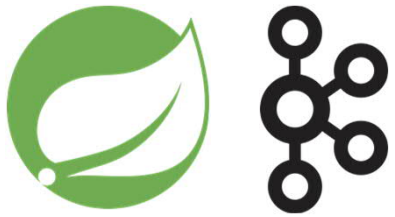
Spring & Kafka



Иван Головко
Альфа-Банк

В докладе рассмотрим

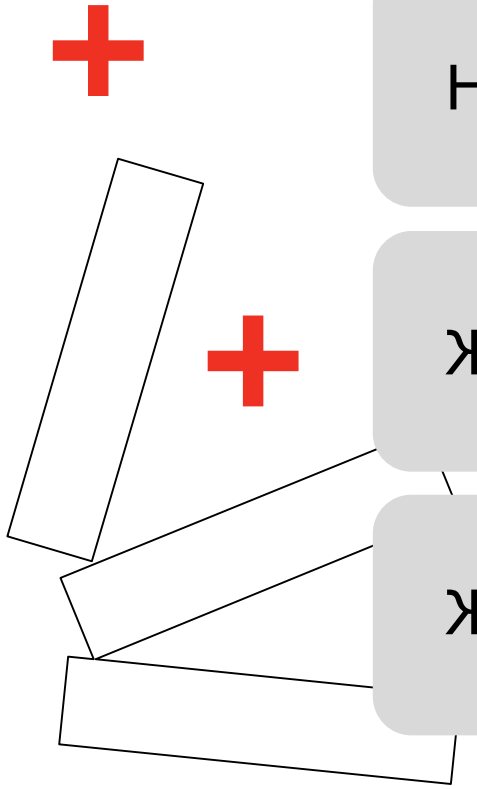
Способы конфигурации Consumer/Producer **на** Spring Kafka

- Без автоконфигурации
- С автоконфигурацией

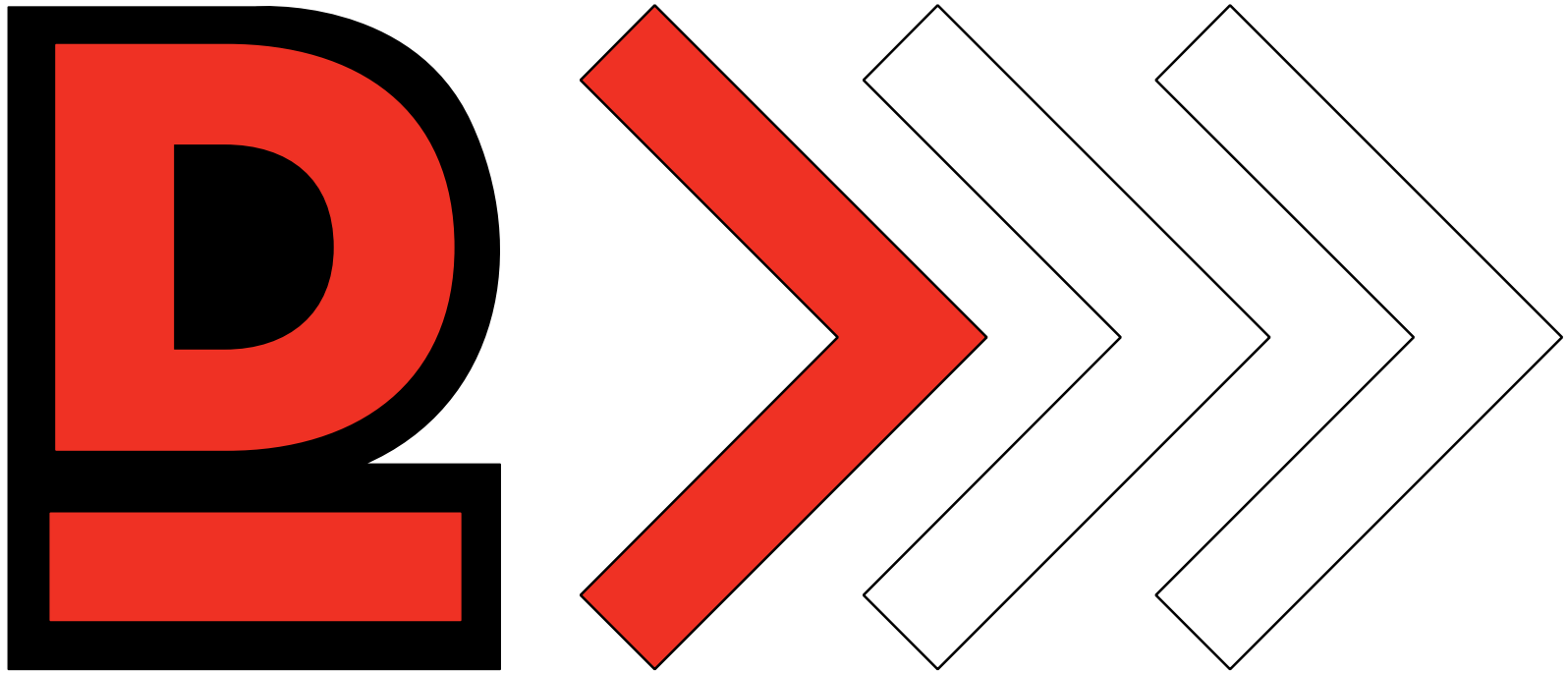
Ценные кейсы

- Подводные камни и грабли
- Полезные (и не очень) фичи

Кому будет полезно



- Новичкам в Kafka
- Желающим переехать с ванильного Kafka клиента на Spring
- Желающим доразобраться



Бизнес-кейс

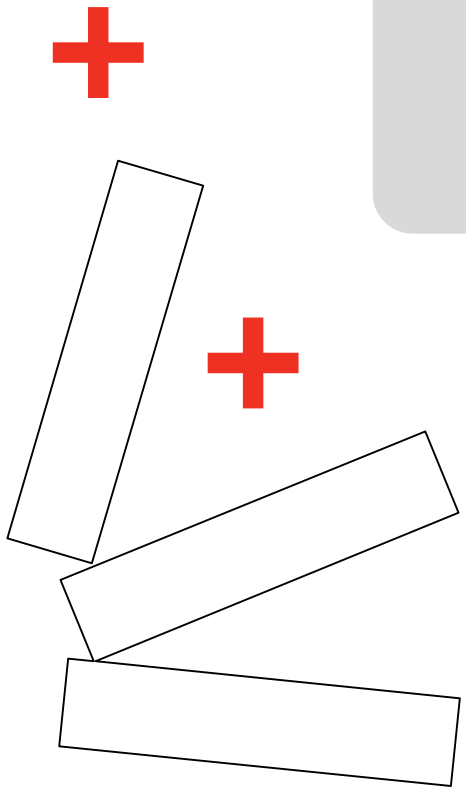
Бизнес-кейс

1. Payment-processor

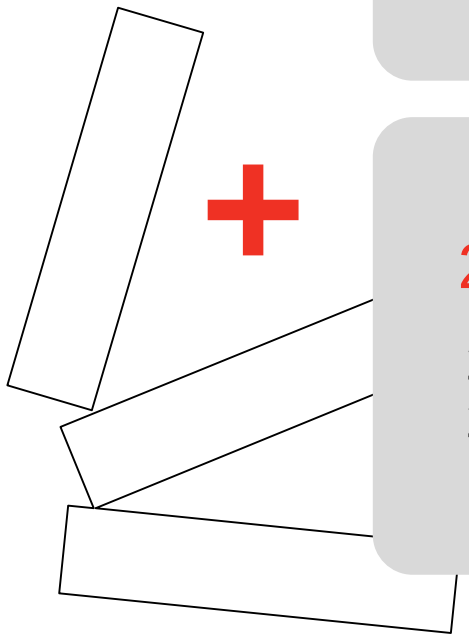
1.1. Обслуживает переводы с карты на карту

1.2. Создает платеж и генерирует OTP код для подтверждения операции

1.3. Кладет OTP в kafka топик



Бизнес-кейс



1. Payment-processor

1.1. Обслуживает переводы с карты на карту

1.2. Создает платеж и генерирует OTP код для подтверждения операции

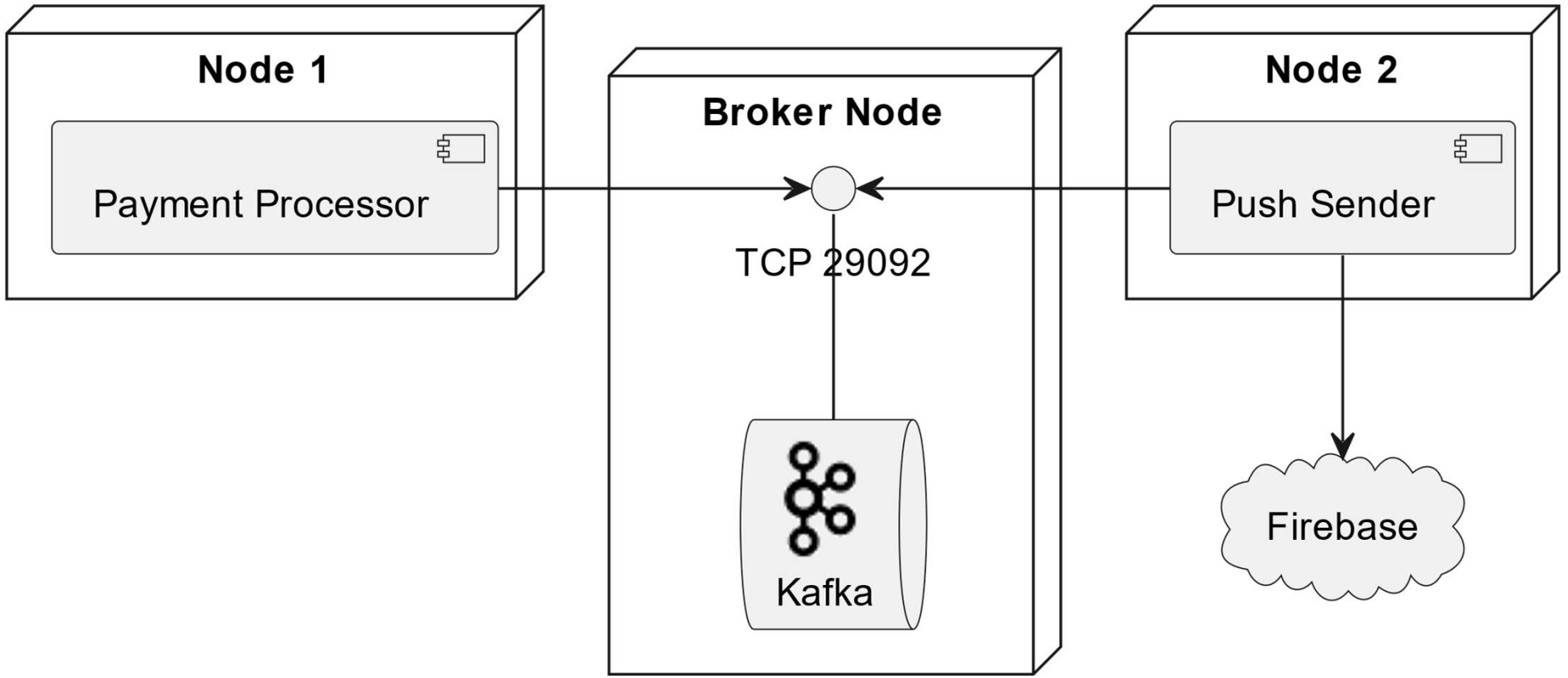
1.3. Кладет OTP в kafka топик

2. Push-sender

2.1. Читает топик кафки

2.2. Отправляет push сообщение с OTP кодом клиенту

Схема взаимодействия



Payment Processor: Application

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter'
    implementation 'org.springframework.kafka:spring-kafka'

    compileOnly "org.projectlombok:lombok"
    annotationProcessor "org.projectlombok:lombok"
}
```

Payment Processor: Application

```
@Scheduled(fixedDelay = 5000)
void doYourJob() {
    for (int i = 0; i < 10; ++i) {
        paymentService.acceptPayment();
    }
    log.info("Sent new batch");
}
```

Payment Processor: Application

1.Раз в 5 секунд запускаем джобу

```
@Scheduled(fixedDelay = 5000)
void doYourJob() {
    for (int i = 0; i < 10; ++i) {
        paymentService.acceptPayment();
    }
    log.info("Sent new batch");
}
```

Payment Processor: Application

```
@Scheduled(fixedDelay = 5000)
void doYourJob() {
    for (int i = 0; i < 10; ++i) {
        paymentService.acceptPayment();
    }
    log.info("Sent new batch");
}
```

1.Раз в 5 секунд запускаем джобу

2.Инициируем 10 платежей

Payment Processor: Application

```
@Scheduled(fixedDelay = 5000)
void doYourJob() {
    for (int i = 0; i < 10; ++i) {
        paymentService.acceptPayment();
    }
    log.info("Sent new batch");
}
```

1.Раз в 5 секунд запускаем джобу

2.Инициируем 10 платежей

3.Логируем факт отправки

Payment Processor: **Конфигурация**

```java
@EnableKafka
@Configuration
public class KafkaTemplateConfiguration {

    public static final String MY_TOPIC = "my-topic";

    @Bean
    @SneakyThrows
    public ProducerFactory<String, OtpDto> producerFactory() {
        Map<String, Object> props = new HashMap<>();
        props.put(
           ProducerConfig.CLIENT_ID_CONFIG,
           InetAddress.getLocalHost().getHostName()
        );
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:29092");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);

        return new DefaultKafkaProducerFactory<>(props);
    }

    @Bean
    public KafkaTemplate<String, OtpDto> kafkaTemplate(ProducerFactory<String,
            OtpDto> producerFactory) {
        var kafkaTemplate = new KafkaTemplate<>(producerFactory);
        kafkaTemplate.setDefaultTopic(MY_TOPIC);
        return kafkaTemplate;
    }
}
```

Payment Processor: **Конфигурация**

```java
@EnableKafka
@Configuration
public class KafkaTemplateConfiguration {

    public static final String MY_TOPIC = "my-topic";

    @Bean
    @SneakyThrows
    public ProducerFactory<String, OtpDto> producerFactory() {
        Map<String, Object> props = new HashMap<>();
        props.put(
           ProducerConfig.CLIENT_ID_CONFIG,
           InetAddress.getLocalHost().getHostName()
        );
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:29092");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);

        return new DefaultKafkaProducerFactory<>(props);
    }

    @Bean
    public KafkaTemplate<String, OtpDto> kafkaTemplate(ProducerFactory<String,
            OtpDto> producerFactory) {
        var kafkaTemplate = new KafkaTemplate<>(producerFactory);
        kafkaTemplate.setDefaultTopic(MY_TOPIC);
        return kafkaTemplate;
    }
}
```

Конфигурим
ProducerFactory:

Payment Processor: **Конфигурация**

```java
@EnableKafka
@Configuration
public class KafkaTemplateConfiguration {

    public static final String MY_TOPIC = "my-topic";

    @Bean
    @SneakyThrows
    public ProducerFactory<String, OtpDto> producerFactory() {
        Map<String, Object> props = new HashMap<>();
        props.put(
           ProducerConfig.CLIENT_ID_CONFIG,
           InetAddress.getLocalHost().getHostName()
        );
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:29092");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);

        return new DefaultKafkaProducerFactory<>(props);
    }

    @Bean
    public KafkaTemplate<String, OtpDto> kafkaTemplate(ProducerFactory<String,
            OtpDto> producerFactory) {
        var kafkaTemplate = new KafkaTemplate<>(producerFactory);
        kafkaTemplate.setDefaultTopic(MY_TOPIC);
        return kafkaTemplate;
    }
}
```

Конфигурим ProducerFactory:

1. Вместо Properties используем Map

Payment Processor: **Конфигурация**

```java
@EnableKafka
@Configuration
public class KafkaTemplateConfiguration {

    public static final String MY_TOPIC = "my-topic";

