MoM

Einführung

In diesem Projekt, werde ich einen Message Producer und einen MessageConsumer erstellen. Diese werden das Prinzip von MoM umsetzen.

Projektbeschreibung

Es existiert bereits eine Applikation, die Wahldaten von einem Wahllokal generiert und ausgibt. Nun sollen diese Daten auf einem anderen Consumer ausgelesen werden.

Theorie

Bei dieser Aufgabe muss man verstehen, wie Apache Kafka funktioniert und wie man sich in eine Topic hinzufügen kann.

Arbeitsschritte

Installation von Apache Kafka

Als erstes musste man sich in Apache Kafka installieren. Das macht man in der Docker Shell. Um zu überprüfen, ob das funktioniert hat, kann man testen, ob man Daten in eine Topic hinzufügen kann, um sie dann in einem anderen Terminal auszulesen:

```
bin/kafka-topics.sh --create --topic quickstart-events --boot
```

```
bin/kafka-console-producer.sh --topic quickstart-events --boo
```

```
bin/kafka-console-consumer.sh --topic quickstart-events --fro
```

Ausprobieren der Demo

Im Kurs gibt es ein Github Repository, bei dem ein Producer und ein Consumer Daten austauschen. Um das auszuprobieren muss man nur die MessageApplication starten und http://localhost:9092 aufrufen. Von dort aus kann man mit dem Parameter /send?message=”Text” einen beliebigen String an den Consumer schicken.

Um konfigurieren auf ElectionData

Als erstes muss man die ElectionData, PreferenceData und PartyVotes Dateien rüberkopieren. Daraufhin kann man die Dateien von der Adresse aufrufen und mit folgendem Befehl auslesen und ausgeben:

```
RestTemplate rt= restTemplateBuilder.build();
ElectionData ed = rt.getForObject(url, ElectionData.class);
aTemplate.send("quickstart-events", ed.toString());
```

ed.toString() hat alle meine Daten in ein String Format zusammengefasst. Diese werden dann vom Consumer ausgelesen.

Umwandeln in XML

```
@RequestMapping(value = "/sendXML", produces = MediaType.APPL
    public ElectionData sendXML() {
        RestTemplate restTemplate = restTemplateBuilder.build
        ElectionData electionData = restTemplate.getForObject
                kafkaTemplate.send("quickstart-events", elect
        return electionData;
    }
```

Diese Methode braucht die jackson-XML Library, diese wird durch

```
implementation 'com.fasterxml.jackson.dataformat:jackson-data
```

in gradle importiert.

Mehrere Wahllokale realisieren

Durch meine vorherige Implementierung der ElectionData kann ich in mir in der Adresse aussuchen welches Wahllokal ich auslese. Das wird durch eine inID

angegeben. Wenn ich diese ID instanzieren will muss ich folgenden Code benutzen:

```
@RequestMapping("/sendJSON/{inID}")
    public ElectionData sendMessage(@PathVariable("inID") Str
```

Hier wird bekanntgegeben, , dass ich die inID als Variable benutzen will, die im Pfad der Adresse eingegeben wird. Dieser String kann dann danach wie jeder andere String weiterbenutzt werden. Für den speziellen Fall, dass ich den String in einen weiteren String einfügen muss, brauche ich die Methode String.format(). Dort ersetze ich den Platz des String mit %s und gebe als zweiten Parameter die inID an. Diese URL wird dann benutzt um die Daten aus der Wahlzentrale zu erhalten.

Success Message

Für den Fall, dass man eine Erfolgsmeldung haben will, muss man ein neues Apache Kafka Topic mit dem oben genannten Befehl erstellen. Dieses hat dann zum Beispiel den Namen “success-response”. Nun, wenn wir beim Consumer eine Nachricht erhalten, können wir eine Nachricht in das neu erstellte Topic einfügen. Diese Nachricht kann dann von unserem Producer aus wieder mit einem KafkaListener wie beim Consumer ausgelesen und auf der Konsole ausgegeben werden.

Zusammenfassung

Durch MoM habe ich eine funktionierende Kommunikation zwischen zwei Rechnern realisiert. Die empfangenen Daten werden entweder im JSON oder im XML Format dargestellt.

Quellen

https://medium.com/@abhishekranjandev/a-comprehensive-guide-to-integrating-kafka-in-a-spring-boot-application-a4b912aee62e

http://www.academictutorials.com/jms/jms-introduction.asp