TP N°3_JMS :

Implémentation d'une communication asynchrone avec JMS, Spring et ActiveMQ (Artemis) En mode Publish/Subscriber

1. Prérequis

- JDK 1.8
- Connexion internet

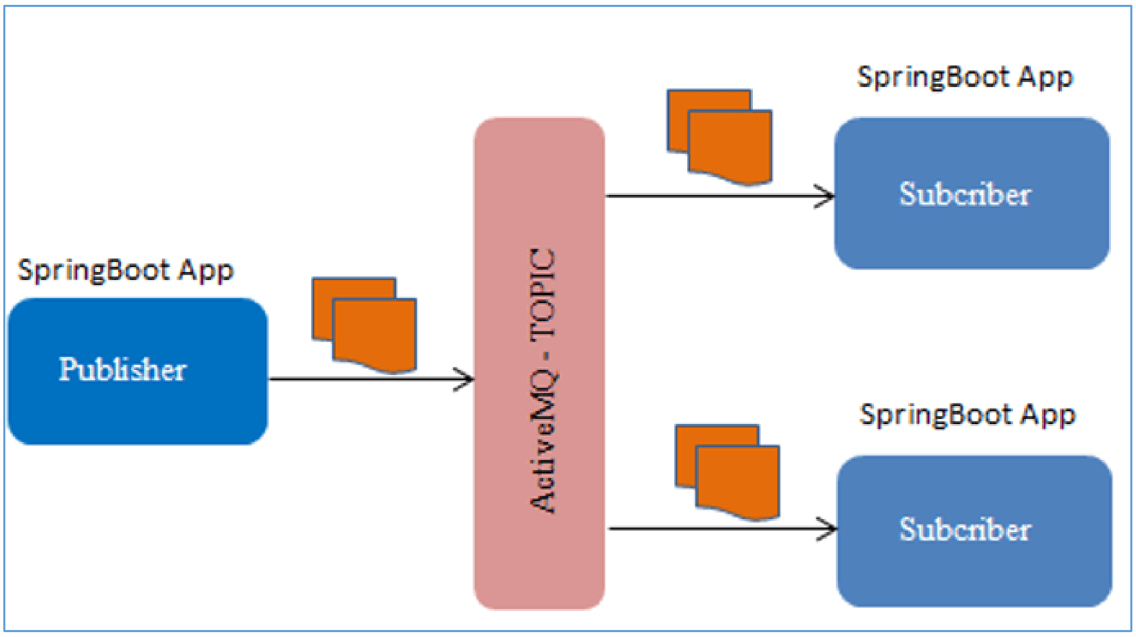
2. Objectifs

1. Développement d'un producer JMS : @EnableJms, JmsTemplate
2. Configuration et Injecton de la configuration du topic avec spring boot :

   `ActiveMQConnectionFactory, JmsTemplate`
3. Communication en mode publish/subscriber via une topic :

   `template.setPubSubDomain(true)`
4. Développement d'un consumer JMS en mode asynchrone : @JmsListener
5. Publier/Consommer une liste d'objets Java « Campany » qui contient une liste de « Product » à travers le Broker de type ActiveMQ(Artemis) externe
6. Console d'administration de Artemis (mode avancé)