    @Bean
    @SneakyThrows
    public ProducerFactory<String, OtpDto> producerFactory() {
        Map<String, Object> props = new HashMap<>();
        props.put(
           ProducerConfig.CLIENT_ID_CONFIG,
           InetAddress.getLocalHost().getHostName()
        );
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:29092");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);

        return new DefaultKafkaProducerFactory<>(props);
    }

    @Bean
    public KafkaTemplate<String, OtpDto> kafkaTemplate(ProducerFactory<String,
            OtpDto> producerFactory) {
        var kafkaTemplate = new KafkaTemplate<>(producerFactory);
        kafkaTemplate.setDefaultTopic(MY_TOPIC);
        return kafkaTemplate;
    }
}
```

Конфигурим ProducerFactory:

1. Вместо Properties используем Map
2. JsonSerializer

Payment Processor: **Конфигурация**

```java
@EnableKafka
@Configuration
public class KafkaTemplateConfiguration {

    public static final String MY_TOPIC = "my-topic";

    @Bean
    @SneakyThrows
    public ProducerFactory<String, OtpDto> producerFactory() {
        Map<String, Object> props = new HashMap<>();
        props.put(
           ProducerConfig.CLIENT_ID_CONFIG,
           InetAddress.getLocalHost().getHostName()
        );
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:29092");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);

        return new DefaultKafkaProducerFactory<>(props);
    }

    @Bean
    public KafkaTemplate<String, OtpDto> kafkaTemplate(ProducerFactory<String,
            OtpDto> producerFactory) {
        var kafkaTemplate = new KafkaTemplate<>(producerFactory);
        kafkaTemplate.setDefaultTopic(MY_TOPIC);
        return kafkaTemplate;
    }
}
```

Конфигурим KafkaTemplate:

Payment Processor: **Конфигурация**

```java
@EnableKafka
@Configuration
public class KafkaTemplateConfiguration {

    public static final String MY_TOPIC = "my-topic";

    @Bean
    @SneakyThrows
    public ProducerFactory<String, OtpDto> producerFactory() {
        Map<String, Object> props = new HashMap<>();
        props.put(
           ProducerConfig.CLIENT_ID_CONFIG,
           InetAddress.getLocalHost().getHostName()
        );
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:29092");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);

        return new DefaultKafkaProducerFactory<>(props);
    }

    @Bean
    public KafkaTemplate<String, OtpDto> kafkaTemplate(ProducerFactory<String,
            OtpDto> producerFactory) {
        var kafkaTemplate = new KafkaTemplate<>(producerFactory);
        kafkaTemplate.setDefaultTopic(MY_TOPIC);
        return kafkaTemplate;
    }
}
```

Конфигурим KafkaTemplate:

1. Используем ProducerFactory

Payment Processor: **Конфигурация**

```java
@EnableKafka
@Configuration
public class KafkaTemplateConfiguration {

    public static final String MY_TOPIC = "my-topic";

    @Bean
    @SneakyThrows
    public ProducerFactory<String, OtpDto> producerFactory() {
        Map<String, Object> props = new HashMap<>();
        props.put(
           ProducerConfig.CLIENT_ID_CONFIG,
           InetAddress.getLocalHost().getHostName()
        );
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:29092");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);

        return new DefaultKafkaProducerFactory<>(props);
    }

    @Bean
    public KafkaTemplate<String, OtpDto> kafkaTemplate(ProducerFactory<String,
            OtpDto> producerFactory) {
        var kafkaTemplate = new KafkaTemplate<>(producerFactory);
        kafkaTemplate.setDefaultTopic(MY_TOPIC);
        return kafkaTemplate;
    }
}
```

Конфигурим KafkaTemplate:

1. Используем ProducerFactory
2. Устанавливаем default topic

Payment Processor: **Как отправлять**?

Payment Processor: **Как отправлять**?

В KafkaTemplate **много перегруженных методов**

Payment Processor: **Как отправлять**?

В KafkaTemplate **много перегруженных методов**

```
private void sendPushAsync(OtpDto otpDto) {
    kafkaTemplate.send(MY_TOPIC, otpDto);
}
```

Отправка в определенный топик

Payment Processor: **Как отправлять**?

В KafkaTemplate **много перегруженных методов**

```
private void sendPushAsync(OtpDto otpDto) {
    kafkaTemplate.send(MY_TOPIC, otpDto);
}
```

Отправка в определенный топик

```
private void sendPushAsync(OtpDto otpDto) {
    var producerRecord = new ProducerRecord(
        MY_TOPIC,
        otpDto
    );
    kafkaTemplate.send(producerRecord);
}
```

Использование ванильного ProducerRecord. **Особенно полезно при переезде на** spring kafka.

Payment Processor: **Как отправлять**?

В KafkaTemplate **много перегруженных методов**

```
private void sendPushAsync(OtpDto otpDto) {
    kafkaTemplate.send(MY_TOPIC, otpDto);
}
```

Отправка в определенный топик

```
private void sendPushAsync(OtpDto otpDto) {
    var producerRecord = new ProducerRecord(
        MY_TOPIC,
        otpDto
    );
    kafkaTemplate.send(producerRecord);
}
```

Использование ванильного ProducerRecord.
Особенно полезно при переезде на spring kafka.

```
private void sendPushAsync(OtpDto otpDto) {
    var message = MessageBuilder.withPayload(otpDto)
            .setHeader(KafkaHeaders.TOPIC, MY_TOPIC)
            .setHeader(KafkaHeaders.PARTITION, 0)
            .setHeader(KafkaHeaders.KEY, "my-key")
            .build();

    kafkaTemplate.send(message);
}
```

Использование Message
из пакета org.springframework.messaging

Payment Processor: **Отправка сообщения**

Payment Processor: **Отправка сообщения**

```
private void sendPushAsync(OtpDto otpDto) {
    kafkaTemlate.sendDefault(otpDto);
}
```

Payment Processor: **Отправка сообщения**

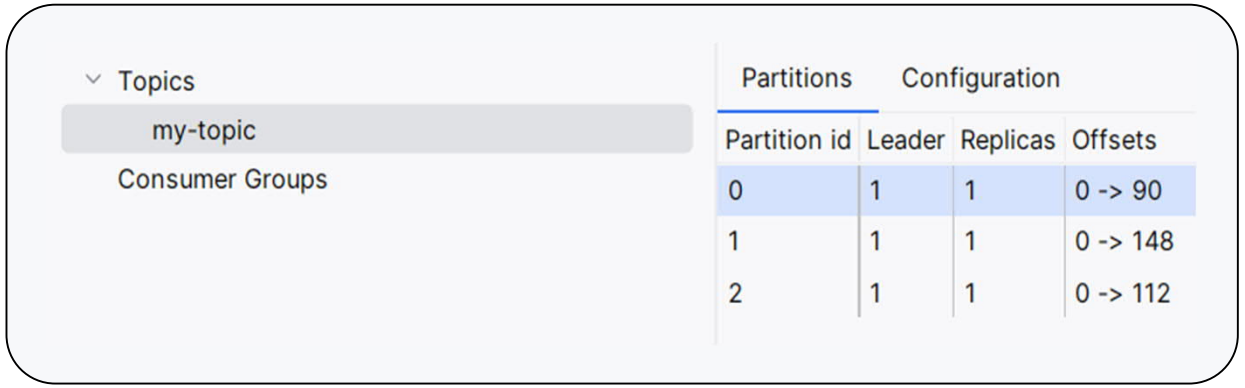
```
private void sendPushAsync(OtpDto otpDto) {
    kafkaTemlate.sendDefault(otpDto);
}


@Builder
public record OtpDto(
        String sender,
        String userId,
        String code,
        LocalDateTime expireTime
) {
}
```

Payment Processor: **Отправка сообщения**

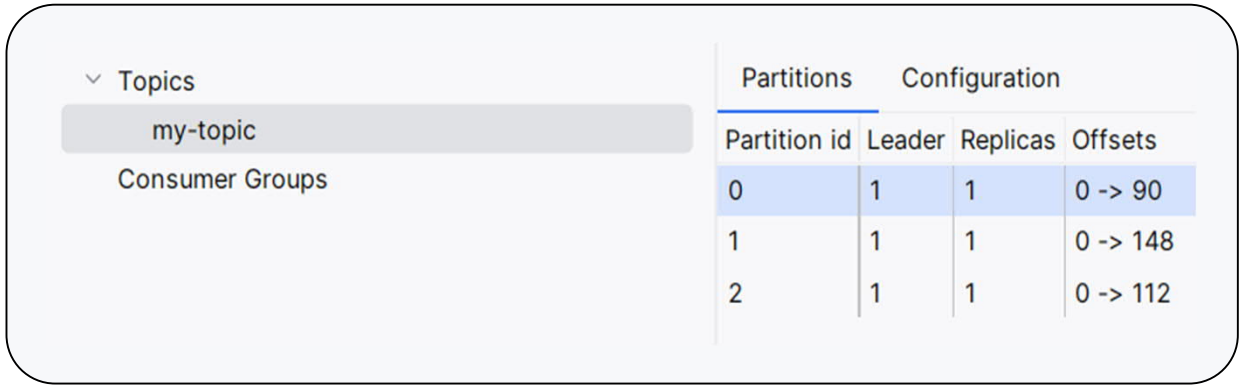
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.1)

2024-07-16T11:07:52.194+03:00  INFO 28772 --- [           main] r.a.j.kafka.template.KafkaTemplateApp    : Starting KafkaTemplateApp using Java 17.0.9 with PID 28772
2024-07-16T11:07:52.196+03:00  INFO 28772 --- [           main] r.a.j.kafka.template.KafkaTemplateApp    : No active profile set, falling back to 1 default profile:
2024-07-16T11:07:52.567+03:00  INFO 28772 --- [           main] r.a.j.kafka.template.KafkaTemplateApp    : Started KafkaTemplateApp in 0.558 seconds (process running
2024-07-16T11:07:52.808+03:00  INFO 28772 --- [   scheduling-1] r.a.j.kafka.template.KafkaTemplateApp    : Sent new batch
2024-07-16T11:07:57.826+03:00  INFO 28772 --- [   scheduling-1] r.a.j.kafka.template.KafkaTemplateApp    : Sent new batch
2024-07-16T11:08:02.845+03:00  INFO 28772 --- [   scheduling-1] r.a.j.kafka.template.KafkaTemplateApp    : Sent new batch
2024-07-16T11:08:07.849+03:00  INFO 28772 --- [   scheduling-1] r.a.j.kafka.template.KafkaTemplateApp    : Sent new batch
2024-07-16T11:08:12.853+03:00  INFO 28772 --- [   scheduling-1] r.a.j.kafka.template.KafkaTemplateApp    : Sent new batch
2024-07-16T11:08:17.872+03:00  INFO 28772 --- [   scheduling-1] r.a.j.kafka.template.KafkaTemplateApp    : Sent new batch
2024-07-16T11:08:22.883+03:00  INFO 28772 --- [   scheduling-1] r.a.j.kafka.template.KafkaTemplateApp    : Sent new batch
```

Payment Processor: Big Data Tools Plugin

Payment Processor: Big Data Tools Plugin

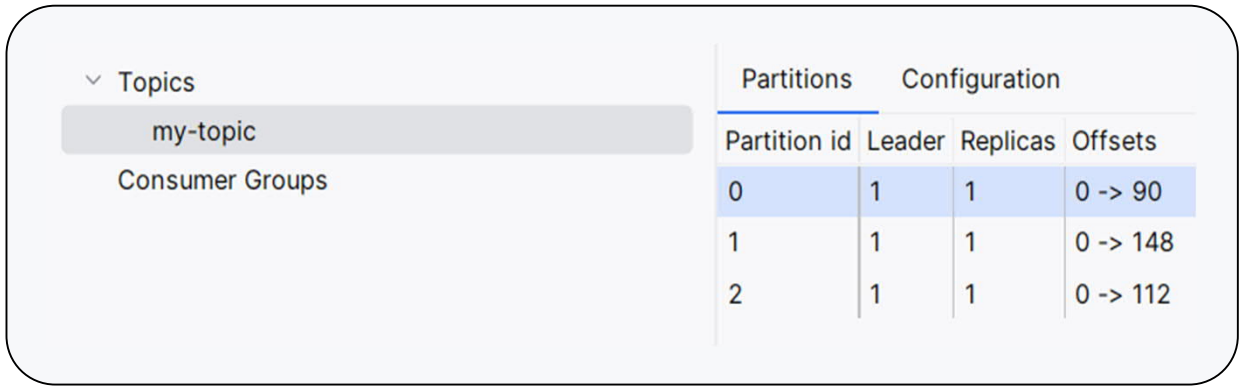


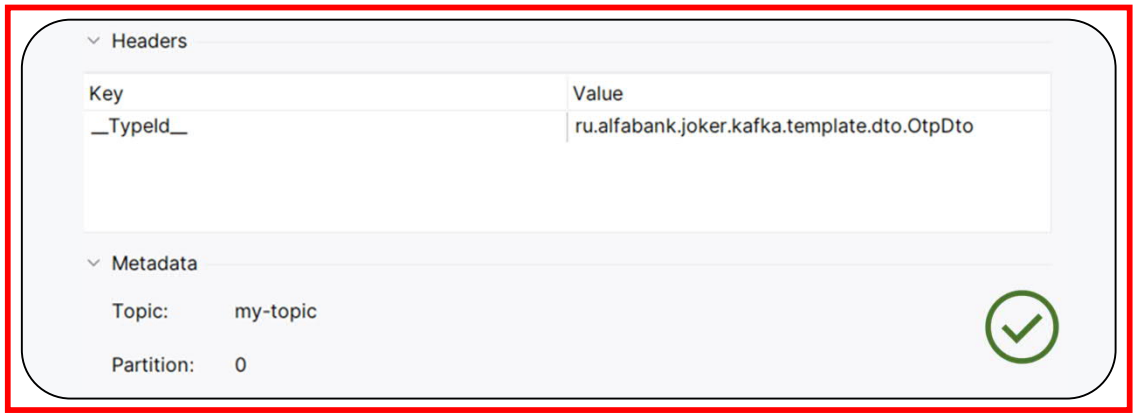
Payment Processor: Big Data Tools Plugin



Timestamp	Key	Value	Partition	Offset
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 343898800]}	0	80
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 343898800]}	0	81
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 344896900]}	0	82
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 345903500]}	0	83
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 345903500]}	0	84
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 345903500]}	0	85
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 346830000]}	0	86
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 347345900]}	0	87
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 347345900]}	0	88
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 348453100]}	0	89
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 455033700]}	1	83
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 688536500]}	1	84
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 689585300]}	1	85
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 689783000]}	1	86
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 689783000]}	1	87
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 689783000]}	1	88
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 689783000]}	1	89

Payment Processor: Big Data Tools Plugin





Timestamp	Key	Value	Partition	Offset
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 343898800]}	0	80
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 343898800]}	0	81
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 344896900]}	0	82
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 345903500]}	0	83
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 345903500]}	0	84
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 345903500]}	0	85
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 346830000]}	0	86
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 347345900]}	0	87
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 347345900]}	0	88
2024-07-16 11:07:07		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 7, 348453100]}	0	89
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 455033700]}	1	83
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 688536500]}	1	84
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 689585300]}	1	85
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 689783000]}	1	86
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 689783000]}	1	87
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 689783000]}	1	88
2024-07-16 11:07:35		{ "sender": "payment-processor", "userId": "me", "code": "my-secret-code", "expireTime": [2024, 7, 16, 11, 8, 35, 689783000]}	1	89

Payment Processor: **Критичные данные**! ⚠

Внимание, подводные камни!

Payment Processor: **Критичные данные**! ⚠

Внимание, подводные камни!

– **плохо**

– **хорошо**

Payment Processor: **Критичные данные**! ⚠

Внимание, подводные камни!

