Dependencia de Maven

Con los cambios sobre la API de Twitter, las nuevas cuentas ya no podrán usar la versión 1 de la API. En esta lección aprenderemos cómo implementar el productor usando la nueva api versión 2 de Twitter.

Existe una dependencia para java que implementa esta nueva API. Para usarla, la incluiremos en nuestra lista de dependencias de Maven editando el fichero pom.xml

```
<dependency>

  <groupId>com.twitter</groupId>

  <artifactId>twitter-api-java-sdk</artifactId>

  <version>2.0.3</version>

</dependency>
```

Junto a la lección tienes el enlace del proyecto, donde podrás ver la última versión y la descripción de todas las funcionalidades que tiene implementadas.

Nuestra clase *ProductorTweets* va a usar la librería para consumir de la API de Twitter en tiempo real y producir los mensajes de Kafka.

API de Twitter

Para interaccionar con la API de Twitter, nos debemos registrar en el portal de desarrolladores y dar de alta una nueva aplicación: https://developer.twitter.com/en

El registro es inmediato, solamente debes indicar la finalidad para la que vas a usar la API con una breve descripción. Puedes indicar aquí que es con una finalidad formativa sin ningún problema.

Si navegamos a la pestaña claves y tokens veremos que nos aparecen varias claves. La que usaremos para implementar la práctica es la llamada bearer token. Debemos hacer click en regenerar y copiarnos la clave.

Nuestra aplicación cogerá esta clave del primer argumento, por lo en nuestro IDE navegaremos a run, edit configurations, seleccionamos la clase, que *ProductorTweets* y la pegamos en el espacio que existe con nombre argumentos.

Implementación del productor

A continuación tienes el código completo para la clase ProductorTweets:

```
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.twitter.clientlib.ApiException;
import com.twitter.clientlib.TwitterCredentialsBearer;
import com.twitter.clientlib.api.TwitterApi;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerConfig;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.HashSet;
import java.util.Properties;
import java.util.Set;

public class ProductorTweets {

    public final static String TOPIC_NAME = "rawtweets";
    public static ObjectMapper objectMapper = new ObjectMapper();

    public static void main (String[] args){

        String bearerToken = args[0];

        Properties props = new Properties();
        props.put("acks", "1");
        props.put("retries", 3);
        props.put("batch.size", 16384);
        props.put("buffer.memory", 33554432);
        props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");

        final KafkaProducer<String, String> prod = new KafkaProducer<>(props);

        TwitterApi apiInstance = new TwitterApi(new
TwitterCredentialsBearer(bearerToken));

        Set<String> tweetFields = new HashSet<>();
        tweetFields.add("entities");
        tweetFields.add("lang");

        try {
            // findTweetById
            InputStream result = apiInstance.tweets().sampleStream()
                    .tweetFields(tweetFields)
                    .execute();
            try{
                BufferedReader reader = new BufferedReader(new
InputStreamReader(result));
```

```
                String line = reader.readLine();
                while (line != null) {
                    if(line.isEmpty()) {
                        System.out.println("==> Empty line");
                        line = reader.readLine();
                        continue;
                    }
                    JsonNode root;
                    try {
                        root = objectMapper.readTree(line);
                        JsonNode hashtagsNode =
root.path("data").path("entities").path("hashtags");
                        if (!hashtagsNode.toString().equals("")) {
                            String value =
root.path("data").path("text").toString();
                            String lang =
root.path("data").path("lang").toString();
                            System.out.println(lang);
                            System.out.println(value);
                            prod.send(new
ProducerRecord<>(ProductorTweets.TOPIC_NAME, lang, value));
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    line = reader.readLine();
                }
            }catch (Exception e) {
                e.printStackTrace();
                System.out.println(e);
            }
        } catch (ApiException e) {
            System.err.println("Exception when calling
TweetsApi#sampleStream");
            System.err.println("Status code: " + e.getCode());
            System.err.println("Reason: " + e.getResponseBody());
            System.err.println("Response headers: " + e.getResponseHeaders());
            e.printStackTrace();
        }
    }
}
```

Como ves, tenemos que definir el nombre del topic, en mi caso voy a usar *rawtweets*.

```
public final static String TOPIC_NAME = "rawtweets";
```

El siguiente paso consiste en configurar las propiedades del productor. Aquí le indicamos los valores del ACK, reintentos, tamaños de batch y memoria, los servidores de bootstrap y los serializadores. En este caso usaremos de tipo string para almacenar el contenido del tweet en formato JSON.

```
Properties props = new Properties();

props.put("acks", "1");

props.put("retries", 3);

props.put("batch.size", 16384);

props.put("buffer.memory", 33554432);

props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");

props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
```

Después, inicializamos el productor de Kafka en la variable llamada *prod*.

```
final KafkaProducer<String, String> prod = new KafkaProducer<>(props);
```

Ahora, debemos configurar la librería. Para ello, inicializamos una instancia de la API llamando a *TwitterApi* y le indicamos que vamos a autenticarnos con el token que acabamos de copiar.

Una vez inicializado, también indicaremos los campos relacionados con cada tweet en los que estamos interesados. En nuestro caso, queremos que el productor escriba en Kafka el contenido de los tweets que contengan al menos un hashtag. También, usaremos el idioma para agrupar más adelante los hashtags en función del idioma y del número de ocurrencias.

Para hacer esto, le indicamos al cliente de twitter que incluya en la información que vamos a recibir el campo entities, que contendrá los hashtags y el campo lang, que indica el idioma.

Ahora, llamamos a la función *samplestream* indicándole los parámetros que acabamos de configurar. Esta función nos devolverá un stream de tweets en la variable result.

Una vez que tenemos el stream vamos a parsear la información recibida, que está en formato JSON. Para ello, ignoramos los mensajes vacíos.

Por último, debemos escribir el bloque de código que se va a encargar de escribir el tweet en Kafka.

Antes de escribir en Kafka, comprobaremos que el tweet contiene algún hashtag. Si no es así, descartaremos este registro y pasaremos al siguiente.

En el caso de que contenga algún hashtag, procederemos a mediante el productor de Kafka un nuevo registro. Aquí, a la función *send*, le indicamos un nuevo objeto de tipo producer record con tres argumentos: el nombre del topic, la clave, que en este caso va a ser el idioma y por último el texto del tweet.

Prueba del productor

Si queremos comprobar que funciona correctamente, podemos comentar la línea que se encarga de enviar la escritura a Kafka, ya que no tenemos levantado el clúster, y añadir en su lugar una escritura por consola:

```
System.out.println(value);
```

Esta línea nos imprimirá cada tweet que enviaríamos a Kafka.

Lo ejecutamos dándole a run -> *ProductorTweets*. Recordad el argumento con el token de twitter

Vemos que en consola nos aparece una serie de tweets.

Con esto, ya tendríamos implementado el primer componente que producirá los tweets a kafka.