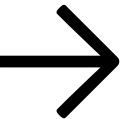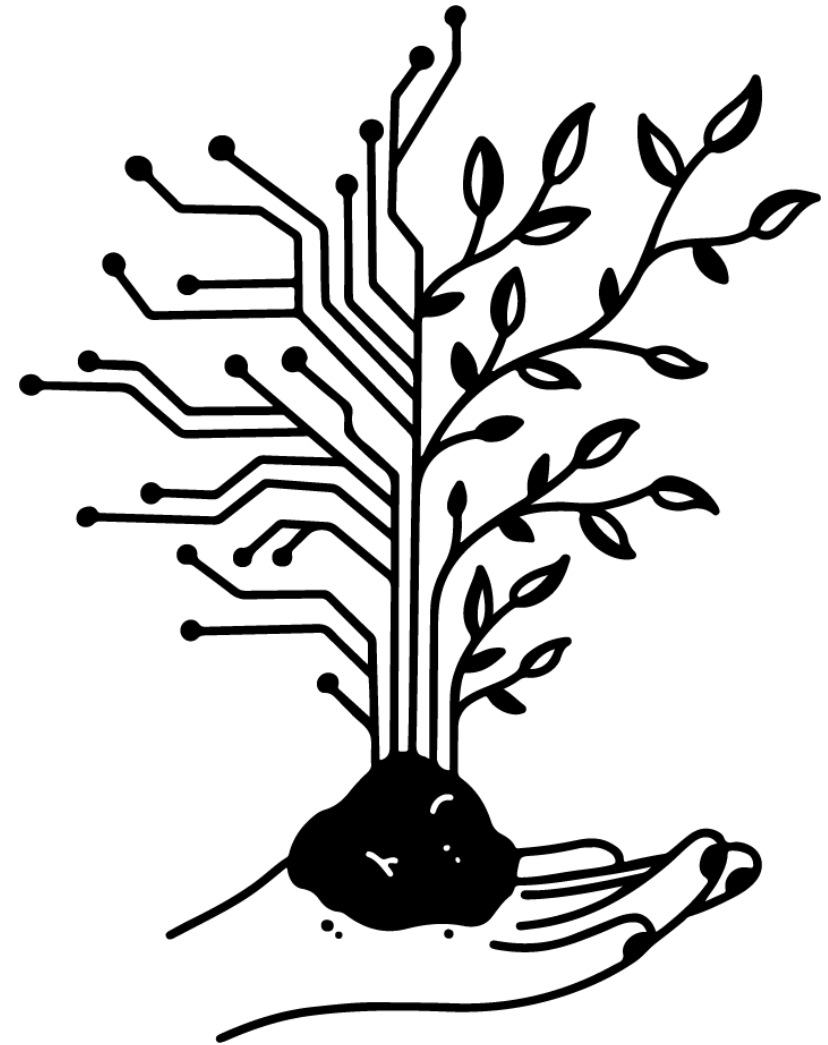CodeBuzz 2025

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);
    }


}
```

exeta

2

TradingPlatform **starten**

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

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

}
```

stockprice-monitor > StockPriceUpdateEvent

3

```
1  record StockPriceUpdateEvent(String symbol, double priceInCents) {
2
3  }
4
5  // record StockPriceUpdateEvent(String symbol, double priceInCents, String name) {
6
7  // }
```
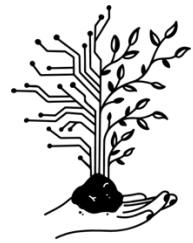
⬇
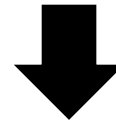
```
1  // record StockPriceUpdateEvent(String symbol, double priceInCents) {
2
3  // }
4
5  record StockPriceUpdateEvent(String symbol, double priceInCents, String name) {
6
7  }
```

stockprice-monitor > StockPriceMonitor

```
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);
```

4

exeta

5

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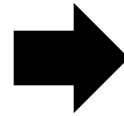
```
39  @Bean
40  FanoutExchange fanoutV2() {
41       return new FanoutExchange("stockPriceUpdateV2");
42  }
```

exeta

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);
```
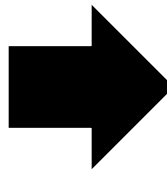
```
44  StockPriceUpdateEvent event = new StockPriceUpdateEvent("A0DJ5J",
45          Double.parseDouble(stockPrice), "Some stock");
46  rabbitTemplate.convertAndSend("stockPriceUpdate", "",
47          event);
48  LOG.info("Event published: {}", event);
49
50  StockPriceUpdateEventV2 eventV2 = new StockPriceUpdateEventV2(
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);
```

exeta

trading-platform > RabbitMQConfig

7

```
19   @Bean
20   public FanoutExchange fanoutExchange() {
21       return new FanoutExchange("stockPriceUpdate");
22   }
23
24   @Bean
25   public Queue stockPriceQueue() {
26       return new Queue("tradingPlattformStockPriceUpdate", false);
27   }
28
29   // @Bean
30   // public FanoutExchange fanoutExchange() {
31   // return new FanoutExchange("stockPriceUpdateV2");
32   // }
33
34   // @Bean
35   // public Queue stockPriceQueue() {
36   // return new Queue("tradingPlattformStockPriceUpdateV2", false);
37   // }
38
```