⊗ – **плохо**

```
void sendPushAsync(OtpDto otpDto) {
    kafkaTemlate.sendDefault(otpDto);
}
```

Payment Processor: **Критичные данные**! ⚠

Внимание, подводные камни!

– плохо

```
void sendPushAsync(OtpDto otpDto) {
    kafkaTemlate.sendDefault(otpDto);
}
```

– хорошо

```
@SneakyThrows
void sendPushSync(OtpDto otpDto) {
    kafkaTemplate.sendDefault(otpDto).get();
}
```

Payment Processor: **Критичные данные**! ⚠

Внимание, подводные камни!

⊗ – **плохо**

```
props.put(ProducerConfig.CLIENT_ID_CONFIG, InetAddress.getLocalHost().getHostName());
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:29092");
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);
```

Payment Processor: **Критичные данные**! ⚠

Внимание, подводные камни!

✓ – **хорошо**

```
props.put(ProducerConfig.CLIENT_ID_CONFIG, InetAddress.getLocalHost().getHostName());
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:29092");
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);
props.put(ProducerConfig.ACKS_CONFIG, "all");
```

Payment Processor: **Критичные данные**!

Для критичных данных:

1. Вызывайте get() на CompletableFuture

Payment Processor: **Критичные данные**!

Для критичных данных:

1. Вызывайте get() на CompletableFuture

2. Используйте настройку acks=all

Payment Processor: **Автоконфигурация**

Payment Processor: **Автоконфигурация**

Нам не нужен собственный KafkaTemplateConfiguration!

```yaml
spring:
  kafka:
    bootstrap-servers: localhost:29092
    producer:
      value-serializer: org.springframework.kafka.support.serializer.JsonSerializer
    template:
      default-topic: my-topic
```

Payment Processor: **Автоконфигурация**

Если не хочется хардкодить класс в application.yml

```
@Configuration
public class KafkaTemplateConfiguration {

    @Bean
    DefaultKafkaProducerFactoryCustomizer serializerCustomizer() {
        JsonSerializer jsonSerializer = new JsonSerializer();
        return producerFactory -> producerFactory.setValueSerializer(jsonSerializer);
    }
}
```

Payment Processor: **Коварный** ObjectMapper

Проблема! **Разный** ObjectMapper **в контексте и в** JsonSerializer!

```
@Configuration
public class KafkaTemplateConfiguration {

    @Bean
    DefaultKafkaProducerFactoryCustomizer serializerCustomizer() {
        JsonSerializer jsonSerializer = new JsonSerializer();
        return producerFactory -> producerFactory.setValueSerializer(jsonSerializer);
    }
}
```

Payment Processor: **Коварный** ObjectMapper

Проблема! **Разный** ObjectMapper **в контексте и в** JsonSerializer!

```
@Bean
Jackson2ObjectMapperBuilderCustomizer objectMapperBuilderCustomizer() {
    return builder -> builder.featuresToDisable(
                SerializationFeature.WRITE_DATES_AS_TIMESTAMPS
            );
}
```

Payment Processor: **Коварный** ObjectMapper

Проблема! **Разный** ObjectMapper **в контексте и в** JsonSerializer!

```
@Bean
Jackson2ObjectMapperBuilderCustomizer objectMapperBuilderCustomizer() {
    return builder -> builder.featuresToDisable(
                SerializationFeature.WRITE_DATES_AS_TIMESTAMPS
            );
}
```

Во всем приложении

```
{
 "sender": "payment-processor",
 "userId": "me",
 "code": "my-secret-code",
 "expireTime": "2024-09-23T13:02:24.6840453"
}
```

Payment Processor: **Коварный** ObjectMapper

Проблема! **Разный** ObjectMapper **в контексте и в** JsonSerializer!

```
@Bean
Jackson2ObjectMapperBuilderCustomizer objectMapperBuilderCustomizer() {
    return builder -> builder.featuresToDisable(
                SerializationFeature.WRITE_DATES_AS_TIMESTAMPS
            );
}
```

Во всем приложении

```
{
 "sender": "payment-processor",
 "userId": "me",
 "code": "my-secret-code",
 "expireTime": "2024-09-23T13:02:24.6840453"
}
```

В Spring Kafka

```
{
 "sender": "payment-processor",
 "userId": "me",
 "code": "my-secret-code",
 "expireTime": [2024, 9, 23, 13, 2, 24,
                684045300]
}
```

Payment Processor: **Коварный** ObjectMapper

Решение

```
@Configuration
public class KafkaTemplateConfiguration {

    @Bean
    DefaultKafkaProducerFactoryCustomizer serializerCustomizer(
                ObjectMapper objectMapper) {
        JsonSerializer jsonSerializer = new JsonSerializer(objectMapper);
        return producerFactory ->producerFactory.setValueSerializer(jsonSerializer);
    }
}
```

Push Sender

Push Sender: **Конфигурация**

```
@Bean
@SneakyThrows
public ConsumerFactory<String, OtpDto> consumerFactory() {
    Map<String, Object> props = new HashMap<>();

    props.put(ConsumerConfig.CLIENT_ID_CONFIG, InetAddress.getLocalHost().getHostName());
    props.put(ConsumerConfig.GROUP_ID_CONFIG, "my-group");
    props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "latest");
    props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:29092");
    props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);
    props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, JsonDeserializer.class);

    return new DefaultKafkaConsumerFactory<>(props);
}
```

Push Sender: **Конфигурация**

```java
public interface MessageListener<K, V> { ❶

    void onMessage(ConsumerRecord<K, V> data);

}

public interface AcknowledgingMessageListener<K, V> { ❷

    void onMessage(ConsumerRecord<K, V> data, Acknowledgment acknowledgment);

}

public interface ConsumerAwareMessageListener<K, V> extends MessageListener<K, V> { ❸

    void onMessage(ConsumerRecord<K, V> data, Consumer<?, ?> consumer);

}

public interface AcknowledgingConsumerAwareMessageListener<K, V> extends MessageListener<K, V> { ❹

    void onMessage(ConsumerRecord<K, V> data, Acknowledgment acknowledgment, Consumer<?, ?> consumer);

}
```

Push Sender: **Конфигурация**

```
public interface MessageListener<K, V> { ❶

    void onMessage(ConsumerRecord<K, V> data);

}

public interface AcknowledgingMessageListener<K, V> { ❷

    void onMessage(ConsumerRecord<K, V> data, Acknowledgment acknowledgment);

}

public interface ConsumerAwareMessageListener<K, V> extends MessageListener<K, V> { ❸

    void onMessage(ConsumerRecord<K, V> data, Consumer<?, ?> consumer);

}

public interface AcknowledgingConsumerAwareMessageListener<K, V> extends MessageListener<K, V> { ❹

    void onMessage(ConsumerRecord<K, V> data, Acknowledgment acknowledgment, Consumer<?, ?> consumer);

}
```

Push Sender: **Конфигурация**

```java
public interface MessageListener<K, V> { ❶
    void onMessage(ConsumerRecord<K, V> data);
}
public interface AcknowledgingMessageListener<K, V> { ❷
    void onMessage(ConsumerRecord<K, V> data, Acknowledgment acknowledgment);
}
public interface ConsumerAwareMessageListener<K, V> extends MessageListener<K, V> { ❸
    void onMessage(ConsumerRecord<K, V> data, Consumer<?, ?> consumer);
}
public interface AcknowledgingConsumerAwareMessageListener<K, V> extends MessageListener<K, V> { ❹
    void onMessage(ConsumerRecord<K, V> data, Acknowledgment acknowledgment, Consumer<?, ?> consumer);
}
```

Push Sender: **Конфигурация**

```
@Bean
MessageListener<String, OtpDto> messageListener() {
    return (record) -> {
        log.info(
           "Received from {}-{}-{}",
           record.topic(),
           record.partition(),
           record.offset()
        );

        this.pushService.sendPush(record.value());

        log.info(
            "Processed from {}-{}-{}",
            record.topic(),
            record.partition(),
            record.offset()
        );
    };
}
```

Push Sender: **Конфигурация**

```
@Bean
KafkaMessageListenerContainer<String, OtpDto>
kafkaListenerContainer(
        ConsumerFactory<String, OtpDto> factory,
        MessageListener<String, OtpDto> listener) {

    ContainerProperties containerProperties =
         new ContainerProperties(MY_TOPIC);

    containerProperties.setMessageListener(listener);

    return new KafkaMessageListenerContainer<>(
        factory,
        containerProperties
    );
}
```

KafkaMessageListenerContainer:

1. Управляет жизненным циклом Consumer

Push Sender: **Конфигурация**

```
@Bean
KafkaMessageListenerContainer<String, OtpDto>
kafkaListenerContainer(
        ConsumerFactory<String, OtpDto> factory,
        MessageListener<String, OtpDto> listener) {

    ContainerProperties containerProperties =
         new ContainerProperties(MY_TOPIC);

    containerProperties.setMessageListener(listener);

    return new KafkaMessageListenerContainer<>(
        factory,
        containerProperties
    );
}
```

KafkaMessageListenerContainer:

1. Управляет жизненным циклом Consumer

2. Однопоточный

Push Sender: **Конфигурация**

```
@Bean
KafkaMessageListenerContainer<String, OtpDto>
kafkaListenerContainer(
        ConsumerFactory<String, OtpDto> factory,
        MessageListener<String, OtpDto> listener) {

    ContainerProperties containerProperties =
         new ContainerProperties(MY_TOPIC);

    containerProperties.setMessageListener(listener);

    return new KafkaMessageListenerContainer<>(
        factory,
        containerProperties
    );
}
```

KafkaMessageListenerContainer:

1. Управляет жизненным циклом Consumer
2. Однопоточный
3. Гибко настраивается под наши нужды

Push Sender: **Чтение сообщения**

Push Sender: **Чтение сообщения**

Пререквизит: **в коде двух сервисов** OtpDto **находится в разных пакетах**

Push Sender: **Чтение сообщения**

Всё сломалось!!!

```
java.lang.IllegalStateException Create breakpoint : This error handler cannot process 'SerializationException's directly; please consider configuring an 'ErrorHandlingDeserializer' in the value and/or key deserializer
    at org.springframework.kafka.listener.DefaultErrorHandler.handleOtherException(DefaultErrorHandler.java:192) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.handleConsumerException(KafkaMessageListenerContainer.java:1925) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.run(KafkaMessageListenerContainer.java:1348) ~[spring-kafka-3.2.1.jar:3.2.1] <2 internal lines>
Caused by: org.apache.kafka.common.errors.RecordDeserializationException Create breakpoint : Error deserializing key/value for partition my-topic-0 at offset 1905. If needed, please seek past the record to continue consumption.
    at org.apache.kafka.clients.consumer.internals.CompletedFetch.parseRecord(CompletedFetch.java:331) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.CompletedFetch.fetchRecords(CompletedFetch.java:283) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.FetchCollector.fetchRecords(FetchCollector.java:168) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.FetchCollector.collectFetch(FetchCollector.java:134) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.Fetcher.collectFetch(Fetcher.java:145) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.LegacyKafkaConsumer.pollForFetches(LegacyKafkaConsumer.java:666) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.LegacyKafkaConsumer.poll(LegacyKafkaConsumer.java:617) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.LegacyKafkaConsumer.poll(LegacyKafkaConsumer.java:590) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.KafkaConsumer.poll(KafkaConsumer.java:874) ~[kafka-clients-3.7.0.jar:na]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.pollConsumer(KafkaMessageListenerContainer.java:1625) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.doPoll(KafkaMessageListenerContainer.java:1600) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.pollAndInvoke(KafkaMessageListenerContainer.java:1405) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.run(KafkaMessageListenerContainer.java:1296) ~[spring-kafka-3.2.1.jar:3.2.1]
    ... 2 common frames omitted
Caused by: java.lang.IllegalArgumentException Create breakpoint : The class 'ru.alfabank.joker.kafka.boot.template.dto.OtpDto' is not in the trusted packages: [java.util, java.lang]. If you believe this class is safe to deseria
    at org.springframework.kafka.support.mapping.DefaultJackson2JavaTypeMapper.getClassIdType(DefaultJackson2JavaTypeMapper.java:124) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.support.mapping.DefaultJackson2JavaTypeMapper.toJavaType(DefaultJackson2JavaTypeMapper.java:98) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.support.serializer.JsonDeserializer.deserialize(JsonDeserializer.java:571) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.apache.kafka.common.serialization.Deserializer.deserialize(Deserializer.java:73) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.CompletedFetch.parseRecord(CompletedFetch.java:321) ~[kafka-clients-3.7.0.jar:na]
    ... 14 common frames omitted
```

Push Sender: **Чтение сообщения**

Всё сломалось!!!

```
java.lang.IllegalStateException Create breakpoint : This error handler cannot process 'SerializationException's directly; please consider configuring an 'ErrorHandlingDeserializer' in the value and/or key deserializer
    at org.springframework.kafka.listener.DefaultErrorHandler.handleOtherException(DefaultErrorHandler.java:192) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.handleConsumerException(KafkaMessageListenerContainer.java:1925) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.run(KafkaMessageListenerContainer.java:1348) ~[spring-kafka-3.2.1.jar:3.2.1] <2 internal lines>
Caused by: org.apache.kafka.common.errors.RecordDeserializationException Create breakpoint : Error deserializing key/value for partition my-topic-0 at offset 1905. If needed, please seek past the record to continue consumption.
    at org.apache.kafka.clients.consumer.internals.CompletedFetch.parseRecord(CompletedFetch.java:331) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.CompletedFetch.fetchRecords(CompletedFetch.java:283) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.FetchCollector.fetchRecords(FetchCollector.java:168) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.FetchCollector.collectFetch(FetchCollector.java:134) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.Fetcher.collectFetch(Fetcher.java:145) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.LegacyKafkaConsumer.pollForFetches(LegacyKafkaConsumer.java:666) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.LegacyKafkaConsumer.poll(LegacyKafkaConsumer.java:617) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.LegacyKafkaConsumer.poll(LegacyKafkaConsumer.java:590) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.KafkaConsumer.poll(KafkaConsumer.java:874) ~[kafka-clients-3.7.0.jar:na]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.pollConsumer(KafkaMessageListenerContainer.java:1625) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.doPoll(KafkaMessageListenerContainer.java:1600) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.pollAndInvoke(KafkaMessageListenerContainer.java:1405) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.run(KafkaMessageListenerContainer.java:1296) ~[spring-kafka-3.2.1.jar:3.2.1]
    ... 2 common frames omitted
Caused by: java.lang.IllegalArgumentException Create breakpoint : The class 'ru.alfabank.joker.kafka.boot.template.dto.OtpDto' is not in the trusted packages: [java.util, java.lang]. If you believe this class is safe to deseria
    at org.springframework.kafka.support.mapping.DefaultJackson2JavaTypeMapper.getClassIdType(DefaultJackson2JavaTypeMapper.java:124) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.support.mapping.DefaultJackson2JavaTypeMapper.toJavaType(DefaultJackson2JavaTypeMapper.java:98) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.support.serializer.JsonDeserializer.deserialize(JsonDeserializer.java:571) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.apache.kafka.common.serialization.Deserializer.deserialize(Deserializer.java:73) ~[kafka-clients-3.7.0.jar:na]
    at org.apache.kafka.clients.consumer.internals.CompletedFetch.parseRecord(CompletedFetch.java:321) ~[kafka-clients-3.7.0.jar:na]
    ... 14 common frames omitted

  If you believe this class is safe to deserialize, please provide its name. If the serialization is only done by a trusted source, you can also enable trust all (*).
```

Push Sender: **Чтение сообщения**

```
props.put(
    JsonDeserializer.TRUSTED_PACKAGES,
    "ru.alfabank.joker.kafka.boot.template.dto"
);
```

```
props.put(JsonDeserializer.TRUSTED_PACKAGES, "*");
```

Push Sender: **Чтение сообщения**

```
props.put(
    JsonDeserializer.TRUSTED_PACKAGES,
    "ru.alfabank.joker.kafka.boot.template.dto"
);
```

```
props.put(JsonDeserializer.TRUSTED_PACKAGES, "*");
```