3. Architecture

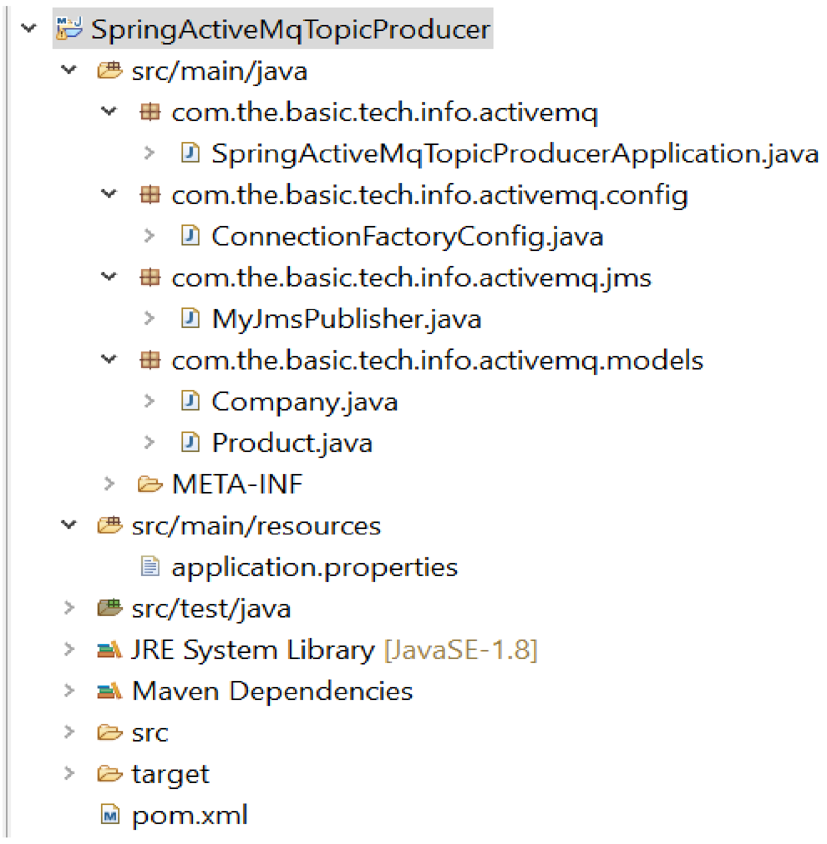


4. Installation de ActiveMQ/Artemis

1. Cf.TP2 de JMS

2. Développement d'un Publisher

- Créer le projet Maven « SpringActiveMqTopicProducer »



```
SpringActiveMqTopicProducer
  src/main/java
    com.the.basic.tech.info.activemq
      SpringActiveMqTopicProducerApplication.java
    com.the.basic.tech.info.activemq.config
      ConnectionFactoryConfig.java
    com.the.basic.tech.info.activemq.jms
      MyJmsPublisher.java
    com.the.basic.tech.info.activemq.models
      Company.java
      Product.java
    META-INF
  src/main/resources
    application.properties
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  pom.xml
```

a. Le fichier « pom.xml » :

```xml
<?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>
	<groupId>com.example.activemq</groupId>
	<artifactId>SpringActiveMqTopicProducer</artifactId>
	<version>0.0.1</version>
	<packaging>jar</packaging>
	<name>SpringActiveMqTopicProducer</name>
	<description>SpringActiveMqTopicProducer</description>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.4.4</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
		<java.version>1.8</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-activemq</artifactId>
		</dependency>
		<dependency>
			    <groupId>com.fasterxml.jackson.core</groupId>
			    <artifactId>jackson-databind</artifactId>
		</dependency>
		<dependency>
			<groupId>org.json</groupId>
			<artifactId>json</artifactId>
			<version>20210307</version>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>
<build><plugins><plugin><groupId>org.springframework.boot</groupId><artifac
tId>spring-boot-maven-plugin</artifactId></plugin>
</plugins></build></project>
```

b. Le fichier « application.properties »

```
jsa.activemq.broker.url=tcp://localhost:61616
jsa.activemq.borker.username=admin
jsa.activemq.borker.password=admin
jsa.activemq.topic=my-topic
spring.jms.pub-sub-domain=true
```