```
19   // @Bean
20   // public FanoutExchange fanoutExchange() {
21   // return new FanoutExchange("stockPriceUpdate");
22   // }
23
24   // @Bean
25   // public Queue stockPriceQueue() {
26   // return new Queue("tradingPlattformStockPriceUpdate", false);
27   // }
28
29   @Bean
30   public FanoutExchange fanoutExchange() {
31       return new FanoutExchange("stockPriceUpdateV2");
32   }
33
34   @Bean
35   public Queue stockPriceQueue() {
36       return new Queue("tradingPlattformStockPriceUpdateV2", false);
37   }
```
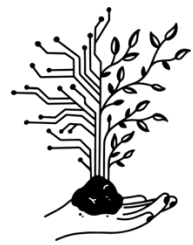
exeta

trading-platform > StockUpdateEventListener
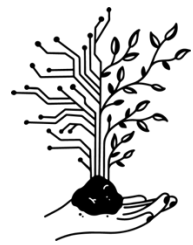
8

```
16  @RabbitListener(queues = "tradingPlattformStockPriceUpdate")
17  public void handleStockPriceUpdate(StockPriceUpdateEvent message) {
18      // Process the stock price update message
19      LOG.info("Received stock price update: {}", message);
20  }
21
22  // @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
23  // public void handleStockPriceUpdate(StockPriceUpdateEventV2 message) {
24  // LOG.info("Received stock price update v2: {}", message);
25  // }
```
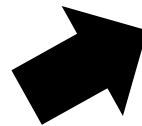
```
16  // @RabbitListener(queues = "tradingPlattformStockPriceUpdate")
17  // public void handleStockPriceUpdate(StockPriceUpdateEvent message) {
18  // // Process the stock price update message
19  // LOG.info("Received stock price update: {}", message);
20  // }
21
22  @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
23  public void handleStockPriceUpdate(StockPriceUpdateEventV2 message) {
24      LOG.info("Received stock price update v2: {}", message);
25  }
```
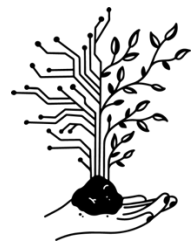
exeta

CodeBuzz 2025

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;
// }
```

exeta
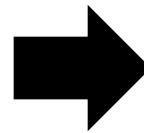
stockprice-monitor > StockPriceMonitor

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);
```

exeta

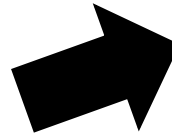CodeBuzz 2025

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);
```

exeta

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);
```

exeta

trading-platform > StockUpdateEventListener

13
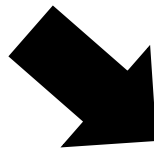
```java
@RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
public void handleStockPriceUpdate(StockPriceUpdateEventV2 message) {
    LOG.info("Received stock price update v2: {}", message);
}

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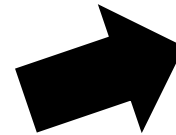
```java
// @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
// public void handleStockPriceUpdate(StockPriceUpdateEventV2 message) {
// LOG.info("Received stock price update v2: {}", message);
// }

@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");
}
```

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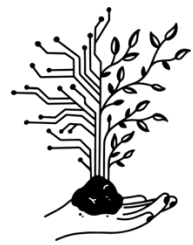
```
27  @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
28  public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message) {
29      LOG.info("Received stock price update v2: {}", message);
30      // Simulate processing logic that might fail
31      // For demonstration,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

exeta

CodeBuzz 2025

15

```
7   @SpringBootApplication
8   // @EnableRetry
9   public class TradingPlatformApplication {
10
11      public static void main(String[] args) {
12          SpringApplication.run(TradingPlatformApplication.class, args);
13      }
14
15  }
```

```
7   @SpringBootApplication
8   @EnableRetry
9   public class TradingPlatformApplication {
10
11      public static void main(String[] args) {
12          SpringApplication.run(TradingPlatformApplication.class, args);
13      }
14
15  }
```

exeta

16