Push Sender: **Чтение сообщения**

Будет работать только для Shared Libraries

```
props.put(
    JsonDeserializer.TRUSTED_PACKAGES,
    "ru.alfabank.joker.kafka.boot.template.dto"
);
```

```
props.put(JsonDeserializer.TRUSTED_PACKAGES, "*");
```

Push Sender: **Чтение сообщения**

Маппинг типов

```
props.put(
    JsonDeserializer.TYPE_MAPPINGS,
    "ru.alfabank.joker.kafka.boot.template.dto.OtpDto:"
        + OtpDto.class.getCanonicalName()
);
```

Push Sender: **Чтение сообщения**

Сломается, если отсутствует header **с типом**

```
props.put(
    JsonDeserializer.TYPE_MAPPINGS,
    "ru.alfabank.joker.kafka.boot.template.dto.OtpDto:"
        + OtpDto.class.getCanonicalName()
);
```

Push Sender: **Чтение сообщения**

Оптимальное решение

```
props.put(JsonDeserializer.USE_TYPE_INFO_HEADERS, false);
props.put(
    JsonDeserializer.VALUE_DEFAULT_TYPE,
    OtpDto.class.getCanonicalName()
);
```

1. **Отказываемся от определения типа из** `header`

Push Sender: **Чтение сообщения**

Оптимальное решение

```
props.put(JsonDeserializer.USE_TYPE_INFO_HEADERS, false);
props.put(
    JsonDeserializer.VALUE_DEFAULT_TYPE,
    OtpDto.class.getCanonicalName()
);
```

1. **Отказываемся от определения типа из** `header`
2. **Ожидаем конкретный тип**

Push Sender: **Чтение сообщения**

Оптимальное решение

```
props.put(JsonDeserializer.USE_TYPE_INFO_HEADERS, false);
props.put(
    JsonDeserializer.VALUE_DEFAULT_TYPE,
    OtpDto.class.getCanonicalName()
);
```

1. **Отказываемся от определения типа из** `header`
2. **Ожидаем конкретный тип**

```
2024-07-18T15:40:49.000+03:00  INFO 47864 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Received from my-topic-2-6449
2024-07-18T15:40:49.000+03:00  INFO 47864 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Processed from my-topic-2-6449
2024-07-18T15:40:49.000+03:00  INFO 47864 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Received from my-topic-2-6450
2024-07-18T15:40:49.000+03:00  INFO 47864 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Processed from my-topic-2-6450
2024-07-18T15:40:49.000+03:00  INFO 47864 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Received from my-topic-2-6451
2024-07-18T15:40:49.000+03:00  INFO 47864 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Processed from my-topic-2-6451
2024-07-18T15:40:52.955+03:00  INFO 47864 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Received from my-topic-0-6039
2024-07-18T15:40:52.956+03:00  INFO 47864 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Processed from my-topic-0-6039
2024-07-18T15:40:52.956+03:00  INFO 47864 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Received from my-topic-0-6040
2024-07-18T15:40:52.956+03:00  INFO 47864 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Processed from my-topic-0-6040
```

Push Sender: Concurrency

Push Sender: Concurrency

```
@Bean
KafkaMessageListenerContainer<String, OtpDto> kafkaListenerContainer(
        ConsumerFactory<String, OtpDto> factory,
        MessageListener<String, OtpDto> listener) {

    var properties = new ContainerProperties(MY_TOPIC);
    properties.setMessageListener(listener);

    return new KafkaMessageListenerContainer<>(factory, properties);
}
```

Было

Push Sender: Concurrency

Было

```
@Bean
KafkaMessageListenerContainer<String, OtpDto> kafkaListenerContainer(
        ConsumerFactory<String, OtpDto> factory,
        MessageListener<String, OtpDto> listener) {

    var properties = new ContainerProperties(MY_TOPIC);
    properties.setMessageListener(listener);

    return new KafkaMessageListenerContainer<>(factory, properties);
}
```

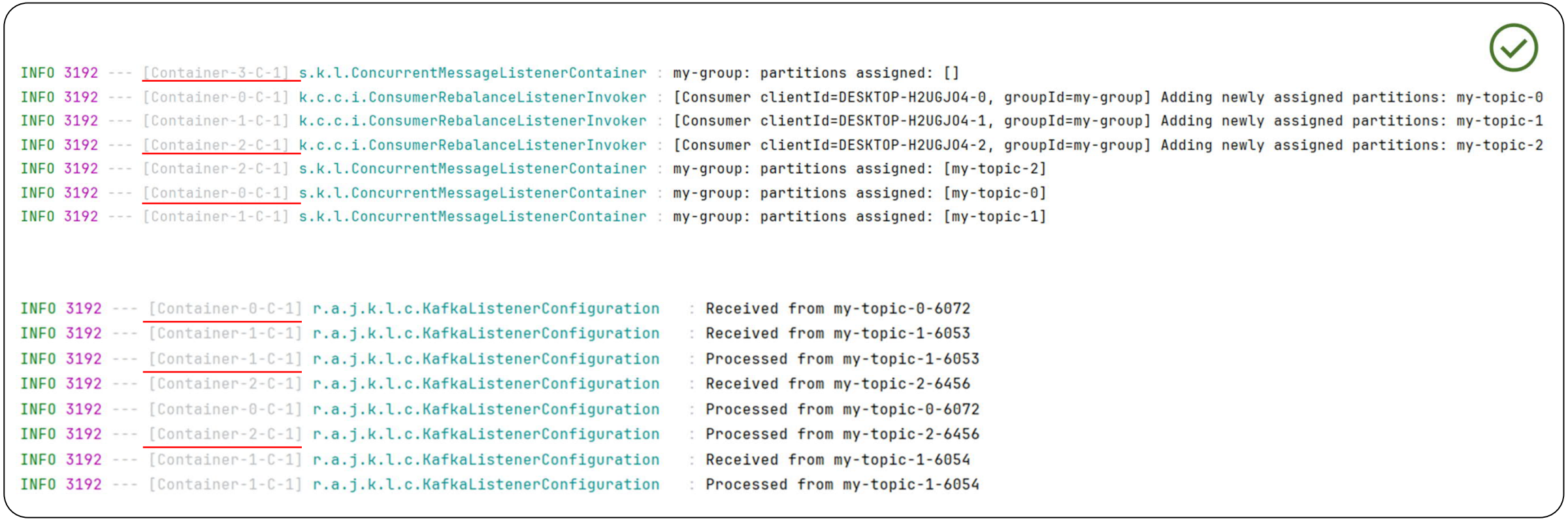
Стало

```
@Bean
ConcurrentMessageListenerContainer<String, OtpDto> kafkaListenerContainer(
        ConsumerFactory<String, OtpDto> factory,
        MessageListener<String, OtpDto> listener
) {
    var properties = new ContainerProperties(MY_TOPIC);
    properties.setMessageListener(listener);

    var container = new ConcurrentMessageListenerContainer<>(factory, properties);
    container.setConcurrency(4);

    return container;
}
```

Push Sender: Concurrency

```
INFO 3192 --- [Container-3-C-1] s.k.l.ConcurrentMessageListenerContainer : my-group: partitions assigned: []
INFO 3192 --- [Container-0-C-1] k.c.c.i.ConsumerRebalanceListenerInvoker : [Consumer clientId=DESKTOP-H2UGJO4-0, groupId=my-group] Adding newly assigned partitions: my-topic-0
INFO 3192 --- [Container-1-C-1] k.c.c.i.ConsumerRebalanceListenerInvoker : [Consumer clientId=DESKTOP-H2UGJO4-1, groupId=my-group] Adding newly assigned partitions: my-topic-1
INFO 3192 --- [Container-2-C-1] k.c.c.i.ConsumerRebalanceListenerInvoker : [Consumer clientId=DESKTOP-H2UGJO4-2, groupId=my-group] Adding newly assigned partitions: my-topic-2
INFO 3192 --- [Container-2-C-1] s.k.l.ConcurrentMessageListenerContainer : my-group: partitions assigned: [my-topic-2]
INFO 3192 --- [Container-0-C-1] s.k.l.ConcurrentMessageListenerContainer : my-group: partitions assigned: [my-topic-0]
INFO 3192 --- [Container-1-C-1] s.k.l.ConcurrentMessageListenerContainer : my-group: partitions assigned: [my-topic-1]
```

Push Sender: Concurrency



```
INFO 3192 --- [Container-3-C-1] s.k.l.ConcurrentMessageListenerContainer : my-group: partitions assigned: []
INFO 3192 --- [Container-0-C-1] k.c.c.i.ConsumerRebalanceListenerInvoker : [Consumer clientId=DESKTOP-H2UGJO4-0, groupId=my-group] Adding newly assigned partitions: my-topic-0
INFO 3192 --- [Container-1-C-1] k.c.c.i.ConsumerRebalanceListenerInvoker : [Consumer clientId=DESKTOP-H2UGJO4-1, groupId=my-group] Adding newly assigned partitions: my-topic-1
INFO 3192 --- [Container-2-C-1] k.c.c.i.ConsumerRebalanceListenerInvoker : [Consumer clientId=DESKTOP-H2UGJO4-2, groupId=my-group] Adding newly assigned partitions: my-topic-2
INFO 3192 --- [Container-2-C-1] s.k.l.ConcurrentMessageListenerContainer : my-group: partitions assigned: [my-topic-2]
INFO 3192 --- [Container-0-C-1] s.k.l.ConcurrentMessageListenerContainer : my-group: partitions assigned: [my-topic-0]
INFO 3192 --- [Container-1-C-1] s.k.l.ConcurrentMessageListenerContainer : my-group: partitions assigned: [my-topic-1]



INFO 3192 --- [Container-0-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Received from my-topic-0-6072
INFO 3192 --- [Container-1-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Received from my-topic-1-6053
INFO 3192 --- [Container-1-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Processed from my-topic-1-6053
INFO 3192 --- [Container-2-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Received from my-topic-2-6456
INFO 3192 --- [Container-0-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Processed from my-topic-0-6072
INFO 3192 --- [Container-2-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Processed from my-topic-2-6456
INFO 3192 --- [Container-1-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Received from my-topic-1-6054
INFO 3192 --- [Container-1-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Processed from my-topic-1-6054
```

Push Sender: Deserialization Error

Нам прислали Integer **в значении**

Push Sender: Deserialization Error

Нам прислали Integer **в значении**

```
    ... 14 common frames omitted
Caused by: com.fasterxml.jackson.databind.exc.MismatchedInputException: Cannot construct instance of `ru.alfabank.joker.kafka.lis
```

Push Sender: Deserialization Error

Решение

```
var jsonDeserializer = new JsonDeserializer<>(OtpDto.class);

ErrorHandlingDeserializer errorHandlingDeserializer
    = new ErrorHandlingDeserializer(jsonDeserializer);

DefaultKafkaConsumerFactory<String, OtpDto> consumerFactory
    = new DefaultKafkaConsumerFactory<>(props);

consumerFactory.setValueDeserializer(
    errorHandlingDeserializer
);
```

Push Sender: Deserialization Error

Решение

```
var jsonDeserializer = new JsonDeserializer<>(OtpDto.class);

ErrorHandlingDeserializer errorHandlingDeserializer
    = new ErrorHandlingDeserializer(jsonDeserializer);

DefaultKafkaConsumerFactory<String, OtpDto> consumerFactory
    = new DefaultKafkaConsumerFactory<>(props);

consumerFactory.setValueDeserializer(
    errorHandlingDeserializer
);
```

1. ErrorHandlerDeserializer

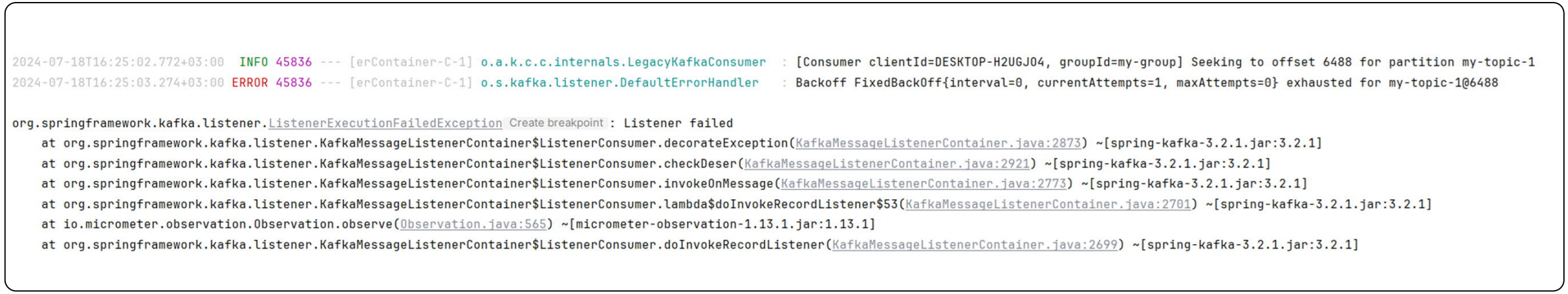
Push Sender: Deserialization Error

Решение

```
var jsonDeserializer = new JsonDeserializer<>(OtpDto.class);

ErrorHandlingDeserializer errorHandlingDeserializer
    = new ErrorHandlingDeserializer(jsonDeserializer);

DefaultKafkaConsumerFactory<String, OtpDto> consumerFactory
    = new DefaultKafkaConsumerFactory<>(props);

consumerFactory.setValueDeserializer(
    errorHandlingDeserializer
);
```

1. ErrorHandlerDeserializer
2. JsonDeserializer – **его делегат**

Push Sender: Deserialization Error



```
2024-07-18T16:25:02.772+03:00  INFO 45836 --- [erContainer-C-1] o.a.k.c.c.internals.LegacyKafkaConsumer  : [Consumer clientId=DESKTOP-H2UGJ04, groupId=my-group] Seeking to offset 6488 for partition my-topic-1
2024-07-18T16:25:03.274+03:00 ERROR 45836 --- [erContainer-C-1] o.s.kafka.listener.DefaultErrorHandler   : Backoff FixedBackOff{interval=0, currentAttempts=1, maxAttempts=0} exhausted for my-topic-1@6488

org.springframework.kafka.listener.ListenerExecutionFailedException Create breakpoint : Listener failed
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.decorateException(KafkaMessageListenerContainer.java:2873) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.checkDeser(KafkaMessageListenerContainer.java:2921) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.invokeOnMessage(KafkaMessageListenerContainer.java:2773) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.lambda$doInvokeRecordListener$53(KafkaMessageListenerContainer.java:2701) ~[spring-kafka-3.2.1.jar:3.2.1]
    at io.micrometer.observation.Observation.observe(Observation.java:565) ~[micrometer-observation-1.13.1.jar:1.13.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.doInvokeRecordListener(KafkaMessageListenerContainer.java:2699) ~[spring-kafka-3.2.1.jar:3.2.1]
```

Push Sender: Deserialization Error

```
2024-07-18T16:25:02.772+03:00  INFO 45836 --- [erContainer-C-1] o.a.k.c.c.internals.LegacyKafkaConsumer  : [Consumer clientId=DESKTOP-H2UGJO4, groupId=my-group] Seeking to offset 6488 for partition my-topic-1
2024-07-18T16:25:03.274+03:00 ERROR 45836 --- [erContainer-C-1] o.s.kafka.listener.DefaultErrorHandler   : Backoff FixedBackOff{interval=0, currentAttempts=1, maxAttempts=0} exhausted for my-topic-1@6488

org.springframework.kafka.listener.ListenerExecutionFailedException Create breakpoint : Listener failed
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.decorateException(KafkaMessageListenerContainer.java:2873) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.checkDeser(KafkaMessageListenerContainer.java:2921) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.invokeOnMessage(KafkaMessageListenerContainer.java:2773) ~[spring-kafka-3.2.1.jar:3.2.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.lambda$doInvokeRecordListener$53(KafkaMessageListenerContainer.java:2701) ~[spring-kafka-3.2.1.jar:3.2.1]
    at io.micrometer.observation.Observation.observe(Observation.java:565) ~[micrometer-observation-1.13.1.jar:1.13.1]
    at org.springframework.kafka.listener.KafkaMessageListenerContainer$ListenerConsumer.doInvokeRecordListener(KafkaMessageListenerContainer.java:2699) ~[spring-kafka-3.2.1.jar:3.2.1]
```