c. La classe model « Company »

```
package com.the.basic.tech.info.activemq.models;

import java.util.List;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import com.fasterxml.jackson.annotation.JsonIdentityInfo;
import com.fasterxml.jackson.annotation.ObjectIdGenerators;

//Used for Serialisation : Helpful when dealing with circular dependencies
among objects: Bidirectional Relationship.
@JsonIdentityInfo(generator=ObjectIdGenerators.IntSequenceGenerator.class,
property="@id", scope = Company.class)
public class Company {
    private String name;
    private List<Product> products;
    public Company(){
    }
    public Company(String name, List<Product> products){
      this.name = name;
      this.products = products;
    }

    // name
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    // products
    public void setProducts(List<Product> products){
      this.products = products;
    }
    public List<Product> getProducts(){
      return this.products;
    }
      /**
       *
       * Show Detail View
       */
      public String toString(){
            JSONObject jsonInfo = new JSONObject();

            try {
                  jsonInfo.put("name", this.name);
```

```
                    JSONArray productArray = new JSONArray();
                    if (this.products != null) {
                          this.products.forEach(product -> {
                                JSONObject subJson = new JSONObject();
                                try {
                                      subJson.put("name",
product.getName());
                                } catch (JSONException e) {}

                                productArray.put(subJson);
                          });
                    }
                    jsonInfo.put("products", productArray);
              } catch (JSONException e1) {}
              return jsonInfo.toString();
        }
}
```

a. La classe model « Product »

```
package com.the.basic.tech.info.activemq.models;

import com.fasterxml.jackson.annotation.JsonIdentityInfo;
import com.fasterxml.jackson.annotation.ObjectIdGenerators;

@JsonIdentityInfo(generator=ObjectIdGenerators.IntSequenceGenerator.class,pr
operty="@id", scope = Product.class)
public class Product {
    private String name;

    private Company company;

    public Product(){
    }
    public Product(String name){
       this.name = name;
    }
    public Product(String name, Company company){
       this.name = name;
       this.company = company;
    }
    // name
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    // products
    public void setCompany(Company company){
       this.company = company;
    }
    public Company getCompany(){
       return this.company;
    }
}
```

b. Classe de Configuration de la ConnectionFactory, topic…

```
package com.the.basic.tech.info.activemq.config;

import javax.jms.ConnectionFactory;

import org.apache.activemq.ActiveMQConnectionFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.jms.annotation.EnableJms;
import org.springframework.jms.core.JmsTemplate;
import
org.springframework.jms.support.converter.MappingJackson2MessageConverter;
import org.springframework.jms.support.converter.MessageConverter;
import org.springframework.jms.support.converter.MessageType;

//Enable JMS listener annotated endpoints that are created under
the cover by a JmsListenerContainerFactory.
//To be used on @Configuration classes
@Configuration
@EnableJms
public class ConnectionFactoryConfig {
      @Value("${jsa.activemq.broker.url}")
      String brokerUrl;

      @Value("${jsa.activemq.borker.username}")
      String userName;

      @Value("${jsa.activemq.borker.password}")
      String password;

      /*
       * Initial ConnectionFactory
       */
    @Bean
    public ConnectionFactory connectionFactory(){
        ActiveMQConnectionFactory connectionFactory = new
ActiveMQConnectionFactory();
        connectionFactory.setBrokerURL(brokerUrl);
        connectionFactory.setUserName(userName);
        connectionFactory.setPassword(password);
        return connectionFactory;
    }

      @Bean // Serialize message content to json using TextMessage
      public MessageConverter jacksonJmsMessageConverter() {
          MappingJackson2MessageConverter converter = new
MappingJackson2MessageConverter();
          converter.setTargetType(MessageType.TEXT);
          converter.setTypeIdPropertyName("_type");
          return converter;
      }
    /*
     * Used for sending Messages.
     */

     @Bean
     public JmsTemplate jmsTemplate(){

       JmsTemplate template = new JmsTemplate();
       template.setConnectionFactory(connectionFactory());
       template.setMessageConverter(jacksonJmsMessageConverter());

//Configure the destination accessor with knowledge of the JMS
domain used.Default is Point-to-Point (Queues).
//Parameters:pubSubDomain "true" for the Publish/Subscribe domain
(Topics),"false" for the Point-to-Point domain (Queues)
        template.setPubSubDomain(true);

      return template;
      }
}
```

