FLOWX.AI

PLATFORM DEEP DIVE / Integrations / creating-a-kafka-producer

Contents

PLATFORM DEEP DIVE / Integrations / Creating a Kafka producer

> **TIP**
>
> This guide focuses on creating a
>
> The fallback content to display on prerendering
> producer using Spring Boot.

Here are some tips, including the required configurations and code samples, to help you implement a Kafka producer in Java.

Required dependencies

Ensure that you have the following dependencies in your project:

```
<dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
</dependency>
```

```
<dependency>
    <groupId>io.strimzi</groupId>
    <artifactId>kafka-oauth-client</artifactId>
    <version>0.6.1</version>
</dependency>

<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>2.5.1</version>
</dependency>

<dependency>
    <groupId>io.opentracing.contrib</groupId>
    <artifactId>opentracing-kafka-client</artifactId>
    <version>0.1.13</version>
</dependency>
```

Configuration

Ensure that you have the following configuration in your `application.yml` or `application.properties` file:

```
spring.kafka:
      bootstrap-servers: URL_OF_THE_KAFKA_SERVER
      producer:
        key-deserializer:
org.apache.kafka.common.serialization.StringSerializer
        value-serializer:
org.springframework.kafka.support.serializer.JsonSerializer
        properties:
          interceptor:
```

```
            classes:
io.opentracing.contrib.kafka.TracingProducerInterceptor
          security.protocol: "SASL_PLAINTEXT"
          sasl.mechanism: "OAUTHBEARER"
          sasl.jaas.config:
"org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginM
required ;"
          sasl.login.callback.handler.class:
io.strimzi.kafka.oauth.client.JaasClientOauthLoginCallbackHand

kafka:
  authorizationExceptionRetryInterval: 10
  ADD_NEEDED_TOPIC_NAMES_HERE # make sure to use the correct na
pattern for topics used to send data to the FLOWX Engine
```

Code sample for a Kafka producer

DANGER

Ensure that you have the necessary KafkaTemplate bean autowired in your producer class. The sendMessage method demonstrates how to send a message to a Kafka topic with the specified headers and payload. Make sure to include all the received Kafka headers in the response that is sent back to the

The fallback content to display on prerendering
.

```
private final KafkaTemplate<String, Object> kafkaTemplate;
```

```
public void sendMessage(String topic, Headers headers,
Object payload) {
  ProducerRecord<String, Object> producerRecord = new
ProducerRecord<>(topic, payload);
  // make sure to send all the received headers back to the
FlowX Engine
  headers.forEach(header ->
producerRecord.headers().add(header));
  kafkaTemplate.send(producerRecord);
}
```

Was this page helpful?