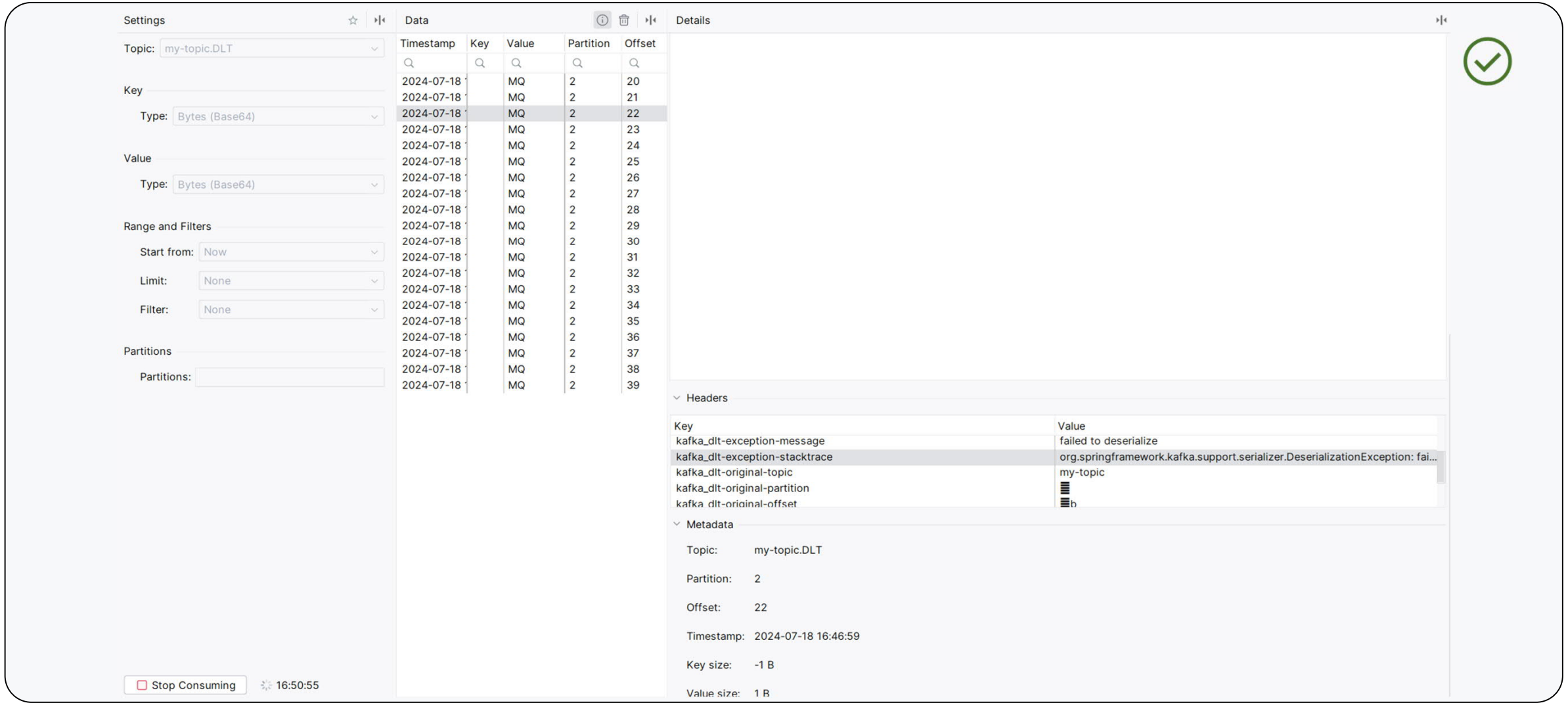
```
2024-07-18T16:25:07.393+03:00  INFO 45836 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Received from my-topic-1-6489
2024-07-18T16:25:07.393+03:00  INFO 45836 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Processed from my-topic-1-6489
2024-07-18T16:25:07.393+03:00  INFO 45836 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Received from my-topic-1-6490
2024-07-18T16:25:07.393+03:00  INFO 45836 --- [erContainer-C-1] r.a.j.k.l.c.KafkaListenerConfiguration   : Processed from my-topic-1-6490
```

Push Sender: Dead Letter Topic

Push Sender: Dead Letter Topic

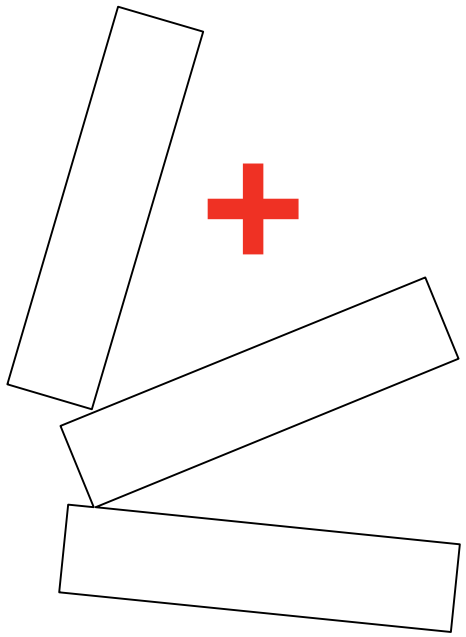
```
@Bean
KafkaMessageListenerContainer<String, OtpDto> kafkaListenerContainer(
        ConsumerFactory<String, OtpDto> factory,
        MessageListener<String, OtpDto> listener,
        KafkaTemplate<byte[], byte[]> template) {

    ContainerProperties containerProperties = new ContainerProperties(MY_TOPIC);
    containerProperties.setMessageListener(listener);

    var listenerContainer = new KafkaMessageListenerContainer<>(
        factory,
        containerProperties
    );

    DeadLetterPublishingRecoverer recoverer = new DeadLetterPublishingRecoverer(template);
    DefaultErrorHandler errorHandler = new DefaultErrorHandler(recoverer);

    listenerContainer.setCommonErrorHandler(errorHandler);

    return listenerContainer;
}
```

Push Sender: Dead Letter Topic



Push Sender: Business Errors

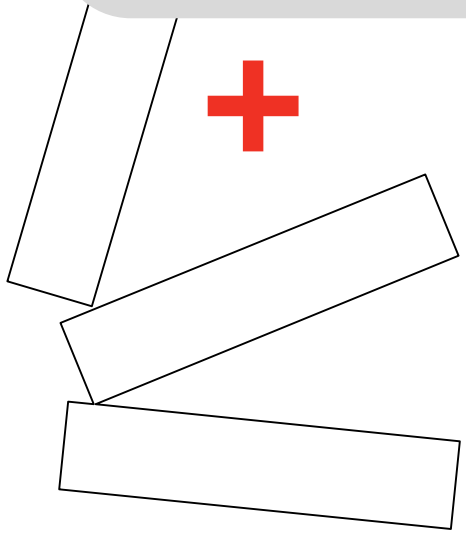
Что ещё может пойти не так?



Push Sender: Business Errors

Что ещё может пойти не так?

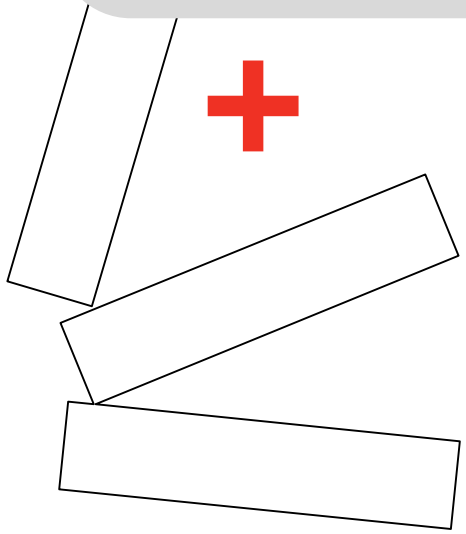
1. **Ошибки могут произойти внутри обработчика**



Push Sender: Business Errors

Что ещё может пойти не так?

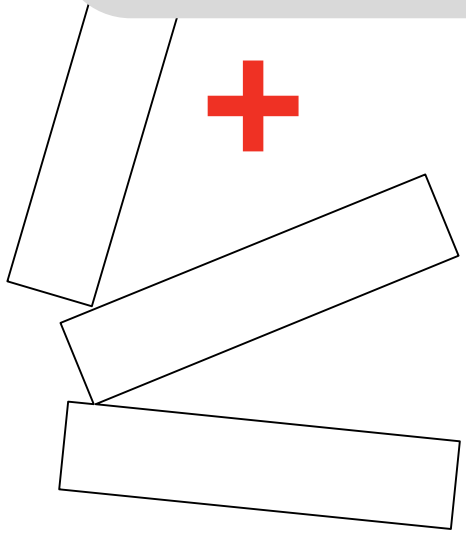
1. **Ошибки могут произойти внутри обработчика**
1.1. Любая непредвиденная



Push Sender: Business Errors

Что ещё может пойти не так?

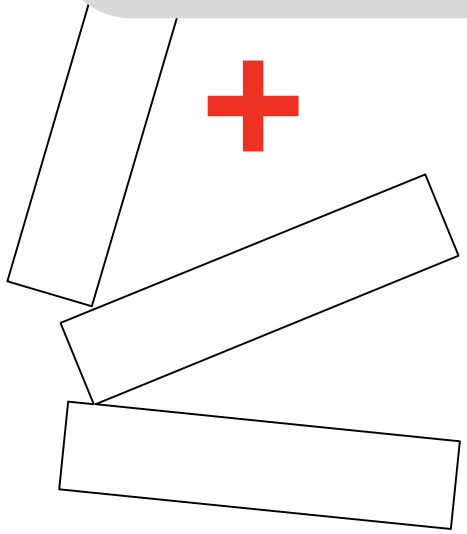
1. **Ошибки могут произойти внутри обработчика**
1.1. Любая непредвиденная
1.2. Недоступен FireBase



Push Sender: Business Errors

Что ещё может пойти не так?

1. **Ошибки могут произойти внутри обработчика**
1.1. Любая непредвиденная
1.2. Недоступен FireBase
1.3. Нас заблокировали в FireBase



Push Sender: Business Errors

Что ещё может пойти не так?

1. **Ошибки могут произойти внутри обработчика**
1.1. Любая непредвиденная
1.2. Недоступен FireBase
1.3. Нас заблокировали в FireBase

2. **Для разных ошибок хорошо бы иметь разные обработчики**

Push Sender: Business Errors

Что ещё может пойти не так?

1. **Ошибки могут произойти внутри обработчика**
1.1. Любая непредвиденная
1.2. Недоступен FireBase
1.3. Нас заблокировали в FireBase

2. **Для разных ошибок хорошо бы иметь разные обработчики**

3. **Для бизнес ошибок и ошибок десериализации хорошо бы иметь разные** DLT

Push Sender: Business Errors

Как это сделать?

Push Sender: Business Errors

CommonDelegatingErrorHandler

```
@Bean
public CommonErrorHandler commonErrorHandler(
        KafkaTemplate<byte[], byte[]> deserializationDltTemplate,
        KafkaTemplate<String, OtpDto> otpKafkaTemplate
) {
    CommonErrorHandler defaultErrorHandler = defaultErrorHandler(otpKafkaTemplate);
    CommonDelegatingErrorHandler delegatingErrorHandler = new CommonDelegatingErrorHandler(defaultErrorHandler);
    delegatingErrorHandler.setErrorHandlers(errorHandlingDelegates(deserializationDltTemplate));
    return delegatingErrorHandler;
}
```

Push Sender: Business Errors

CommonDelegatingErrorHandler

```
@Bean
public CommonErrorHandler commonErrorHandler(
        KafkaTemplate<byte[], byte[]> deserializationDltTemplate,
        KafkaTemplate<String, OtpDto> otpKafkaTemplate
) {
    CommonErrorHandler defaultErrorHandler = defaultErrorHandler(otpKafkaTemplate);
    CommonDelegatingErrorHandler delegatingErrorHandler = new CommonDelegatingErrorHandler(defaultErrorHandler);
    delegatingErrorHandler.setErrorHandlers(errorHandlingDelegates(deserializationDltTemplate));
    return delegatingErrorHandler;
}
```

```
private CommonErrorHandler defaultErrorHandler(
        KafkaTemplate<String, OtpDto> otpKafkaTemplate
) {
    DefaultErrorHandler defaultErrorHandler = new DefaultErrorHandler(
            new DeadLetterPublishingRecoverer(otpKafkaTemplate),
            new FixedBackOff(0, 2)
    );
    return defaultErrorHandler;
}
```

Push Sender: Business Errors

```
private LinkedHashMap<Class<? extends Throwable>, CommonErrorHandler>
errorHandlingDelegates(
        KafkaTemplate<byte[], byte[]> deserializationDltTemplate
) {
    LinkedHashMap<Class<? extends Throwable>, CommonErrorHandler> delegates
        = new LinkedHashMap<>();
    delegates.put(
        DeserializationException.class,
        serDeErrorHandler(deserializationDltTemplate)
    );

    ExponentialBackOff exponentialBackOff = new ExponentialBackOff(250, 2);
    exponentialBackOff.setMaxElapsedTime(3000);
    delegates.put(
            FireBaseUnavailableException.class,
            new DefaultErrorHandler(exponentialBackOff)
    );

    delegates.put(
            FireBaseAccountLockedException.class
            new CommonContainerStoppingErrorHandler()
    );

    return delegates;
}
```

DeserializationException

Push Sender: Business Errors

```
private LinkedHashMap<Class<? extends Throwable>, CommonErrorHandler>
errorHandlingDelegates(
        KafkaTemplate<byte[], byte[]> deserializationDltTemplate
) {
    LinkedHashMap<Class<? extends Throwable>, CommonErrorHandler> delegates
        = new LinkedHashMap<>();
    delegates.put(
        DeserializationException.class,
        serDeErrorHandler(deserializationDltTemplate)
    );

    ExponentialBackOff exponentialBackOff = new ExponentialBackOff(250, 2);
    exponentialBackOff.setMaxElapsedTime(3000);
    delegates.put(
            FireBaseUnavailableException.class,
            new DefaultErrorHandler(exponentialBackOff)
    );

    delegates.put(
            FireBaseAccountLockedException.class
            new CommonContainerStoppingErrorHandler()
    );

    return delegates;
}
```

FireBaseUnavailableException

Push Sender: Business Errors

```
private LinkedHashMap<Class<? extends Throwable>, CommonErrorHandler>
errorHandlingDelegates(
        KafkaTemplate<byte[], byte[]> deserializationDltTemplate
) {
    LinkedHashMap<Class<? extends Throwable>, CommonErrorHandler> delegates
        = new LinkedHashMap<>();
    delegates.put(
        DeserializationException.class,
        serDeErrorHandler(deserializationDltTemplate)
    );

    ExponentialBackOff exponentialBackOff = new ExponentialBackOff(250, 2);
    exponentialBackOff.setMaxElapsedTime(3000);
    delegates.put(
            FireBaseUnavailableException.class,
            new DefaultErrorHandler(exponentialBackOff)
    );

    delegates.put(
            FireBaseAccountLockedException.class
            new CommonContainerStoppingErrorHandler()
    );

    return delegates;
}
```

FireBaseAccountLockedException

Push Sender: Business Errors

```
private CommonErrorHandler serDeErrorHandler(
        KafkaTemplate<byte[], byte[]> deserializationDltTemplate
) {
    DeadLetterPublishingRecoverer serDeRecoverer = new DeadLetterPublishingRecoverer(
            deserializationDltTemplate,
            // BiFunction<ConsumerRecord<?, ?>, Exception, TopicPartition> destinationResolver
            (record, e) -> {
                // my-topic.serde.DLT
                return new TopicPartition(
                    String.format("%s.%s.%s", MY_TOPIC, "serde", "DLT"),
                    record.partition()
                );
            }
    );
    return new DefaultErrorHandler(
            serDeRecoverer,
            new FixedBackOff(0, 0)
    );
}
```

Push Sender: Committing Offsets

Push Sender: Committing Offsets

Какие режимы доступны?