```java
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   }
34
35   // @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
36   // @Retryable(maxAttempts = 3, backoff =
37   // @org.springframework.retry.annotation.Backoff(delay = 4000))
38   // public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message)
39   // {
40   // LOG.info("Received stock price update v2: {}", message);
41   // // Simulate processing logic that might fail
42   // // For demonstration, we throw an exception to trigger the retry mechanism
43   // throw new RuntimeException("Simulated error in StockUpdateEventListener");
44   // }
```
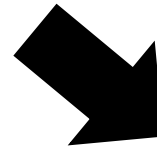
```java
27   // @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
28   // public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message) {
29   //     LOG.info("Received stock price update v2: {}", message);
30   //     // Simulate processing logic that might fail
31   //     // For demonstration, we throw an exception to trigger the retry mechanism
32   //     throw new AmqpRejectAndDontRequeueException("Simulated error in StockUpdateEventListener");
33   // }
34
35   @RabbitListener(queues = "tradingPlattformStockPriceUpdateV2")
36   @Retryable(maxAttempts = 3, backoff = @org.springframework.retry.annotation.Backoff(delay = 4000))
37   public void handleStockPriceUpdateWithRetry(StockPriceUpdateEventV2 message) {
38       LOG.info("Received stock price update v2: {}", message);
39       // Simulate processing logic that might fail
40       // For demonstration, we throw an exception to trigger the retry mechanism
41       throw new RuntimeException("Simulated error in StockUpdateEventListener");
42   }
```
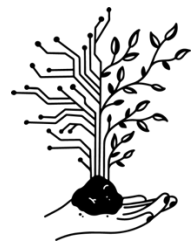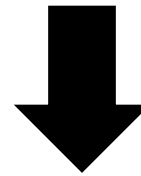
exeta

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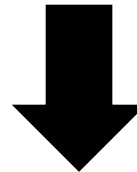
```
36  @Recover
37  public void recover(RuntimeException e, StockPriceUpdateEventV2 message) {
38      LOG.error("Failed to process stock price update after retries: {}", message,
39              e);
40      throw new AmqpRejectAndDontRequeueException("Failed to process stock price update", e);
41 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

trading-platform > „Terminal"

18

STRG+C

Rabbitmq (15672) > RabbitMQ Management UI

trading-platform > RabbitMQConfig

20

```
34    // @Bean
35    // public Queue stockPriceQueue() {
36    //     return new Queue("tradingPlattformStockPriceUpdateV2", false);
37    // }
```

```
56    // @Bean
57    // public Queue deadLetterQueue() {
58    // return new Queue("tradingPlattformStockPriceUpdateV2.dlq", false);
59    // }
60
61    // @Bean
62    // public Binding deadLetterBinding(Queue deadLetterQueue) {
63    // return BindingBuilder
64    // .bind(deadLetterQueue)
65    // .to(dlx())
66    // .with("");
67    // }
68
69    // @Bean
70    // DirectExchange dlx() {
71    // return new DirectExchange("tradingPlatformStockPriceUpdate.dlx");
72    // }
73
74    // @Bean
75    // public Queue stockPriceQueue() {
76    // return QueueBuilder
77    // .nonDurable("tradingPlattformStockPriceUpdateV2")
78    // .withArgument("x-dead-letter-exchange",
79    // "tradingPlatformStockPriceUpdate.dlx")
80    // .withArgument("x-dead-letter-routing-key", "")
81    // .build();
82    // }
```

```
56    @Bean
57    public Queue deadLetterQueue() {
58        return new Queue("tradingPlattformStockPriceUpdateV2.dlq", false);
59    }
60
61    @Bean
62    public Binding deadLetterBinding(Queue deadLetterQueue) {
63        return BindingBuilder
64                .bind(deadLetterQueue)
65                .to(dlx())
66                .with("");
67    }
68
69    @Bean
70    DirectExchange dlx() {
71        return new DirectExchange("tradingPlatformStockPriceUpdate.dlx");
72    }
73
74    @Bean
75    public Queue stockPriceQueue() {
76        return QueueBuilder
77                .nonDurable("tradingPlattformStockPriceUpdateV2")
78                .withArgument("x-dead-letter-exchange",
79                        "tradingPlatformStockPriceUpdate.dlx")
80                .withArgument("x-dead-letter-routing-key", "")
81                .build();
82    }
```
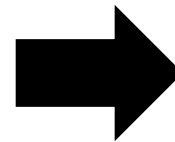
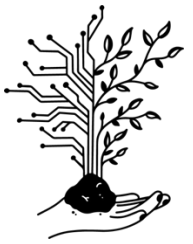# trading-platform > TradingPlatformApplication

21

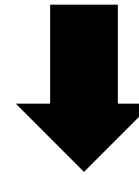TradingPlatformApplication
starten

```java
7    @SpringBootApplication
8    @EnableRetry
9    public class TradingPlatformApplication {
10
11       public static void main(String[] args) {
12           SpringApplication.run(TradingPlatformApplication.class, args);
13       }
14
15   }
```

exeta

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=430.419674122215
95, 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=992.594097128523
9, 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=849.898312468387
1, 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

| Overview | Connections | Channels | Exchanges | **Queues and Streams** | Admin |

Queues

▼ **All queues (3)**

Pagination

Page [1 ▾] of 1 - Filter: [_____] ☐ Regex **?**

Overview					Messages			Message rates			+/-
Virtual host	**Name**	**Type**	**Features**	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**	
/	**tradingPlattformStockPriceUpdate**	classic		▮ running	202	0	202	0.00/s	0.00/s	0.00/s	
/	**tradingPlattformStockPriceUpdateV2**	classic	DLX DLK	▮ running	0	0	0	0,00/s	0.00/s	0.00/s	
/	**tradingPlattformStockPriceUpdateV2.dlq**	classic		▮ running	9	0	9				

▸ **Add a new queue**

HTTP API Documentation Tutorials New releases Commercial edition Commercial support Discussions Discord Plugins GitHub