c. Développement du Publisher « MyJmsPublisher »

```
package com.the.basic.tech.info.activemq.jms;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.jms.core.JmsTemplate;
import org.springframework.stereotype.Component;

import com.the.basic.tech.info.activemq.models.Company;

@Component
public class MyJmsPublisher {
      private static final Logger logger =
LoggerFactory.getLogger(MyJmsPublisher.class);
//Default settings for JMS Sessions are "not transacted" and "auto-
acknowledge".As defined by the Java EE specification,

      @Autowired
      JmsTemplate jmsTemplate;

      @Value("${jsa.activemq.topic}")
      String topic;

     public void send(Company apple){
            jmsTemplate.convertAndSend(topic, apple);
            logger.info("Message : {} published to topic: {}
successfully.", apple.toString(), topic);
      }
}
```

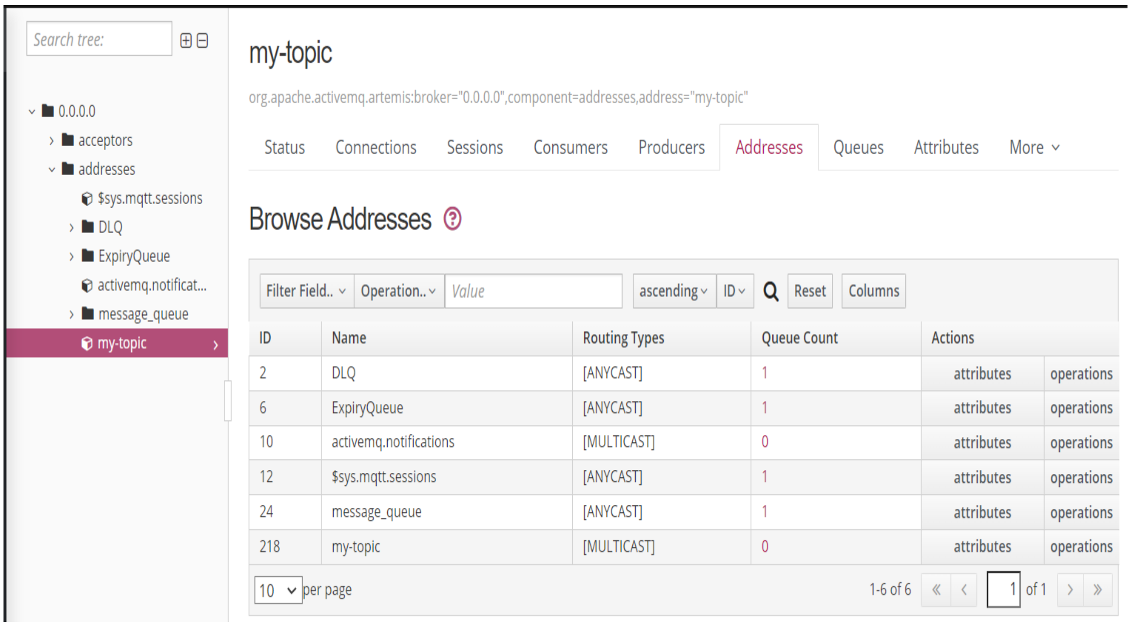
d. Classe principale « SpringActiveMqTopicProducerApplication »

```
package com.the.basic.tech.info.activemq;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

import com.the.basic.tech.info.activemq.jms.MyJmsPublisher;
import com.the.basic.tech.info.activemq.models.Company;
import com.the.basic.tech.info.activemq.models.Product;

@SpringBootApplication
public class SpringActiveMqTopicProducerApplication implements
CommandLineRunner {

      @Autowired
      MyJmsPublisher publisher;

      public static void main(String[] args) {

      SpringApplication.run(SpringActiveMqTopicProducerApplication.
class, args);
      }

      @Override
      public void run(String... args) throws Exception {
            /*
             * Apple company & products
             */
            // initial company and products
            Product iphone7 = new Product("Iphone X");
            Product iPadPro = new Product("IPadPro");

            List<Product> appleProducts = new
ArrayList<Product>(Arrays.asList(iphone7, iPadPro));

            Company apple = new Company("Apple", appleProducts);

            // send message to ActiveMQ
            publisher.send(apple);

            /*
             * Samsung company and products
             */
            Product galaxySx = new Product("Galaxy SXX");
            Product gearSy = new Product("Gear YYY");
            List<Product> samsungProducts = new
ArrayList<Product>(Arrays.asList(galaxySx, gearSy));

            Company samsung = new Company("Samsung",
samsungProducts);

            /*
             * send message to ActiveMQ
             */
            publisher.send(samsung);
      }
}
```