Push Sender: Committing Offsets

Какие режимы доступны?

1. RECORD
2. BATCH
3. TIME
4. COUNT
5. COUNT_TIME
6. MANUAL
7. MANUAL_IMMEDIATE

Push Sender: Committing Offsets

Наш выбор

1. RECORD
2. BATCH
3. TIME
4. COUNT
5. COUNT_TIME
6. MANUAL
7. MANUAL_IMMEDIATE

Push Sender: Committing Offsets

```
props.put(
    ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG,
    false
);
```

Если этого не сделать,
то будет конфликт с Consumer

Push Sender: Committing Offsets

```
props.put(
    ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG,
    false
);
```

Если этого не сделать,
то будет конфликт с Consumer

```
containerProperties.setAckMode(
    ContainerProperties.AckMode.COUNT_TIME
);
containerProperties.setAckTime(5000);
containerProperties.setAckCount(20);
```

1. **Коммитим не реже** 5 **секунд**
2. **Либо при успешной обработке** 20 **сообщений**
3. **Либо после обработки всех сообщений из** poll

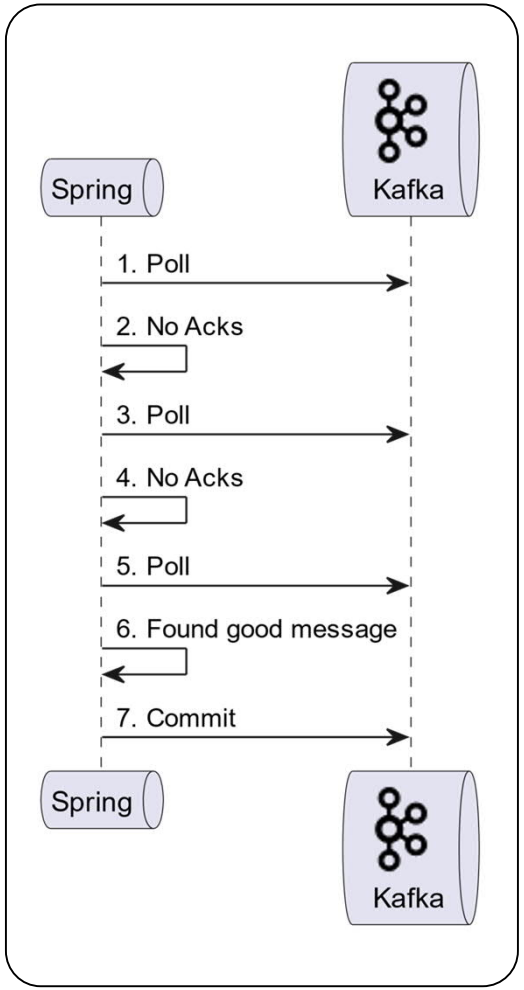
Push Sender: isAckAfterHandle

```
/**
 * Listener container error handling contract.
 *
 * @author Gary Russell
 * @since 2.8
 *
 */
public interface CommonErrorHandler extends DeliveryAttemptAware {

  /**
   * Return true if the offset should be committed for a handled error (no exception
   * thrown).
   * @return true to commit.
   */
  default boolean isAckAfterHandle() {
      return true;
  }
```

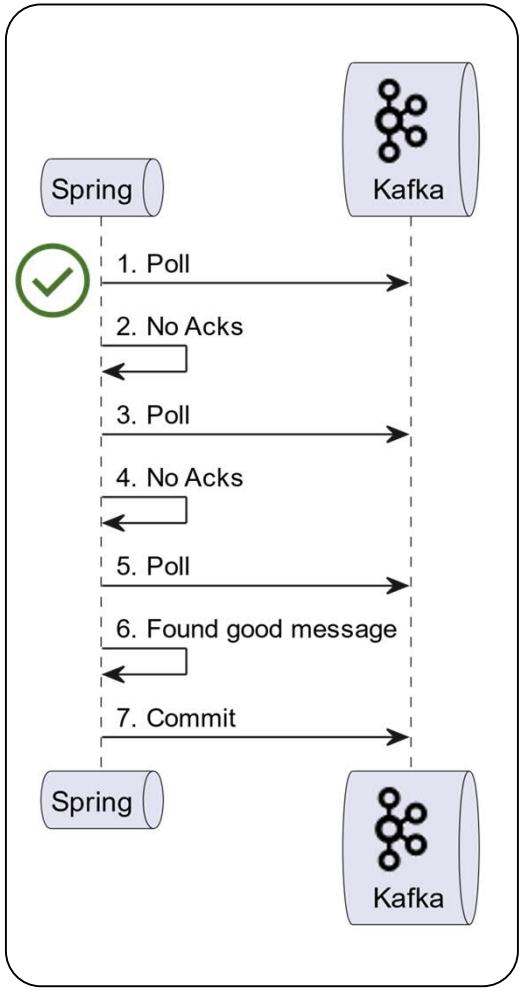
Push Sender: isAckAfterHandle

```
defaultErrorHandler.setAckAfterHandle(false);
```

Push Sender: isAckAfterHandle

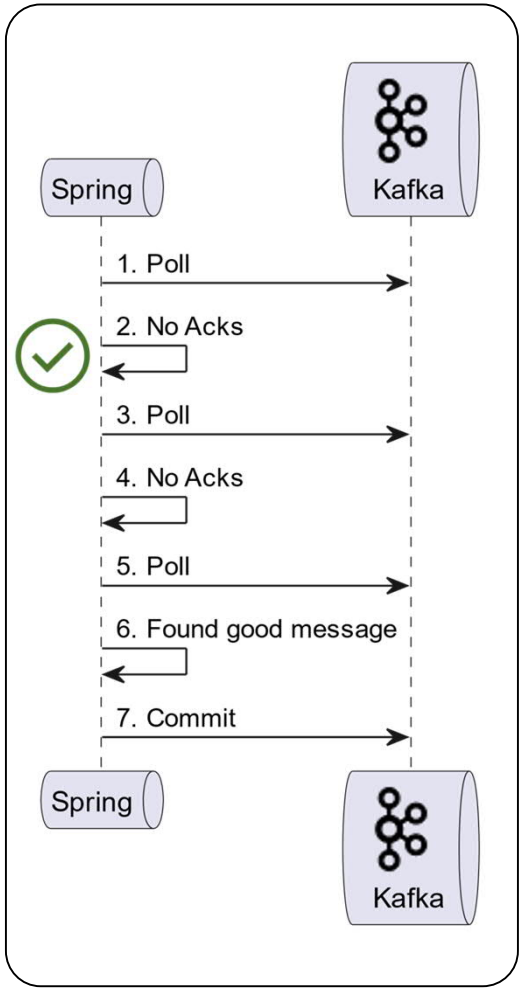


Push Sender: isAckAfterHandle



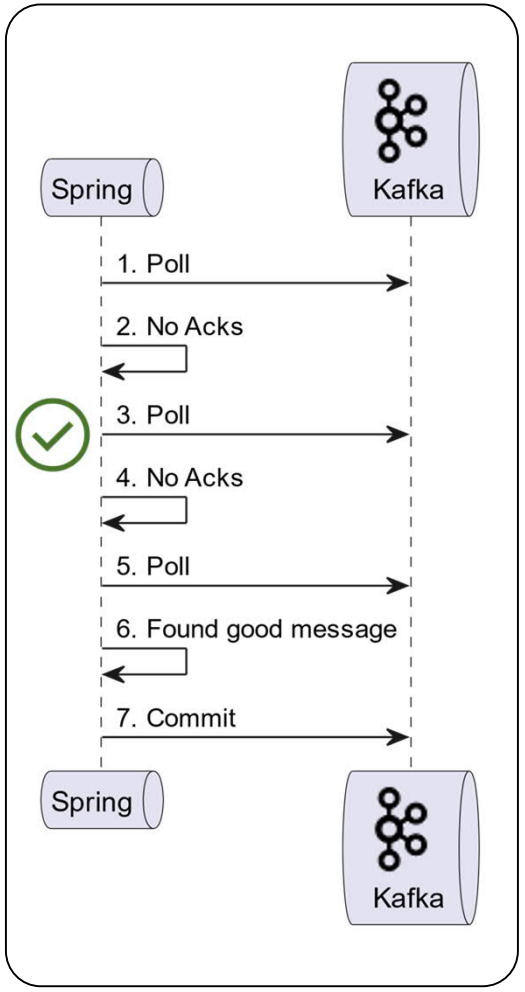
1. Вызван Poll. Получили сообщения с offset 11-20

Push Sender: isAckAfterHandle



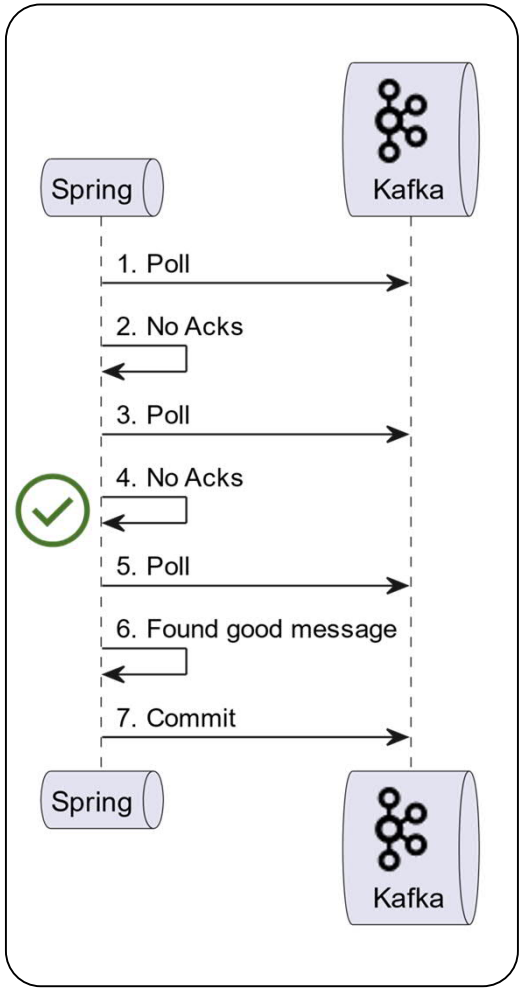
1. Вызван Poll. Получили сообщения с offset 11-20
2. No Acks – на всех сообщениях ошибки

Push Sender: isAckAfterHandle



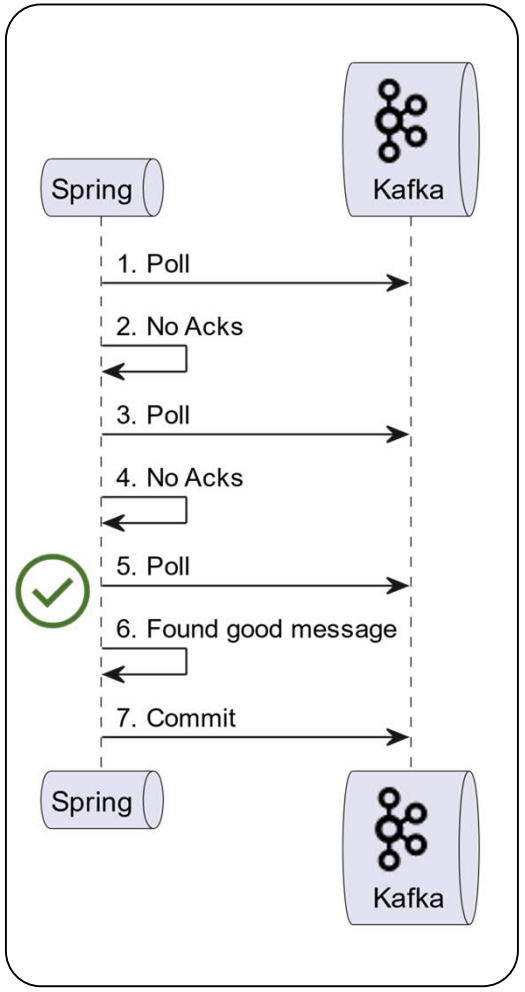
1. Вызван Poll. Получили сообщения с offset 11-20
2. No Acks – на всех сообщениях ошибки
3. Вызван Poll. Получили сообщения с offset 21-30

Push Sender: isAckAfterHandle



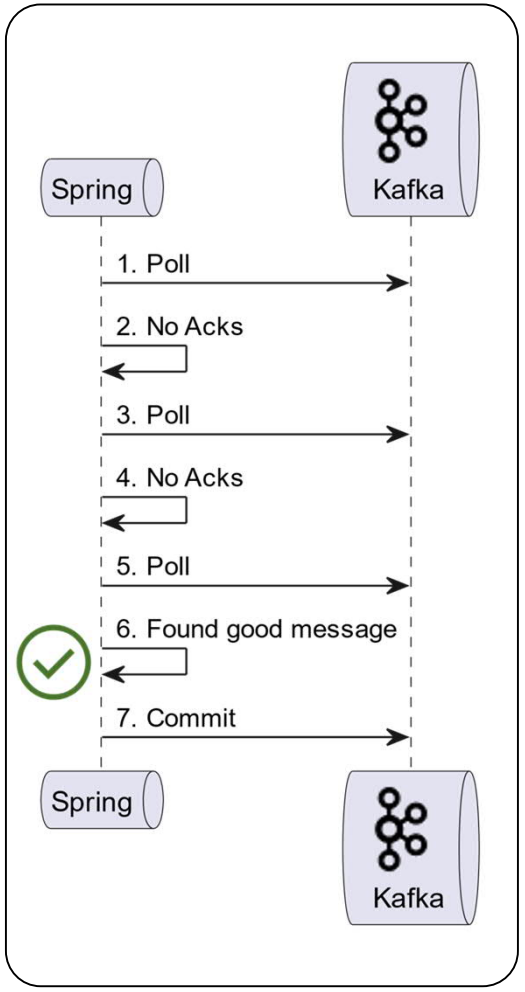
1. Вызван Poll. Получили сообщения с offset 11-20
2. No Acks – на всех сообщениях ошибки
3. Вызван Poll. Получили сообщения с offset 21-30
4. No Acks – на всех сообщениях ошибки

Push Sender: isAckAfterHandle



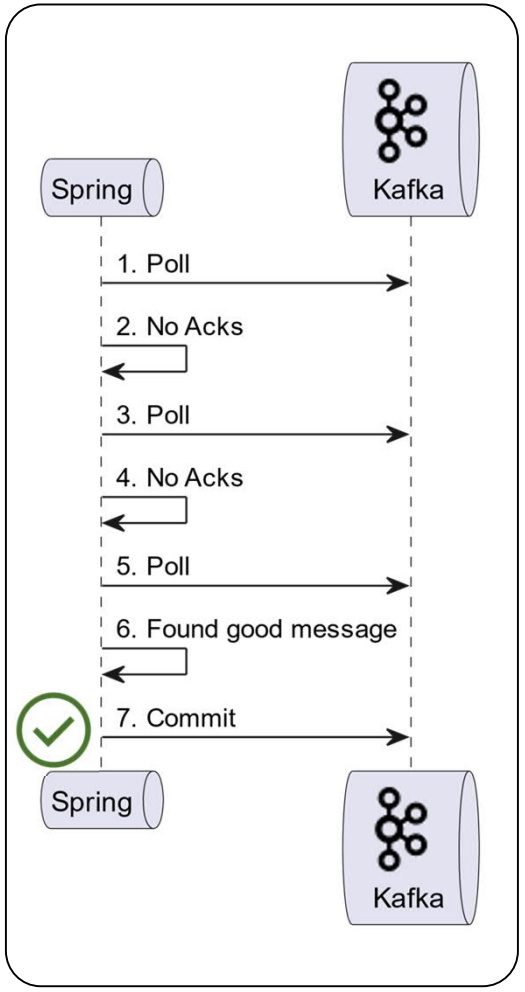
1. Вызван Poll. Получили сообщения с offset 11-20
2. No Acks – на всех сообщениях ошибки
3. Вызван Poll. Получили сообщения с offset 21-30
4. No Acks – на всех сообщениях ошибки
5. Вызван Poll. Получили сообщения с offset 31-40

