Spring Boot + Apache Kafka API

=====

==========

=> Spring Boot gives built-in support for apache kafka.. It even

gives certain objects automatically through AutoConfiguration process

added

=> if spring-boot-starter-apache-kafka depenecies to app then we get KafkaTemplate<K,V> class object through autoConfiguration.. which internally takes care of creating KafkaProducer, ProducerRecord objects that are required to send messages/data.

In Producer class or comp

@Autowired

=>Kafka Producer internally sends the Message as ProducerRecord object

private KafkaTemplate<String, String> template;

=> we can place @KafkaListener(topicName=".....", groupId=".....") annotation

on the methodListener class to make Message broker to collect the message

recived to topic section i.e it internally takes care of creating KafkaConsumer<String, String> obj and ConsumerRecord object.

=>Kafka Cosnumer internally recives the Message as Consumer Record object

In Listener class

note:: KafkaProducer sends the message as ProducerRecord object recieves note:: KafkaConsumer the message as Consumer Record object

@KafkaListener(topicName="nit-tpc-fri", groupId="nit-grp1")

public void readMessage(String msg){

//logic to read the message.

}

=>We need to add

@EnableKafka on the top of starter class (main class)

=> we can get message from endusers as request parameter through

RestController/MicorService of spring MVC/Spring Rest App, So we need to make

the Producer App taking message

through RestController to send kafka setup.

Similarly we need to read the message from kafka setup using Consumer or MEssageListener to send to endusers through RestController.

http://localhost:4041/KafkaApp/send?message=welcome

http://localhost:4041/KafkaApp/send?message=quotation value 2000
http://localhost:4041/KafkaApp/send?message=how are u

MessageProducer

(1a)

KafkaApp/send?message=welcome

(1b)

(1c)

ser1

Success

(1i)

Dispather Servlet

(1h)

RestController (MicroService) (1g)

(spring Bean) @Autowired KafkaTemlate template;

Apache Kafka Eco System

Bootstrap server

(1d)

(1f)

template.send(msg)

MEssage broker

Topic

(1e)

KafkaApp/readAll (2e)

(2f)

(2g) MEssageStore

MessageListener (Consumer) @kafkalistener(.....)

Message broker

2a

List collection public void

user2

2b

2i

readMessaage(String msg){

2h

2d

gives

2j

2c

all messages

}

(1a) --- 1i :: 1st request -response (sending messages)

(2a) --- 2j :: 2nd request-response (reading all messages at a time)

order of development

of

=>MessageProducer having injection KafkaTemplate obj

=> Restcontroller with handler method with "/send" request path

having injection of MessageProducer object

=> MessageStore as spring bean

=>MessageListner injected with MessageStore

=> above Restcontroller with another handler method with "/readAll"

request path having injection of MessageStore object

step1) Create spring boot starter Project adding the

following dependencies web, kafka, lombok api, devtools

step2) add @Enablekafka on the top of main class/starter class.

```
@SpringBootApplication
@EnableKafka
public class BootKafka Proj2RestWithKafkaApplication {
}
public static void main(String[] args) {
}
SpringApplication.run(BootKafka Proj2RestWithKafkaApplication.class, args);
```

step3) add the following properties in application.properties file

applicaiton.properties

```
#server port
server.port=4041
#context path
server.servlet.context-path=/RestKafkaApp
#topic name
app.topic.name=nit-tpc-sat1
#Producer properites
spring.kafka.producer.bootstrap-servers=localhost:9092
spring.kafka.producer.key-serializer=org.apache.kafka.common.serialization. StringSerializer
Producer/Publisher sends the message
spring.kafka.producer.value-serializer=org.apache.kafka.common.serialization. StringSerializer to kafka
setup as Serialized key=value pair
#Consumer properites
spring.kafka.consumer.bootstrap-servers=localhost:9092
spring.kafka.consumer.key-deserializer=org.apache.kafka.common.serialization. StringDeserializer
spring.kafka.consumer.value-deserializer=org.apache.kafka.common.serialization.StringDeserializer
```

Consumer/Subscriber reads the message from kafka setup as DeSerialized key=value pair

step4) MessageProducer.java

```
package com.nt.producer;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value; import
org.springframework.kafka.core.KafkaTemplate;
import org.springframework.stereotype.Component;
@Component("msgProducer")
public class MessageProducer {
@Autowired
private KafkaTemplate<String, String> template; @Value("${app.topic.name}")
private String topicName;
public String sendMessage(String message) {
template.send(topicName,message);
return " message delivered ";
}
}
```

step5)