➔ Exécuter le Publisher : SpringActiveMqTopicProducerApplication (x3 par exemple)

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.4)

Starting SpringActiveMqTopicProducerApplication using Java
1.8.0_92

MyJmsPublisher --> jmsTemplate.convertAndSend(topic,
campany); Message :
{"name":"Apple","products":[{"name":"Iphone
X"},{"name":"IPadPro"}]} published to topic: my-topic
successfully.

MyJmsPublisher --> jmsTemplate.convertAndSend(topic,
campany); Message :
{"name":"Samsung","products":[{"name":"Galaxy SXX"}
,{"name":"Gear YYY"}]} published to topic: my-topic
successfully.
```

➔ Vérifier au niveau de la console de Artemis

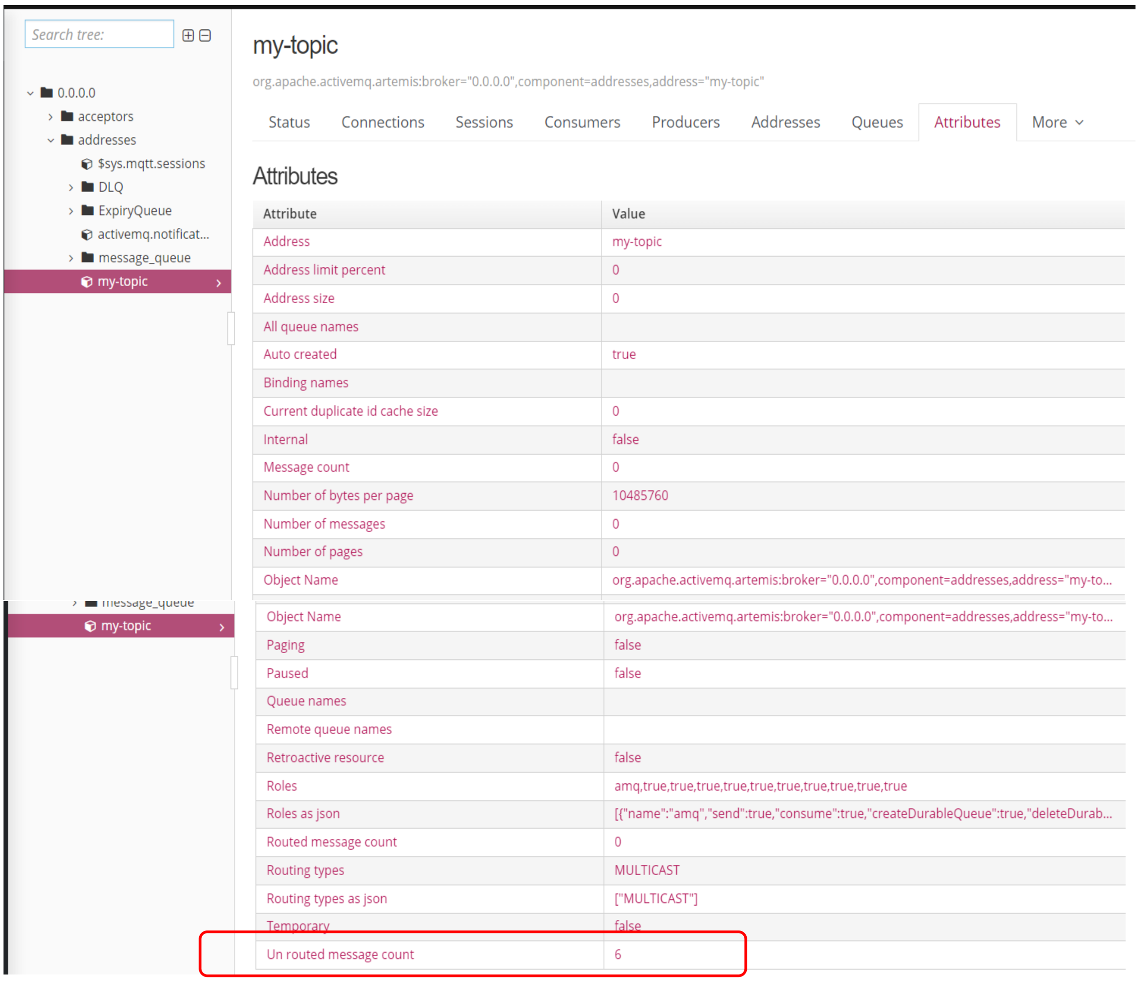


my-topic

org.apache.activemq.artemis:broker="0.0.0.0",component=addresses,address="my-topic"

Browse Addresses

ID	Name	Routing Types	Queue Count	Actions	
2	DLQ	[ANYCAST]	1	attributes	operations
6	ExpiryQueue	[ANYCAST]	1	attributes	operations
10	activemq.notifications	[MULTICAST]	0	attributes	operations
12	$sys.mqtt.sessions	[ANYCAST]	1	attributes	operations
24	message_queue	[ANYCAST]	1	attributes	operations