Push Sender: isAckAfterHandle



1. Вызван Poll. Получили сообщения с offset 11-20
2. No Acks – на всех сообщениях ошибки
3. Вызван Poll. Получили сообщения с offset 21-30
4. No Acks – на всех сообщениях ошибки
5. Вызван Poll. Получили сообщения с offset 31-40
6. No Acks – сообщение с offset=36 успешно обработалось

Push Sender: isAckAfterHandle



1. Вызван Poll. Получили сообщения с offset 11-20
2. No Acks – на всех сообщениях ошибки
3. Вызван Poll. Получили сообщения с offset 21-30
4. No Acks – на всех сообщениях ошибки
5. Вызван Poll. Получили сообщения с offset 31-40
6. No Acks – сообщение с offset=36 успешно обработалось
7. Произошел коммит offset=36

Push Sender: **Автоконфигурация**

Push Sender: **Автоконфигурация**

```yaml
spring:
  kafka:
    bootstrap-servers: localhost:29092
    consumer:
      enable-auto-commit: false
      group-id: "my-group"
      auto-offset-reset: latest
      client-id: "IDDQD"
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
    listener:
      ack-mode: count_time
      ack-count: 20
      ack-time: 5s
      concurrency: 3
```

Push Sender: **Автоконфигурация**

```yaml
spring:
  kafka:
    bootstrap-servers: localhost:29092
    consumer:
      enable-auto-commit: false
      group-id: "my-group"
      auto-offset-reset: latest
      client-id: "IDDQD"
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
    listener:
      ack-mode: count_time
      ack-count: 20
      ack-time: 5s
      concurrency: 3
```

Push Sender: **Автоконфигурация**

```yaml
spring:
  kafka:
    bootstrap-servers: localhost:29092
    consumer:
      enable-auto-commit: false
      group-id: "my-group"
      auto-offset-reset: latest
      client-id: "IDDQD"
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
    listener:
      ack-mode: count_time
      ack-count: 20
      ack-time: 5s
      concurrency: 3
```

Push Sender: **Автоконфигурация**

```
@Slf4j
@Component
@RequiredArgsConstructor
public class MyListener {
    private final PushService pushService;

    @KafkaListener(topics = {"my-topic"})
    public void listen(ConsumerRecord<String, OtpDto> record) {
        log.info(...);
        this.pushService.sendPush(record.value());
        log.info(...);
    }
}
```

Push Sender: **Автоконфигурация**

Из ручной работы работы остается

Push Sender: **Автоконфигурация**

Из ручной работы работы остается

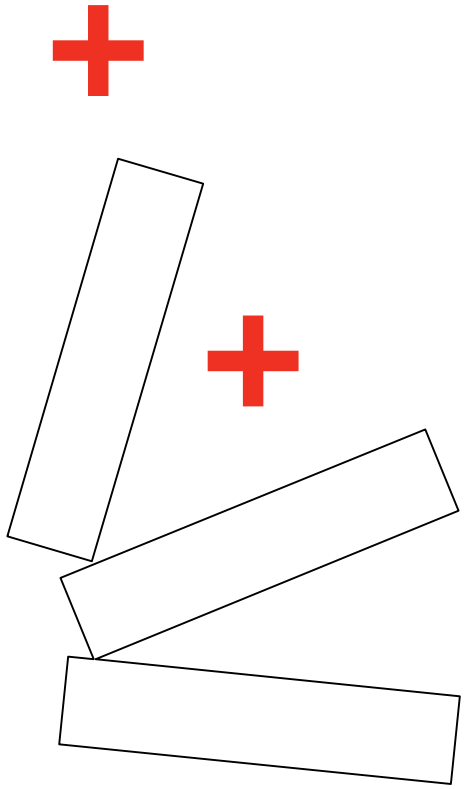
1. Настроить Customizers для Consumer/Producer

Push Sender: **Автоконфигурация**

Из ручной работы работы остается

1. Настроить Customizers для Consumer/Producer
2. Подготовить CommonDelegatingErrorHandler

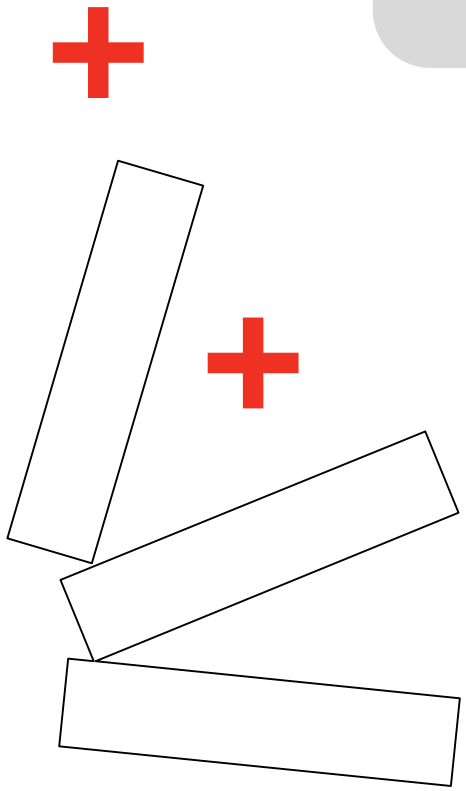
Полезные фичи



Полезные фичи: MessageConverter

Проблемы использования кастомных SerDe

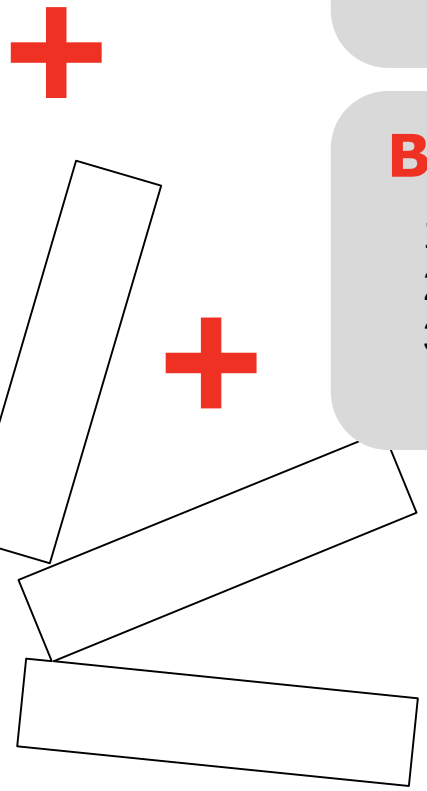
1. Вынуждает погружаться в особенности настройки таких SerDe
2. Часто приводит к ошибкам
3. Вынуждает использовать ErrorHandlingDeserializer



Полезные фичи: MessageConverter

Проблемы использования кастомных SerDe

1. Вынуждает погружаться в особенности настройки таких SerDe
2. Часто приводит к ошибкам
3. Вынуждает использовать ErrorHandlingDeserializer



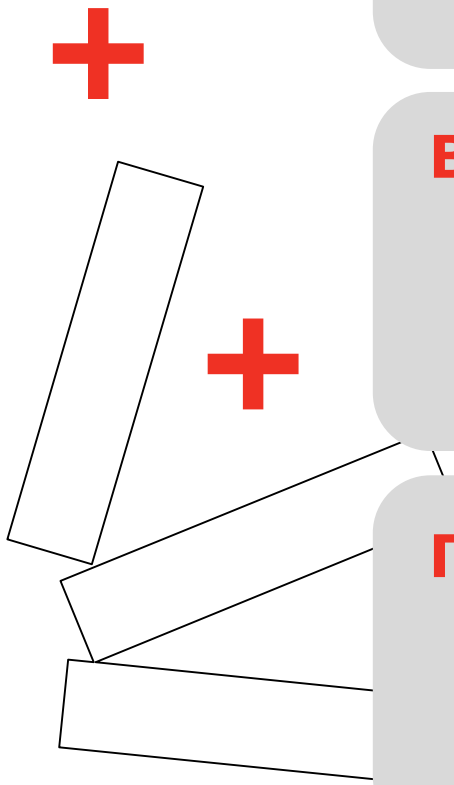
Вариант решения - MessageConverter

1. Для KafkaTemplate используем только String/Bytes/ByteArray Serializers
2. Для Listener используем только String/Bytes/ByteArray Serializers
3. Отказываемся от Consumer/Producer Records в пользу org.springframework.messaging.Message

Полезные фичи: MessageConverter

Проблемы использования кастомных SerDe

1. Вынуждает погружаться в особенности настройки таких SerDe
2. Часто приводит к ошибкам
3. Вынуждает использовать ErrorHandlingDeserializer



Вариант решения - MessageConverter

1. Для KafkaTemplate используем только String/Bytes/ByteArray Serializers
2. Для Listener используем только String/Bytes/ByteArray Serializers
3. Отказываемся от Consumer/Producer Records в пользу org.springframework.messaging.Message

Получаем

1. Освобождение от деталей реализации сложных SerDe
2. Невозможность застрять с ошибками SerDe
3. ConversionException вместо SerDe Exceptions

Полезные фичи: MessageConverter

Для payment-processor

```
@Bean
JsonMessageConverter
messageConverter(ObjectMapper objectMapper) {
    return new StringJsonMessageConverter(
        objectMapper
    );
}
```

Вместо DefaultKafkaProducerFactoryCustomizer
регистрируем JsonMessageConverter

Полезные фичи: MessageConverter

Для payment-processor

```
@Bean
JsonMessageConverter
messageConverter(ObjectMapper objectMapper) {
    return new StringJsonMessageConverter(
        objectMapper
    );
}
```

Вместо DefaultKafkaProducerFactoryCustomizer
регистрируем JsonMessageConverter

```
Message<OtpDto> message = MessageBuilder.withPayload(
    otpDto
).build();
kafkaTemplate.send(message)
```

Используем Message **вместо** Dto

Полезные фичи: MessageConverter

Для push-sender

```
@Bean
JsonMessageConverter
messageConverter(ObjectMapper objectMapper) {
    return new StringJsonMessageConverter(
        objectMapper
    );
}
```

Вместо
DefaultKafkaConsumerFactoryCustomizer **и**
DefaultKafkaProducerFactoryCustomizer
регистрируем JsonMessageConverter

Полезные фичи: MessageConverter

Для push-sender

```
@Bean
JsonMessageConverter
messageConverter(ObjectMapper objectMapper) {
    return new StringJsonMessageConverter(
        objectMapper
    );
}
```

Вместо DefaultKafkaConsumerFactoryCustomizer **и** DefaultKafkaProducerFactoryCustomizer **регистрируем** JsonMessageConverter

```
delegates.put(
    ConversionException.class,
    serDeErrorHandler(deserializationDltTemplate)
);
```

Не смогли распарсить json – **идем в** DLT

Полезные фичи: MessageConverter

Для push-sender

```
@KafkaListener(topics = {"my-topic"})
public void listen(OtpDto otpDto, ConsumerRecordMetadata metadata) {
    …
}
```

Полезные фичи: MessageConverter

Для push-sender

```
@KafkaListener(topics = {"my-topic"})
public void listen(OtpDto otpDto, ConsumerRecordMetadata metadata) {
    …
}
```

```
@KafkaListener(topics = {"my-topic"})
public void listen(OtpDto otpDto,
        @Header(KafkaHeaders.RECEIVED_TOPIC) String topic,
        @Header(KafkaHeaders.RECEIVED_PARTITION) int partition,
        @Header(KafkaHeaders.OFFSET) int offset
        ) {
            …
}
```

Полезные фичи: MessageConverter

Для push-sender

```
@KafkaListener(topics = {"my-topic"})
public void listen(OtpDto otpDto, ConsumerRecordMetadata metadata) {
    …
}
```

```
@KafkaListener(topics = {"my-topic"})
public void listen(OtpDto otpDto,
        @Header(KafkaHeaders.RECEIVED_TOPIC) String topic,
        @Header(KafkaHeaders.RECEIVED_PARTITION) int partition,
        @Header(KafkaHeaders.OFFSET) int offset
        ) {
            …
}
```

```
@KafkaListener(topics = {"my-topic"})
public void listen(Message<OtpDto> message) {
        …
}
```

Полезные фичи: Filter 

Idempotent Receiver

Полезные фичи: Filter 

Idempotent Receiver

```
@KafkaListener(topics = {"my-topic"}, filter = "otpFilterStrategy")
```

Полезные фичи: Filter 

Idempotent Receiver

```
@Bean
public RecordFilterStrategy<String, String>
otpFilterStrategy(ObjectMapper objectMapper) {

    return new RecordFilterStrategy<String, String>() {
        @Override
        @SneakyThrows
        public boolean filter(ConsumerRecord<String, String> record) {
            OtpDto otpDto = objectMapper.readValue(
                                record.value(),
                                OtpDto.class
            );
            return otpDto.expireTime().isBefore(LocalDateTime.now());
        }
    };

}
```

Полезные фичи: @SendTo

```
@SendTo("second-topic")
@KafkaListener(topics = {"my-topic"}, filter = "otpFilterStrategy")
public ResultDto listen(OtpDto otpDto, ConsumerRecordMetadata metadata)
{
    ...
    return new ResultDto(otpDto.userId(), otpDto.code());
}
```

Полезные фичи: @KafkaHandler

Полезные фичи: @KafkaHandler

```java
@Slf4j
@Component
@RequiredArgsConstructor
@KafkaListener(topics = {"my-topic"})
public class MyListener {

    @KafkaHandler
    @SendTo("second-topic")
    public ResultDto listen(
        OtpDto otpDto,
        ConsumerRecordMetadata metadata) {
       ...
    }

    @KafkaHandler
    public void listen(GiveMeYourMoneyDto dto) {
        log.info("Give me your money!");
    }

    @KafkaHandler(isDefault = true)
    public void listenDefault(String str) {
        log.info("What the hell are you? {}", str);
    }
}
```

Полезные фичи: @KafkaHandler

```
@Slf4j
@Component
@RequiredArgsConstructor
@KafkaListener(topics = {"my-topic"})
public class MyListener {

    @KafkaHandler
    @SendTo("second-topic")
    public ResultDto listen(
        OtpDto otpDto,
        ConsumerRecordMetadata metadata) {
       ...
    }

    @KafkaHandler
    public void listen(GiveMeYourMoneyDto dto) {
        log.info("Give me your money!");
    }

    @KafkaHandler(isDefault = true)
    public void listenDefault(String str) {
        log.info("What the hell are you? {}", str);
    }
}
```

Для default **не будет работать**!

```
@Header(KafkaHeaders.RECEIVED_TOPIC)
String topic
```