Restcontroller

```
package com.nt.controller;
import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.web.bind.annotation.GetMapping; import
org.springframework.web.bind.annotation.RequestParam; import
org.springframework.web.bind.annotation.RestController;
import com.nt.consumer.MessageStore; import com.nt.producer.MessageProducer;
@RestController
public class KafkaMessageHandlingController {
package com.nt.advice;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ExceptionHandler; import
org.springframework.web.bind.annotation.RestControllerAdvice;
@RestControllerAdvice
public class Kafka MessageOpeartionsControllerAdvice {
public KafkaMessageOpeartionsControllerAdvice() {
}
System.out.println("KafkaMessageOpeartionsControllerAdvice: 0-param construtor");
@ExceptionHandler(Exception.class)
```

```
@Autowired
private MessageProducer producer;
@Autowired
private MessageStore store;
@GetMapping("/send")
public String sendMessage(@RequestParam("message") String message) {
String status-producer.sendMessage(message);
return "<h1>"+status+"</h1>";
}
@GetMapping("/readAll")
public String fetchAllMessage() {
return "<h1>"+store.getAllMessages()+"</h1>";
}
}
```

step6)

MessageStore.java

==========

```
package com.nt.consumer;
```

step7)

```
import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Component;
@Component
public class MessageStore {
private List<String> listMessages=new ArrayList();
public void addMessage(String message) {
}
listMessages.add(message);
public String getAllMessages() {
return listMessages.toString();
}
```

MessageConsumer.java

```
package com.nt.consumer;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Component;
@Component
```

```
public class MessageConsumer {
@Autowired
private MessageStore store;
@KafkaListener(topics = "${app.topic.name}",groupId = "grp1")
public void readMessage(String message) {
//add message to store
store.addMessage(message);
}
}
```

Exection order

i) start bootstrap server (zookeeper)

```
}
public ResponseEntity<String> handleAllExceptions (Exception e){
System.out.println("Kafka MessageOpeartionsControllerAdvice.handleAllExceptions()");
return new ResponseEntity<String>("Internal Problem:" +
e.getMessage(),HttpStatus.INTERNAL_SERVER_ERROR);
```

E:\kafka_2.13-3.1.0\bin\windows>zookeeper-server-start.bat E:\kafka_2.13-3.1.0\config\zookeeper.properties

ii) start kafka server setup

E:\kafka_2.13-3.1.0\bin\windows>kafka-server-start.bat E:\kafka_2.13-3.1.0\config\server.properties

note:: No need of creating topic seperately.. @EnableKakfa will take care of creating topic dynamically

iii) Run the application as spring boot App or on server

and give requests

http://localhost:4041/RestkafkaApp/send?messsage=raja
http://localhost:4041/RestkafkaApp/send?messsage=rani

http://localhost:4041/RestkafkaApp/send?messsage=hello

http://localhost:4041/RestkafkaApp/readAll

[raja,rani,hello]

Example App on Apache kakfka (As Spring boot Standalone Apps)

=========

==========

>

BootKafkaProj01-Producer [boot] [devtools]

src/main/java

#com.nt place @EnableKafka in main class

› BootKafkaProj01 ProducerApplication.java

com.nt.producer

> KafkaMessageProducer.java

src/main/resources

application.properties

src/test/java

> JRE System Library [JavaSE-17]

>

Maven Dependencies

> target/generated-sources/annotations

target/generated-test-sources/test-annotations

application.properties

spring.application.name=BootKafka Proj01-Producer

#topic name (user-defined)

app.tpc.name=tpc619

#Producer properties

spring.kafka.producer.bootstrap-server-localhost:9092

spring.kafka.producer.key-serializer=org.apache.kafka.common.serialization.StringSerializer
spring.kafka.producer.value-serializer=org.apache.kafka.common.serialization.StringSerializer

src

> target

HELP.md

mvnw

mvnw.cmd

Mpom.xml

Publisher App

==========

package com.nt.producer;

import java.util.Scanner;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.boot.CommandLineRunner; import org.springframework.kafka.core.Kafka Template;

import org.springframework.stereotype.Component;

@Component

public class Kafka MessageProducer implements CommandLineRunner { @Autowired

private Kafka Template<String, String> template;

@Value("${app.tpc.name}")

private String tpcName;

