**Assisted Practice: 2.5 Sending and Receiving Messages with Apache Kafka**

*It is assumed you already have Apache Kafka and Zookeeper installed and running*

This section will guide you to:

* Set up Eclipse to work with Spring Boot using the STS plugin
* Create configuration classes for Kafka Producer and Consumer roles
* Create a Controller class to send messages to Kafka
* Configure the main application as a Kafka listener

**Development Environment**

* Eclipse IDE for Enterprise Java Developers v2019-03 (4.11.0)
* Apache Tomcat Server v9.0
* JRE: OpenJDK Runtime Environment 11.0.2
* Spring Boot STS 4
* All other software components are configured automatically by Spring Boot

This lab has ten subsections, namely:

* + 1. Installing the STS plugin in Eclipse
    2. Creating a Spring Boot Starter Project which is web enabled and has Apache Kafka dependencies
    3. Creating a KafkaProducerConfig class
    4. Creating a KafkaConsumerConfig class
    5. Creating MainController to send a Kafka message
    6. Configuring SpringRestApplication to listen to Kafka messages
    7. Building the project
    8. Publishing and starting the project
    9. Running the project
    10. Pushing the code to your GitHub repositories

**Step 2.5.1:** Installing the STS plugin in Eclipse

* Spring Tool Suite is already installed as an Eclipse plugin in your practice lab. (Refer FSD: Lab Guide - Phase 3 to verify the installation.)

**Step 2.5.2:** Creating a Spring Boot Starter Project which is web enabled and has Apache Kafka dependencies

* Open Eclipse
* Go the **File** menu. Choose **New->Other**
* In the **Wizards** list, select **Spring Boot->Spring Starter Project**
* In **Name** enter SpringKafka, **Type** as Maven, **Packaging** as Jar, **Group** as com.ecommerce, and **Package** as com.ecommerce
* Click on **Next**
* In the list of **Available** dependencies, scroll down to select **Web->Spring Web Starter**
* Scroll to select **Messaging->Spring for Apache Kafka** and **Messaging->Spring for Apache Kafka Streams**
* Click on **Next**
* Click on **Finish**
* This will create the project files in the Project Explorer
* In the Project Explorer, expand **SpringKafka** and double click **pom.xml**
* Add the following entries:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.1.6.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>com.ecommerce</groupId>

<artifactId>SpringKafka</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringKafka</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.apache.kafka</groupId>

<artifactId>kafka-streams</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.kafka</groupId>

<artifactId>spring-kafka</artifactId>

<version>2.1.6.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>5.1.5.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.kafka</groupId>

<artifactId>spring-kafka</artifactId>

<version>2.2.7.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**Step 2.5.3:** Creating a KafkaProducerConfig class

* In the Project Explorer, expand **SpringKafka->src->main->java->com->ecommerce**
* Right click on **ecommerce** and click on **New->Other**
* In the **Wizard** list, choose **Class** and click on **Next**
* In **Package,** entercom.ecommerce and in **Name** enter KafkaProducerConfig and click on **Finish**
* Add the following code:

**package** com.ecommerce;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.apache.kafka.clients.producer.ProducerConfig;

**import** org.apache.kafka.common.serialization.StringSerializer;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.kafka.core.DefaultKafkaProducerFactory;

**import** org.springframework.kafka.core.KafkaTemplate;

**import** org.springframework.kafka.core.ProducerFactory;

**@Configuration**

**public** **class** KafkaProducerConfig {

**@Bean**

**public** ProducerFactory<**String**, **String**> producerFactory() {

**Map**<**String**, **Object**> configProps = **new** **HashMap**<>();

configProps.put(ProducerConfig.BOOTSTRAP\_SERVERS\_CONFIG, "localhost:9092");

configProps.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class);

configProps.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class);

**return** **new** DefaultKafkaProducerFactory<>(configProps);

}

**@Bean**

**public** KafkaTemplate<**String**, **String**> kafkaTemplate() {

**return** **new** KafkaTemplate<>(producerFactory());

}

}

**Step 2.5.4:** Creating a KafkaConsumerConfig class

* In the Project Explorer, expand **SpringKafka->src->main->java->com->ecommerce**
* Right click on **ecommerce** and click on **New->Other**
* In the **Wizard** list, choose **Class** and click on **Next**
* In **Package,** entercom.ecommerce and in **Name** enter KafkaConsumerConfig and click on **Finish**
* Add the following code:

