stock");
60  // CorrelationData correlationData = new CorrelationData(eventV2.toString());
61
62  // rabbitTemplate.convertAndSend(
63  // "stockPriceUpdateV2",
64  // "",
65  // eventV2, correlationData);
```



```
50  // StockPriceUpdateEventV2 eventV2 = new StockPriceUpdateEventV2(
51  // new Symbol(UUID.randomUUID().toString(), "A0DJ5J"),
52  // Double.parseDouble(stockPrice), "Some stock");
53  // rabbitTemplate.convertAndSend("stockPriceUpdateV2", "",
54  // eventV2);
55  // LOG.info("Event V2 published: {}", eventV2);
56
57  StockPriceUpdateEventV2 eventV2 = new StockPriceUpdateEventV2(
58          new Symbol(UUID.randomUUID().toString(), "A0DJ5J"),
59          Double.parseDouble(stockPrice), "Some stock");
60  CorrelationData correlationData = new CorrelationData(eventV2.toString());
61
62  rabbitTemplate.convertAndSend(
63          "stockPriceUpdateV2",
64          "",
65          eventV2, correlationData);
```

stockprice-monitor > StockPriceMonitor

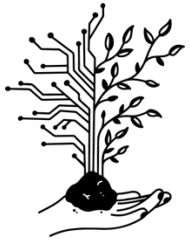


11

```
57  StockPriceUpdateEventV2 eventV2 = new StockPriceUpdateEventV2(
58          new Symbol(UUID.randomUUID().toString(), "A0DJ5J"),
59          Double.parseDouble(stockPrice), "Some stock");
60  CorrelationData correlationData = new CorrelationData(eventV2.toString());
61
62  rabbitTemplate.convertAndSend(
63          "stockPriceUpdateV2",
64          "",
65          eventV2, correlationData);
```

```
57  StockPriceUpdateEventV2 eventV2 = new StockPriceUpdateEventV2(
58          new Symbol(UUID.randomUUID().toString(), "A0DJ5J"),
59          Double.parseDouble(stockPrice), "Some stock");
60  CorrelationData correlationData = new CorrelationData(eventV2.toString());
61
62  rabbitTemplate.convertAndSend(
63          "ABC_NON_EXISTING",
64          "",
65          eventV2, correlationData);
```

stockprice-monitor > StockPriceMonitor

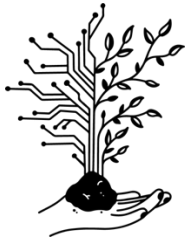


12

```
57  StockPriceUpdateEventV2 eventV2 = new StockPriceUpdateEventV2(
58          new Symbol(UUID.randomUUID().toString(), "A0DJ5J"),
59          Double.parseDouble(stockPrice), "Some stock");
60  CorrelationData correlationData = new CorrelationData(eventV2.toString());
61
62  rabbitTemplate.convertAndSend(
63          "ABC_NON_EXISTING",
64          "",
65          eventV2, correlationData);
```

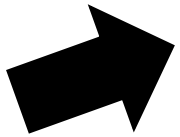
```
57  StockPriceUpdateEventV2 eventV2 = new StockPriceUpdateEventV2(
58          new Symbol(UUID.randomUUID().toString(), "A0DJ5J"),
59          Double.parseDouble(stockPrice), "Some stock");
60  CorrelationData correlationData = new CorrelationData(eventV2.toString());
61
62  rabbitTemplate.convertAndSend(
63          "stockPriceUpdateV2",
64          "",
65          eventV2, correlationData);
```

trading-platform > StockUpdateEventListener



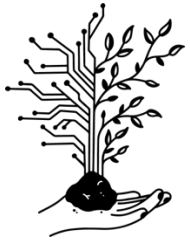
13

```java
@RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
public void handleStockPriceUpdate(StockPriceUpdateEventV2 message) {
    LOG.info("Received stock price update v2: {}", message);
}

// @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
// @Retryable(maxAttempts = 3, backoff =
// @org.springframework.retry.annotation.Backoff(delay = 4000))
// public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message)
// {
// LOG.info("Received stock price update v2: {}", message);
// // Simulate processing logic that might fail
// // For demonstration, we throw an exception to trigger the retry mechanism
// throw new RuntimeException("Simulated error in StockUpdateEventListener");
// }
```



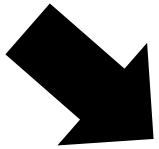
```java
// @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
// public void handleStockPriceUpdate(StockPriceUpdateEventV2 message) {
// LOG.info("Received stock price update v2: {}", message);
// }

@RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message) {
    LOG.info("Received stock price update v2: {}", message);
    // Simulate processing logic that might fail
    // For demonstration, we throw an exception to trigger the retry mechanism
    throw new RuntimeException("Simulated error in StockUpdateEventListener");
}
```

trading-platform > StockUpdateEventListener



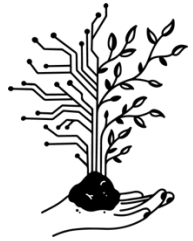
14

```
22  // @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
23  // public void handleStockPriceUpdate(StockPriceUpdateEventV2 message) {
24  // LOG.info("Received stock price update v2: {}", message);
25  // }
26
27  @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
28  public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message) {
29      LOG.info("Received stock price update v2: {}", message);
30      // Simulate processing logic that might fail
31      // For demonstration, we throw an exception to trigger the retry mechanism
32      throw new RuntimeException("Simulated error in StockUpdateEventListener");
33  }
```



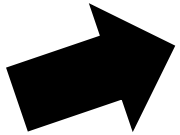
```
27  @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
28  public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message) {
29      LOG.info("Received stock price update v2: {}", message);
30      // Simulate processing logic that might fail
31      // For demonstration, we throw an exception to trigger the retry mechanism
32      throw new AmqpRejectAndDontRequeueException("Simulated error in StockUpdateEventListener");
33  }
```

trading-platform > TradingPlatformApplication



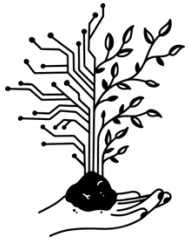
15

```java
@SpringBootApplication
// @EnableRetry
public class TradingPlatformApplication {

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

}
```



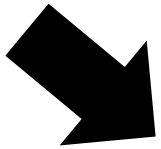
```java
@SpringBootApplication
@EnableRetry
public class TradingPlatformApplication {

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

}
```

trading-platform > TradingPlatformApplication



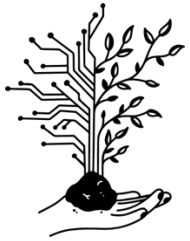
16

```java
@RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message) {
    LOG.info("Received stock price update v2: {}", message);
    // Simulate processing logic that might fail
    // For demonstration, we throw an exception to trigger the retry mechanism
    throw new AmqpRejectAndDontRequeueException("Simulated error in StockUpdateEventListener");
}

// @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
// @Retryable(maxAttempts = 3, backoff =
// @org.springframework.retry.annotation.Backoff(delay = 4000))
// public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message)
// {
// LOG.info("Received stock price update v2: {}", message);
// // Simulate processing logic that might fail
// // For demonstration, we throw an exception to trigger the retry mechanism
// throw new RuntimeException("Simulated error in StockUpdateEventListener");
// }
```



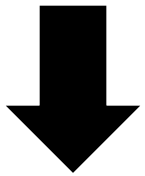
```java
// @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
// public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message) {
//     LOG.info("Received stock price update v2: {}", message);
//     // Simulate processing logic that might fail
//     // For demonstration, we throw an exception to trigger the retry mechanism
//     throw new AmqpRejectAndDontRequeueException("Simulated error in StockUpdateEventListener");
// }

@RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
@Retryable(maxAttempts = 3, backoff = @org.springframework.retry.annotation.Backoff(delay = 4000))
public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message) {
    LOG.info("Received stock price update v2: {}", message);
    // Simulate processing logic that might fail
    // For demonstration, we throw an exception to trigger the retry mechanism
    throw new RuntimeException("Simulated error in StockUpdateEventListener");
}
```

trading-platform > StockUpdateEventListener



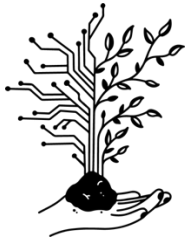
17