@Override

public void run(String... args) throws Exception {

Scanner sc=new Scanner(System.in);

String msg=null;

do {

//read the message

System.out.println("Enter message::"); msg=sc.next();

//send the message

template.send(tpcName,msg);

}while(!msg.equals("end"));

}

}

application.properties

BootKafka Proj02-Subscriber [boot]

spring.application.name=BootKafka Proj02-Subscriber

src/main/java

#topic name (user-defined)

#com.nt

place @EnableKafka in main class BootKafkaProj02SubscriberApplication.java

#com.nt.subscriber

> KafkaSubscriber.java

src/main/resources

application.properties

src/test/java

> JRE System Library [JavaSE-17]

>

Maven Dependencies

target/generated-sources/annotations

target/generated-test-sources/test-annotations

src

>

target

HELP.md

mvnw

mvnw.cmd

pom.xml

Execution order

app.tpc.name=tpc619

#Producer properties

spring.kafka.consumer.bootstrap-server-localhost:9092

spring.kafka.consumer.key-deserializer=org.apache.kafka.common.serialization. String Deserializer

spring.kafka.consumer.value-deserializer=org.apache.kafka.common.serialization.StringDeserializer

// Subscriber (KafkaListner)

package com.nt.subscriber;

import org.springframework.kafka.annotation.KafkaListener;

import org.springframework.stereotype.Component;

@Component

public class KafkaSubscriber {

@KafkaListener(topics ="${app.tpc.name}",groupid = "grp3")

public void readMessage(String msg) {

}

}

System.out.println(msg);

==========

a) Start the zoo keeper as the bootstrap server

| D:\kafka_2.13-3.6.1\bin\windows>zookeeper-server-start.bat D:\kafka_2.13-3.6.1\config\zookeeper.properties

b) start kafka server setup

D:\kafka_2.13-3.6.1\bin\windows>kafka-server-start.bat D:\kafka_2.13-3.6.1\config\server.properties

c) Run the subscriber app for multiple times with different group ids

d) Run the publisher app once to send messages

(observe these messages coming over the consoles of consumers)

Producer:

Enter message::

hello

Enter message ::

jani

2025-02-23T08:23:04.

2025-02-23T08:23:04.

hello

jani

2025-02-23T08:32:04.

Sending Object as Message in Spring apache Kafka Messaging

==========

========

========

=> In Kafka Producer App, we need to use StringSerializer for key serialization

and we need to use JsonSerializer for Object (value) Serialization

=> In Kafka Consumer App, we need to use String DeSerializer for key deserialization

and we need to use JsonDeserializer for Object (value) DeSerialization

BootKafka Proj03-WorkingObjs [boot] [devtools]

> Deployment Descriptor: BootKafkaProj03-WorkingC

>

~

JAX-WS Web Services

Starters to add :: web, apache kafka, Lombok,

dev Tools

Java Resources

=> Place @Enablekafka on the top of main class

src/main/java

〈 #com.nt

BootKafkaProj03WorkingObjsApplication.ja

>

> ServletInitializer.java

com.nt.advice

> KafkaMessageOpertionsControllerAdvice.ja

com.nt.model

> IndianFestival.java

#com.nt.publisher

> KafkaMessageSender.java

✓com.nt.rest

> KafkaMessageOperationsController.java

#com.nt.subscriber

> KafkaMessageSubscriber.java

src/main/resources

static

templates

application.properties

> src/test/java

target/generated-sources/annotations > target/generated-test-sources/test-annotations

<

Libraries

Deployed Resources

>

>

src

> target

application.properties spring.application.name=BootKafka Proj03-WorkingObjs

#topic name (user-defined)

app.tpc.name=nittpc621

#Context path of the app

server.servlet.context-path=/Boot Kafka Proj03

#Embedded server port

server.port=4041

#Producer properties

spring.kafka.producer.bootstrap-server=localhost:9092

spring.kafka.producer.key-serializer=org.apache.kafka.common.serialization. StringSerializer

spring.kafka.producer.value-serializer=org.springframework.kafka.support.serializer.JsonSerializer

#Consumer properties

spring.kafka.consumer.bootstrap-server=localhost:9092

spring.kafka.consumer.key-deserializer=org.apache.kafka.common.serialization. StringDeserializer

spring.kafka.consumer.value-deserializer=org.springframework.kafka.support.serializer.JsonDeserializer

spring.kafka.consumer.properties.spring.json.trusted.packages=com.nt.model

//Model class

package com.nt.model;

import java.io.Serializable;

import lombok.AllArgsConstructor;

import lombok.Data; import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class IndianFestival implements Serializable{

private Integer fid;

private String fname;

private String season;

private String reason;

private String dressCode;