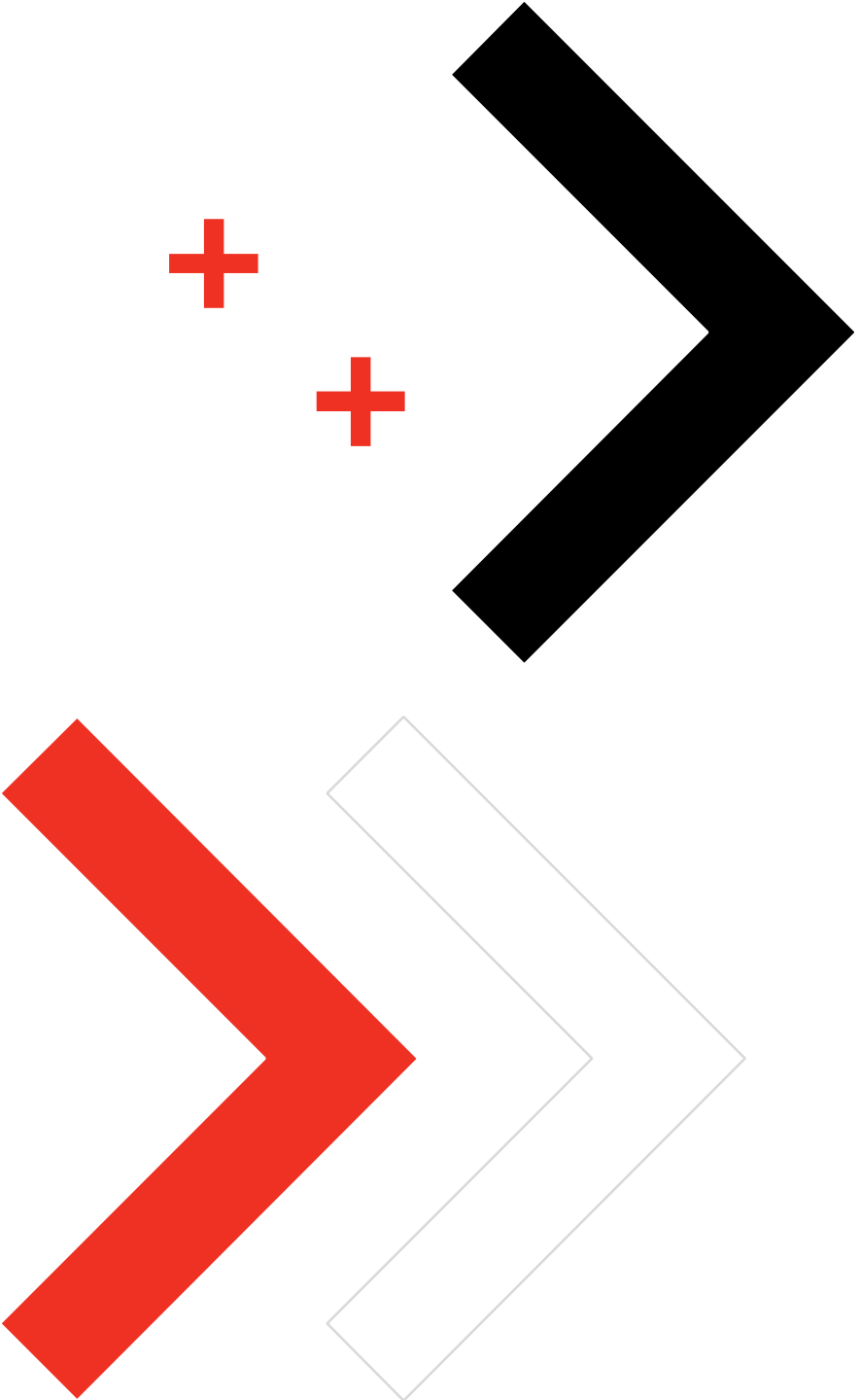
Используйте ConsumerRecordMetadata
или Message.getHeaders()

Полезные фичи: @KafkaHandler

Для доклада все Dto перенесены в один пакет

```
@Bean
JsonMessageConverter messageConverter(ObjectMapper objectMapper) {
    var messageConverter = new StringJsonMessageConverter( objectMapper);

    Jackson2JavaTypeMapper typeMapper = new DefaultJackson2JavaTypeMapper();
    typeMapper.addTrustedPackages("ru.alfabank.joker.kafka.boot.dto");
    typeMapper.setTypePrecedence(Jackson2JavaTypeMapper.TypePrecedence.TYPE_ID);

    messageConverter.setTypeMapper(typeMapper);

    return messageConverter;
}
```

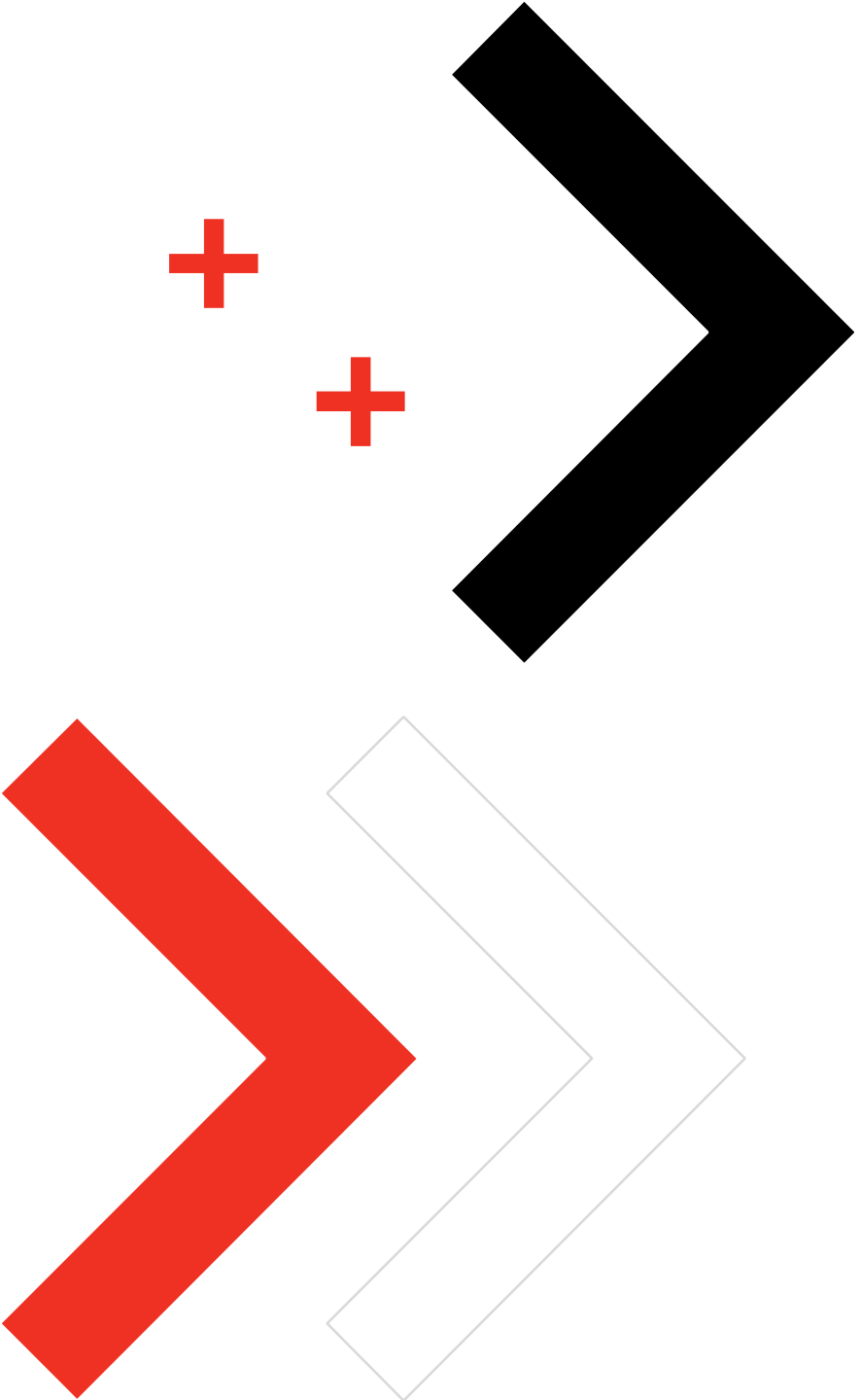
Выводы



Выводы

Spring Kafka

Выводы



Spring Kafka

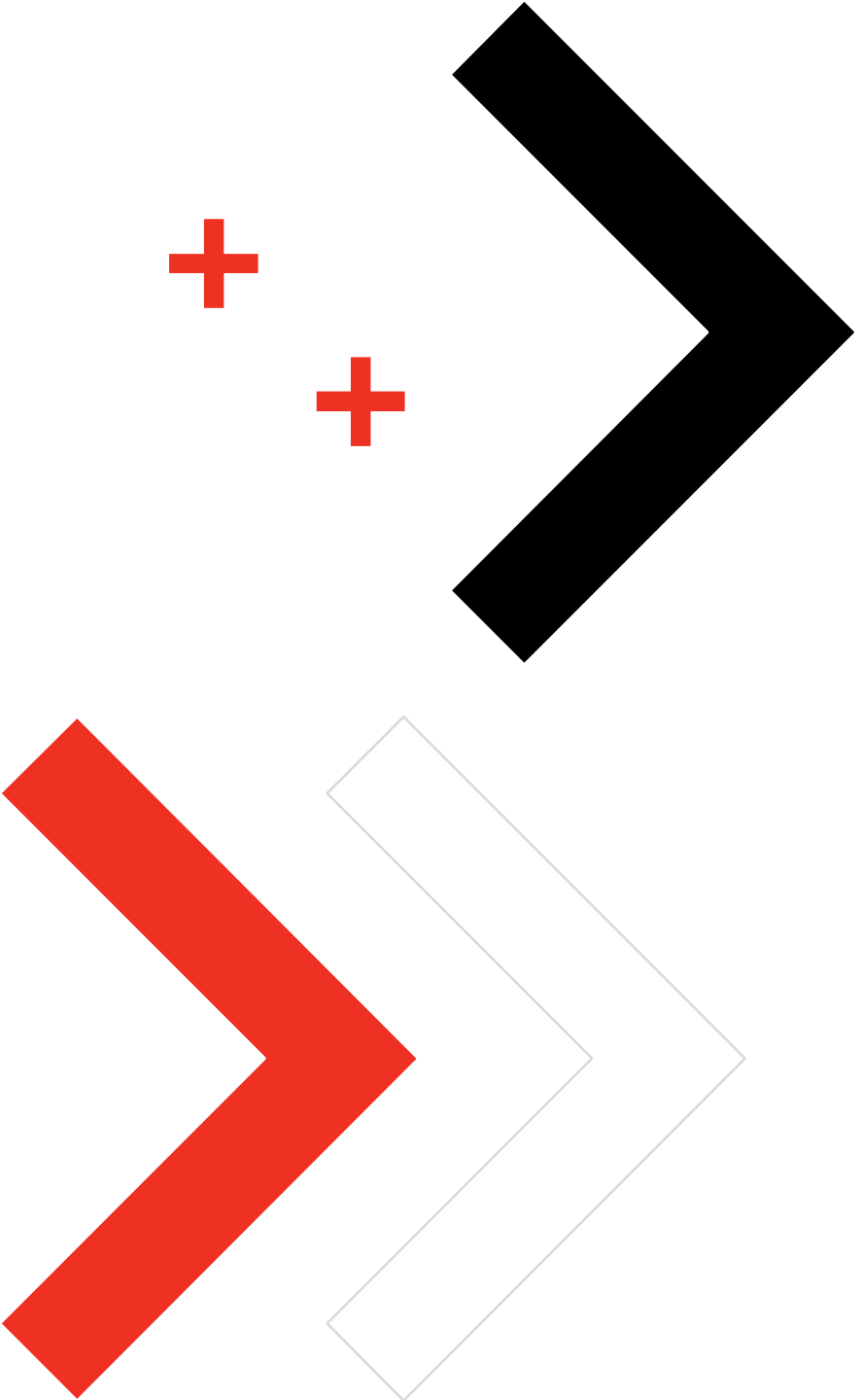
- Мощный инструмент

Выводы

Spring Kafka

- Мощный инструмент
- Неоднозначный

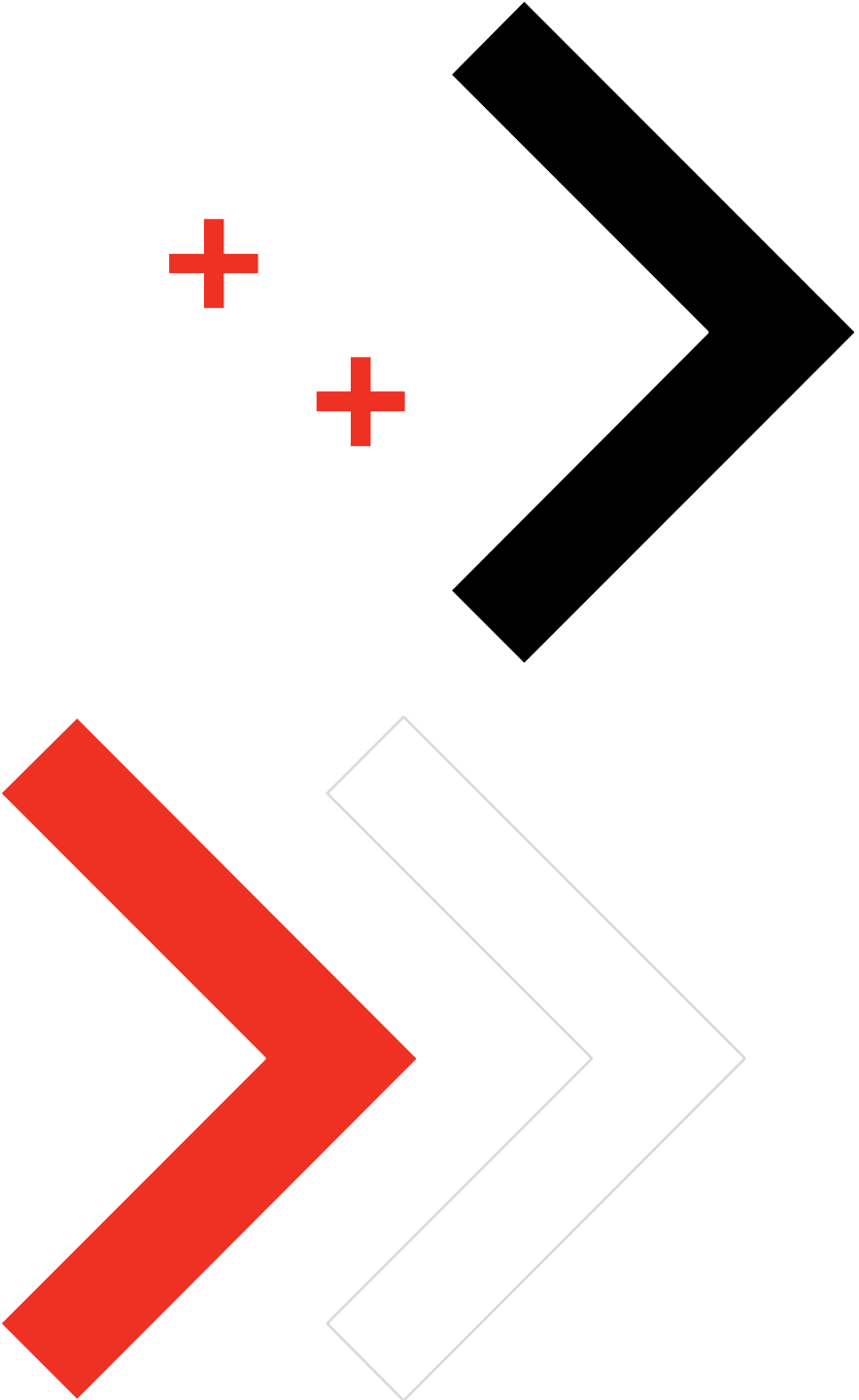
Выводы



Spring Kafka

- Мощный инструмент
- Неоднозначный
- Магии не бывает

Выводы



Spring Kafka

- Мощный инструмент
- Неоднозначный
- Магии не бывает
- Проявляйте активность в Community

СПАСИБО!

Иван Головко

Репозиторий с примерами и ссылками

Alfa Digital

digital.alfabank.ru

Рассказываем о работе в IT и Digital в Альфа-Банке, делимся интересными вакансиями, новостями и полезными советами, иногда шутим