```
36  // @Recover
37  // public void recover(RuntimeException e, StockPriceUpdateEventV2 message) {
38  // LOG.error("Failed to process stock price update after retries: {}", message,
39  // e);
40  // // default: ack
41  // // with the following, the message will not be requeued:
42  // throw new AmqpRejectAndDontRequeueException("Failed to process stock price
43  // update", e);
44  // }
```



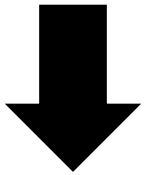
```
36  @Recover
37  public void recover(RuntimeException e, StockPriceUpdateEventV2 message) {
38      LOG.error("Failed to process stock price update after retries: {}", message,
39              e);
40      throw new AmqpRejectAndDontRequeueException("Failed to process stock price update", e);
41  }
```

trading-platform > „Terminal“



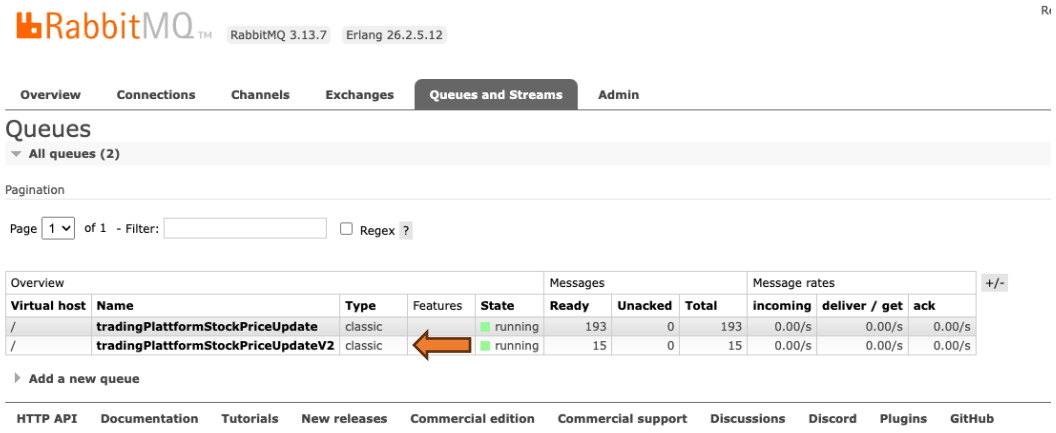
18

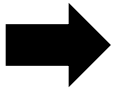
STRG+C

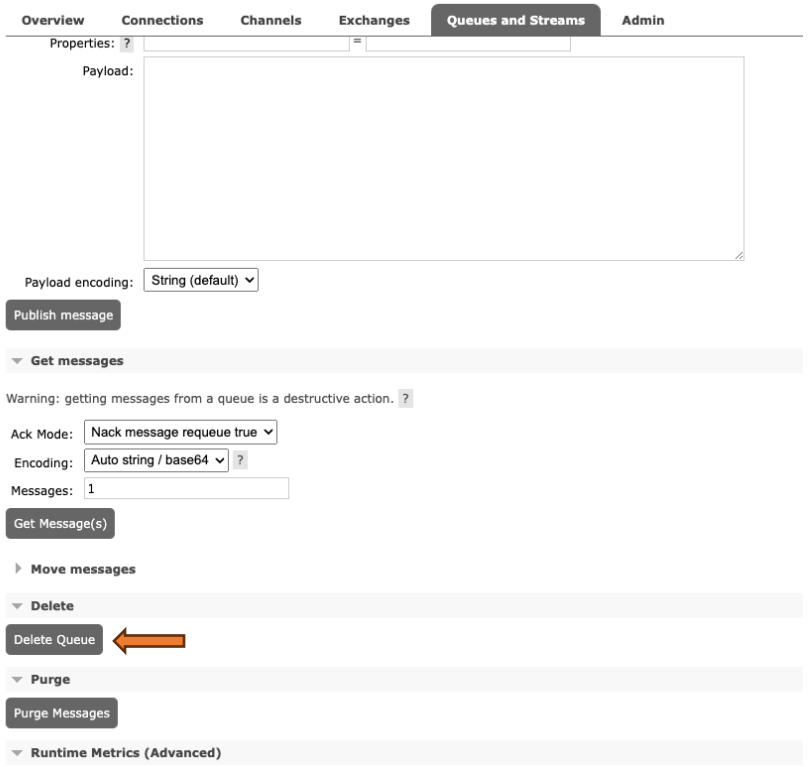