**package** com.ecommerce;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.apache.kafka.clients.consumer.ConsumerConfig;

**import** org.apache.kafka.common.serialization.StringDeserializer;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.kafka.annotation.EnableKafka;

**import** org.springframework.kafka.config.ConcurrentKafkaListenerContainerFactory;

**import** org.springframework.kafka.core.ConsumerFactory;

**import** org.springframework.kafka.core.DefaultKafkaConsumerFactory;

**@EnableKafka**

**@Configuration**

**public** **class** KafkaConsumerConfig {

**@Bean**

**public** ConsumerFactory<**String**, **String**> consumerFactory() {

**Map**<**String**, **Object**> props = **new** **HashMap**<>();

props.put(ConsumerConfig.BOOTSTRAP\_SERVERS\_CONFIG, "localhost:2181");

props.put(ConsumerConfig.GROUP\_ID\_CONFIG, "group-id");

props.put(ConsumerConfig.KEY\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class);

props.put(ConsumerConfig.VALUE\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class);

**return** **new** DefaultKafkaConsumerFactory<>(props);

}

**@Bean**

**public** ConcurrentKafkaListenerContainerFactory<**String**, **String**> kafkaListenerContainerFactory() {

ConcurrentKafkaListenerContainerFactory<**String**, **String**>

factory = **new** ConcurrentKafkaListenerContainerFactory<>();

factory.setConsumerFactory(consumerFactory());

**return** factory;

}

}

**Step 2.5.5:** Creating MainController to send a Kafka message

* In the Project Explorer, expand **SpringKafka->src->main>java>com->ecommerce->controllers**
* Right click on **controllers** and click on **New->Other**
* In the **Wizard** list, choose **Class** and click on **Next**
* In **Name,** enter **MainController** and click on **Finish**
* Add the following code:

**package** com.commerce.controllers;

**import** java.util.Calendar;

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

**import** org.springframework.kafka.core.DefaultKafkaProducerFactory;

**import** org.springframework.kafka.core.KafkaTemplate;

**import** org.springframework.kafka.core.ProducerFactory;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

**@Controller**

**public** **class** MainController {

**@Autowired**

**private** KafkaTemplate<**String**, **String**> kafkaTemplate;

**@RequestMapping(value = "/")**

**public** **String** index() {

**this**.sendMessage("This is a message sent at " + **Calendar**.getInstance().getTime());

**return** "Check Eclipse console for kafka output";

}

**private** void sendMessage(**String** msg) {

kafkaTemplate.send("ecommerce", msg);

}

}

**Step 2.5.6:** Configuring SpringRestApplication to listen to Kafka messages

* In the Project Explorer, expand **SpringKafka->src->main>java>com->ecommerce->**
* Double click on **SpringBootKafkaApplication**
* Enter the following code:

**package** com.ecommerce;

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

**import** org.springframework.boot.ApplicationArguments;

**import** org.springframework.boot.ApplicationRunner;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.kafka.annotation.KafkaListener;

**import** org.springframework.kafka.core.KafkaTemplate;

**@SpringBootApplication**

**public** **class** SpringBootKafkaApplication {

**@Autowired**

**private** KafkaTemplate<**String**, **String**> kafkaTemplate;

**public** **static** void main(**String**[] args) {

SpringApplication.run(SpringBootKafkaApplication.class, args);

}

**@KafkaListener(topics = "ecommerce", groupId = "group-id")**

**public** void listen(**String** message) {

**System**.out.println("Received Message in group - group-id: " + message);

}

}

**Step 2.5.7:** Building the project

* From the **Project** menu at the top, click on **Build**
* If any compile errors are shown, fix them as required

**Step 2.5.8:** Publishing and starting the project

* In the Project Explorer, right click on **SpringKafka->Run As->Spring Boot App**
* Check in the Eclipse Console for the message **Started SpringBootStarterApplication**

**Step 2.5.9:** Running the project

* To run the project, open a web browser and type [**http://localhost:8080/**](http://localhost:8080/products)
* To see the Kafka messages captured by the main application, check the Eclipse console

**Step 2.5.10:** Pushing the code to your GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**