**Prerequisites**

* Please follow this guide to Kafka on your machine.

**2. Application Configuration**

In application.properties file, we have added following configuration.

|  |
| --- |
| application.properties |
| server.port=9000    spring.kafka.consumer.bootstrap-servers: localhost:9092  spring.kafka.consumer.group-id: group-id  spring.kafka.consumer.auto-offset-reset: earliest  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=\*    spring.kafka.producer.bootstrap-servers: localhost:9092  spring.kafka.producer.key-serializer: org.apache.kafka.common.serialization.StringSerializer  spring.kafka.producer.value-serializer: org.springframework.kafka.support.serializer.JsonSerializer |

* spring.kafka.consumer.key-deserializer specifies the deserializer class for keys.
* spring.kafka.consumer.value-deserializer specifies the deserializer class for values.
* spring.kafka.consumer.properties.spring.json.trusted.packages specifies comma-delimited list of package patterns allowed for deserialization. '\*' means deserialize all packages.
* spring.kafka.producer.key-deserializer specifies the serializer class for keys.
* spring.kafka.producer.value-deserializer specifies the serializer class for values.

**3. Model class**

We have created User class, which we will send to Kafka. Its instance will be serialized by JsonSerializer to byte array. Kafka finally stores this byte array into the given partition.

During deserialization, JsonDeserializer is used to for receiving JSON from Kafka as byte array and return User object to application.

|  |
| --- |
| User.java |
| public class User  {    private long userId;      private String firstName;      private String lastName;      public long getUserId() {      return userId;    }    public void setUserId(long userId) {      this.userId = userId;    }    public String getFirstName() {      return firstName;    }    public void setFirstName(String firstName) {      this.firstName = firstName;    }    public String getLastName() {      return lastName;    }    public void setLastName(String lastName) {      this.lastName = lastName;    }      @Override    public String toString() {      return "User [userId=" + userId + ", firstName="              + firstName + ", lastName=" + lastName + "]";    }  } |

**4. Kafka Producer**

The producer API simply consumers the user information in a HTTP POST API. It then creates a new User object and send to Kafka using [KafkaTemplate](https://docs.spring.io/spring-kafka/api/org/springframework/kafka/core/KafkaTemplate.html" \t "_blank).

|  |
| --- |
| KafkaProducerController.java |
| @PostMapping(value = "/createUser")  public void sendMessageToKafkaTopic(      @RequestParam("userId") long userId,      @RequestParam("firstName") String firstName,      @RequestParam("lastName") String lastName) {      User user = new User();    user.setUserId(userId);    user.setFirstName(firstName);    user.setLastName(lastName);      this.producerService.saveCreateUserLog(user);  } |
| KafKaProducerService.java |
| @Autowired  private KafkaTemplate<String, Object> kafkaTemplate;    public void saveCreateUserLog(User user)  {    logger.info(String.format("User created -> %s", user));    this.kafkaTemplate.send(AppConstants.TOPIC\_NAME\_USER\_LOG, user);  } |

**5. Kafka Consumer**

The consumer is implemented as @KafkaListener which gets notified everytime a new entry is added in topic.

|  |
| --- |
| KafKaConsumerService.java |
| @KafkaListener(topics = AppConstants.TOPIC\_NAME\_USER\_LOG,          groupId = AppConstants.GROUP\_ID)  public void consume(User user)  {    logger.info(String.format("User created -> %s", user));  } |

**6. Test**

Use any REST API tester and post few messages to API http://localhost:9000/kafka/createUser as below.

Message post : http://localhost:9000/kafka/createUser?userId=1&firstName=Lokesh&lastName=Gupta

Observe the console logs:

|  |
| --- |
| Console |
| 2020-05-24 23:36:47.132  INFO 2092 --- [nio-9000-exec-4]  2020-05-26 01:03:52.722  INFO 11924 --- [nio-9000-exec-6] c.h.k.demo.service.KafKaProducerService  : User created -> User [userId=1, firstName=Lokesh, lastName=Gupta]    2020-05-26 01:03:52.729  INFO 11924 --- [ntainer#1-0-C-1] c.h.k.demo.service.KafKaConsumerService  : User created -> User [userId=1, firstName=Lokesh, lastName=Gupta] |

**7. Conclusion**

In this **spring boot kafka JsonSerializer example**, we learned to use JsonSerializer to serialize and deserialize the Java objects and store in Kafka.