```
5]
        at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.doReceiveAndExecute(SimpleMessageListenerContainer.java:1085) ~[spring-rabbit-3.2.5.jar:3.2.
5]
        at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.receiveAndExecute(SimpleMessageListenerContainer.java:1021) ~[spring-rabbit-3.2.5.jar:3.2.5]
        at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer$AsyncMessageProcessingConsumer.mainLoop(SimpleMessageListenerContainer.java:1423) ~[spring-r
abbit-3.2.5.jar:3.2.5]
        at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer$AsyncMessageProcessingConsumer.run(SimpleMessageListenerContainer.java:1324) ~[spring-rabbit
-3.2.5.jar:3.2.5]
        at java.base/java.lang.Thread.run(Thread.java:1447) ~[na:na]

2025-06-02T06:30:15.171Z  WARN 4014 --- [trading-platform] [ntContainer#0-1] o.s.a.r.l.SimpleMessageListenerContainer : Rejecting received message(s) because the listener cont
ainer has been stopped: (Body:'[B@63ca41ad(byte[125])' MessageProperties [headers={spring_returned_message_correlation=StockPriceUpdateEventV2[symbol=Symbol[id=32496366-0a15-4
9d1-a42d-ae4289daa718, name=A0DJ5J], priceInCents=903.2784843672245, name=Some stock], __TypeId__=com.exxeta.codebuzz.stockpricemonitor.StockPriceUpdateEventV2}, contentType=a
pplication/json, contentEncoding=UTF-8, contentLength=0, receivedDeliveryMode=PERSISTENT, priority=0, redelivered=true, receivedExchange=stockPriceUpdateV2, receivedRoutingKey
=, deliveryTag=3, consumerTag=amq.ctag-1QXwno7y6LKJajWJRdBNhQ, consumerQueue=tradingPlattformStockPriceUpdateV2])
2025-06-02T06:30:15.172Z  INFO 4014 --- [trading-platform] [ntContainer#0-2] o.s.a.r.l.SimpleMessageListenerContainer : Successfully waited for workers to finish.
2025-06-02T06:30:15.175Z  INFO 4014 --- [trading-platform] [ionShutdownHook] o.s.b.w.e.tomcat.GracefulShutdown        : Commencing graceful shutdown. Waiting for active reques
ts to complete
2025-06-02T06:30:15.177Z  INFO 4014 --- [trading-platform] [tomcat-shutdown] o.s.b.w.e.tomcat.GracefulShutdown        : Graceful shutdown complete
vscode ➜ /workspaces/eventdriven-spring (main) $
```

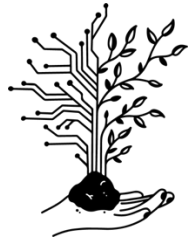
Rabbitmq (15672) > RabbitMQ Management UI





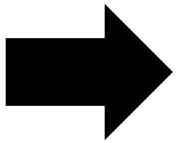


trading-platform > RabbitMQConfig



20

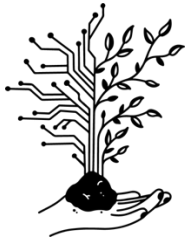
```
// @Bean
// public Queue deadLetterQueue() {
// return new Queue("tradingPlattformStockPriceUpdateV2.dlq", false);
// }

// @Bean
// public Binding deadLetterBinding(Queue deadLetterQueue) {
// return BindingBuilder
// .bind(deadLetterQueue)
// .to(dlx())
// .with("");
// }

// @Bean
// DirectExchange dlx() {
// return new DirectExchange("tradingPlatformStockPriceUpdate.dlx");
// }

// @Bean
// public Queue stockPriceQueue() {
// return QueueBuilder
// .nonDurable("tradingPlattformStockPriceUpdateV2")
// .withArgument("x-dead-letter-exchange",
// "tradingPlatformStockPriceUpdate.dlx")
// .withArgument("x-dead-letter-routing-key", "")
// .build();
// }
```



```
// @Bean
// public Queue stockPriceQueue() {
//     return new Queue("tradingPlattformStockPriceUpdateV2", false);
// }
```

```
@Bean
public Queue deadLetterQueue() {
    return new Queue("tradingPlattformStockPriceUpdateV2.dlq", false);
}

@Bean
public Binding deadLetterBinding(Queue deadLetterQueue) {
    return BindingBuilder
            .bind(deadLetterQueue)
            .to(dlx())
            .with("");
}