218	my-topic	[MULTICAST]	0	attributes	operations



my-topic

org.apache.activemq.artemis:broker="0.0.0.0",component=addresses,address="my-topic"

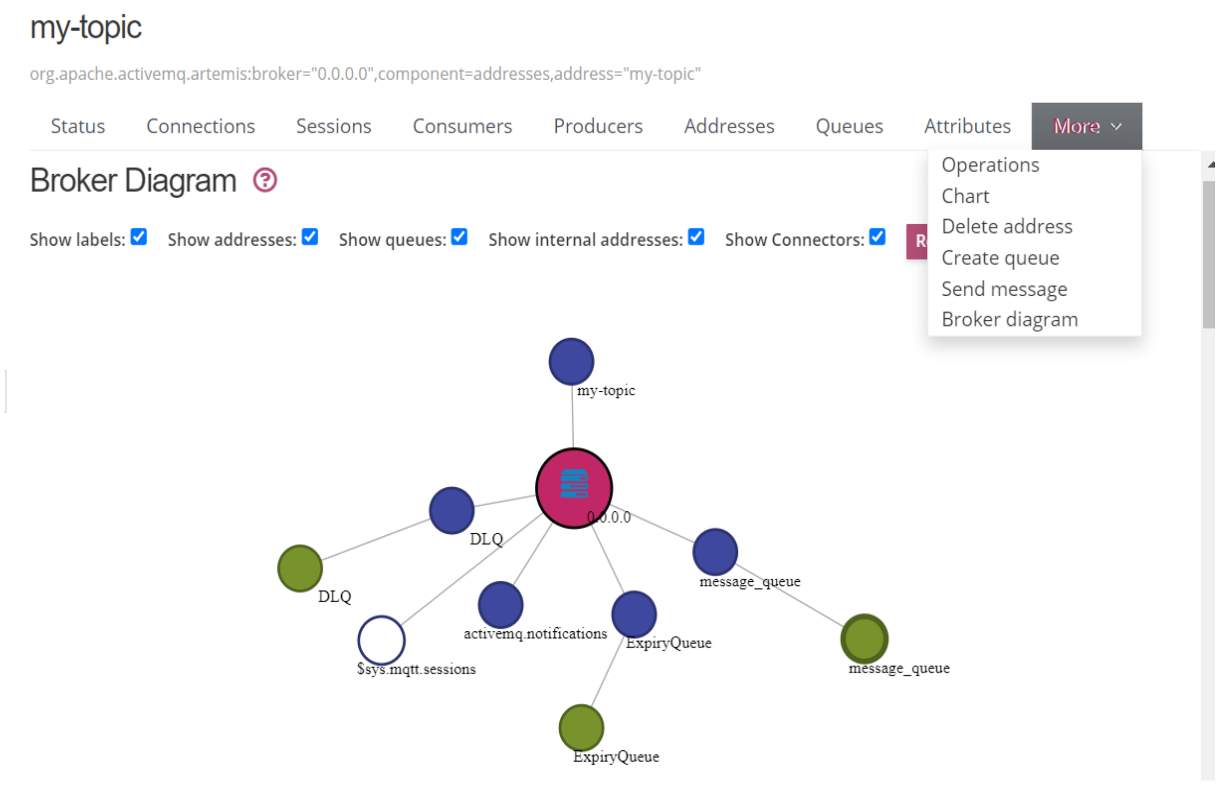
Attributes

Attribute	Value
Address	my-topic
Address limit percent	0
Address size	0
All queue names	
Auto created	true
Binding names	
Current duplicate id cache size	0
Internal	false
Message count	0
Number of bytes per page	10485760
Number of messages	0
Number of pages	0
Object Name	org.apache.activemq.artemis:broker="0.0.0.0",component=addresses,address="my-to...
Object Name	org.apache.activemq.artemis:broker="0.0.0.0",component=addresses,address="my-to...
Paging	false
Paused	false
Queue names	
Remote queue names	
Retroactive resource	false
Roles	amq,true,true,true,true,true,true,true,true,true,true
Roles as json	[{"name":"amq","send":true,"consume":true,"createDurableQueue":true,"deleteDurab...
Routed message count	0
Routing types	MULTICAST
Routing types as json	["MULTICAST"]
Temporary	false
Un routed message count	6