}

//KafkaMessageSender.java

package com.nt.publisher;

import org.springframework.beans.factory.annotation.Autowired;

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.kafka.core.Kafka Template;
import org.springframework.stereotype.Component;
import com.nt.model.IndianFestival;
@Component
public class Kafka MessageSender {
@Autowired
private KafkaTemplate<String, IndianFestival> template;
@Value("${app.tpc.name}")
private String topicName;
public String sendMessage(Indian Festival festival) {
template.send(topicName, "festival Info", festival);
return "message sent";
}
}
//Subscriber
package com.nt.subscriber;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Component;
import com.nt.model.Indian Festival;
@Component
public class Kafka MessageSubscriber {
private Indian Festival festival;
@KafkaListener(topics = "${app.tpc.name}", groupId = "grp1")
public void readMessage(Indian Festival festival) {
this.festival=festival;
System.out.println("message is ::"+festival);
}
public Indian Festival getCurrentMessage() {
return festival;
}
}
//KafkaMessageOperationsController
package com.nt.rest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import
org.springframework.web.bind.annotation.GetMapping; import
org.springframework.web.bind.annotation.PostMapping; import
```

```
org.springframework.web.bind.annotation.RequestBody; import
org.springframework.web.bind.annotation.RequestMapping; import
org.springframework.web.bind.annotation.RestController;
import com.nt.model.Indian Festival; import com.nt.publisher.Kafka MessageSender;
import com.nt.subscriber.Kafka MessageSubscriber;
@RestController
@RequestMapping("/message-api")
public class Kafka MessageOperationsController {
@Autowired
private KafkaMessageSender sender;
@Autowired private KafkaMessageSubscriber subscriber;
}
//end points
@PostMapping("/send")
public ResponseEntity<String> pushMessage(@RequestBody Indian Festival festival){
}
//send the message
String msg=sender.sendMessage(festival);
//return ResponseEntity obj
return new ResponseEntity<String>(msg,HttpStatus.OK);
@GetMapping("/read")
public ResponseEntity< Indian Festival> readMessage(){
//read message
IndianFestival festival=subscriber.getCurrentMessage();
//return ResponseEntity obj
return new ResponseEntity< Indian Festival>(festival, HttpStatus.OK);
}
//RestControllerAdvice
package com.nt.advice;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;
@RestControllerAdvice
public class Kafka MessageOpertionsControllerAdvice {
@ExceptionHandler(exception = Exception.class)
public ResponseEntity<String> handleAllExceptions(Exception e){
return new ResponseEntity<String>(e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
```

}

}

POST (a)

V

http://localhost:3131/BootKafkaProj03-WorkingObjs/message-api/send

(b)

(c)

Params

Authorization

Headers (9)

Body ⚫ Scripts

Settings

none

O form-data

Ox-www-form-urlencoded

O raw

O binary

O GraphQL

JSON

<

(d)

(e)

{

"fid": 3,

12345 ~

6

7

}

"fname": "Navratri",

"season": "Autumn",

"reason": "Worship of Goddess Durga",

"dressCode": "Colorful traditional dresses"

Body

Cookies Headers (5)

Test Results

T= Raw

▷ Preview

Visualize

1

message sent

(h) --->look for this output

HTTP KafkaProj03/req1-readMessage

(f)-->type this

Send

(g)

Cookies

Beautify

200 OK ⚫ 392 ms. 176 B

e.g. Save Response

000

GET

(a)

http://localhost:3131/BootKafkaProj03-WorkingObjs/message-api/read

Params Authorization Headers (7) Body

(b)

Scripts

Settings

Query Params

Key

Value

Body

Cookies Headers (5) Test Results

{} JSON ✓

▷ Preview

Visualize

```
1 {
2
"fid": 3,
3
"fname": "Navratri",
4
```

"season": "Autumn",

5

6

"reason": "Worship of Goddess Durga",

"dressCode": "Colorful traditional dresses"

7

}

Save

V

Share

Send

(c)

Cookies

Description

000

Bulk Edit

200 OK

48 ms

289 B

e.g. Save Response

000

(Look for this output)