@Bean
DirectExchange dlx() {
    return new DirectExchange("tradingPlatformStockPriceUpdate.dlx");
}

@Bean
public Queue stockPriceQueue() {
    return QueueBuilder
            .nonDurable("tradingPlattformStockPriceUpdateV2")
            .withArgument("x-dead-letter-exchange",
                    "tradingPlatformStockPriceUpdate.dlx")
            .withArgument("x-dead-letter-routing-key", "")
            .build();
}
```

trading-platform > TradingPlatformApplication

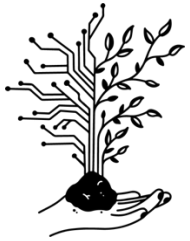


21

TradingPlatformApplication
starten

```java
@SpringBootApplication
@EnableRetry
public class TradingPlatformApplication {

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

}
```

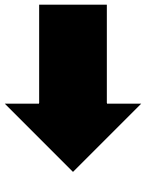
stockprice-monitor > „Terminal“



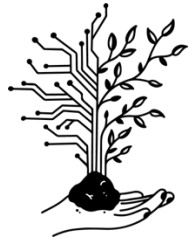
22

StockPriceMonitor
beenden

STRG+C



```
Message published successfully: StockPriceUpdateEventV2[symbol=Symbol[id=873b8722-8ef6-459b-8dd9-2351cb361cd0, name=A0DJ5J], priceInCents=734.6355940345718, name=Some stock]
2025-06-02T06:34:49.754Z  INFO 5864 --- [stockprice-monitor] [   scheduling-1] c.e.c.s.StockPriceMonitor               : Stock price received: 430.41967412221595
2025-06-02T06:34:49.755Z  INFO 5864 --- [stockprice-monitor] [   scheduling-1] c.e.c.s.StockPriceMonitor               : Event published: StockPriceUpdateEvent[symbol=A0DJ5J, priceInCents=430.41967412221595, name=Some stock]
Message published successfully: StockPriceUpdateEventV2[symbol=Symbol[id=67c5b85c-ba5f-47da-88cf-a2e920f1d0f7, name=A0DJ5J], priceInCents=430.41967412221595, name=Some stock]
Message published successfully: null
2025-06-02T06:34:54.755Z  INFO 5864 --- [stockprice-monitor] [   scheduling-1] c.e.c.s.StockPriceMonitor               : Stock price received: 992.5940971285239
2025-06-02T06:34:54.756Z  INFO 5864 --- [stockprice-monitor] [   scheduling-1] c.e.c.s.StockPriceMonitor               : Event published: StockPriceUpdateEvent[symbol=A0DJ5J, priceInCents=992.5940971285239, name=Some stock]
Message published successfully: null
Message published successfully: StockPriceUpdateEventV2[symbol=Symbol[id=238f44a4-4f7f-4e8c-b3e6-0b1fe28c0dce, name=A0DJ5J], priceInCents=992.5940971285239, name=Some stock]
2025-06-02T06:34:59.754Z  INFO 5864 --- [stockprice-monitor] [   scheduling-1] c.e.c.s.StockPriceMonitor               : Stock price received: 849.8983124683871
2025-06-02T06:34:59.755Z  INFO 5864 --- [stockprice-monitor] [   scheduling-1] c.e.c.s.StockPriceMonitor               : Event published: StockPriceUpdateEvent[symbol=A0DJ5J, priceInCents=849.8983124683871, name=Some stock]
Message published successfully: null
Message published successfully: StockPriceUpdateEventV2[symbol=Symbol[id=c49f78ff-2f45-4a0a-9738-8713832c5aad, name=A0DJ5J], priceInCents=849.8983124683871, name=Some stock]
^C2025-06-02T06:35:04.643Z  INFO 5864 --- [stockprice-monitor] [ionShutdownHook] o.s.b.w.e.tomcat.GracefulShutdown        : Commencing graceful shutdown. Waiting for active requests to complete
2025-06-02T06:35:04.645Z  INFO 5864 --- [stockprice-monitor] [tomcat-shutdown] o.s.b.w.e.tomcat.GracefulShutdown        : Graceful shutdown complete
vscode ➜ /workspaces/eventdriven-spring (main) $
```



23

RabbitMQ 15672

Nachrichten
liegen in der DLQ