Remarquer qu'il y'a 6 messages qui ne sont pas encore routés, car le consumer n'est pas encore démarré. De ce fait, le topic va persister ces messages jusqu'à leur consommation par le Consumer une fois ce dernier est démarré.

Remarque : le champ « Adresse » dans Artemis peut représenter un topic.

➔ Représentation graphique du broker :


my-topic

org.apache.activemq.artemis:broker="0.0.0.0",component=addresses,address="my-topic"

Broker Diagram

Show labels: ☑ Show addresses: ☑ Show queues: ☑ Show internal addresses: ☑ Show Connectors: ☑

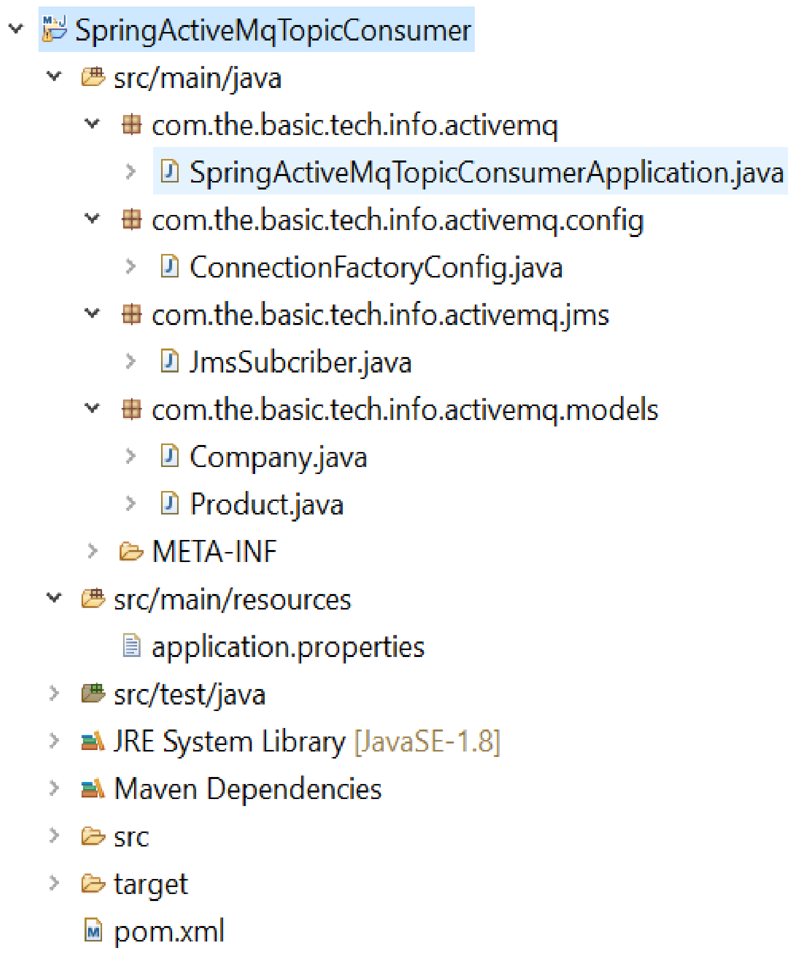
Cette page est une représentation graphique de la topologie du cluster Artemis. Il affiche tous les brokers du cluster ainsi que toutes les topics et files d'attente sur le broker auquel la console est connectée.

Il est possible de visualiser les attributs des topics, des files d'attente et du broker connecté en faisant un clic gauche sur chaque nœud.

Les Topics ont une couleur bleu, et les Queues ont une couleur verte.

3. Développement du Subscriber (Consumer de la topic en mode Asynchrone)

a. Créer le projet Maven « SpringActiveMqTopicConsumer »



```
SpringActiveMqTopicConsumer
  src/main/java
    com.the.basic.tech.info.activemq
      SpringActiveMqTopicConsumerApplication.java
    com.the.basic.tech.info.activemq.config
      ConnectionFactoryConfig.java
    com.the.basic.tech.info.activemq.jms
      JmsSubcriber.java
    com.the.basic.tech.info.activemq.models
      Company.java
      Product.java
    META-INF
  src/main/resources
    application.properties
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  pom.xml
```

a. Le fichier « pom.xml » :

```xml
<?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>

	<groupId>com.example.activemq</groupId>
	<artifactId>SpringActiveMqTopicConsumer</artifactId>
	<version>0.0.1</version>
	<packaging>jar</packaging>

	<name>SpringActiveMqTopicConsumer</name>
	<description>SpringActiveMqTopicConsumer</description>

	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.4.4</version>
```

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

      <properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
            <java.version>1.8</java.version>
      </properties>
      <dependencies>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter</artifactId>
            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-
activemq</artifactId>
            </dependency>
            <dependency>
                  <groupId>com.fasterxml.jackson.core</groupId>
                  <artifactId>jackson-databind</artifactId>
            </dependency>
            <dependency>
                  <groupId>org.json</groupId>
                  <artifactId>json</artifactId>
                  <version>20210307</version>
            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-test</artifactId>
                  <scope>test</scope>
            </dependency>
            <dependency>
                  <groupId>junit</groupId>
                  <artifactId>junit</artifactId>
                  <scope>test</scope>
            </dependency>
      </dependencies>
<build><plugins><plugin><groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId></plugin></plugins>
</build></project>
```

b. Le fichier « application.properties » est le même que dans le Publisher

```
jsa.activemq.broker.url=tcp://localhost:61616
jsa.activemq.borker.username=admin
jsa.activemq.borker.password=admin
jsa.activemq.topic=my-topic
spring.jms.pub-sub-domain=true
```

c. La classe « Campany » et « Product » sont les mêmes que dans le Publisher

d. La Classe de Configuration « ConnectionFactoryConfig » est la même que dans le Publisher

e. Classe « JmsSubcriber »

```
package com.the.basic.tech.info.activemq.jms;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.jms.annotation.JmsListener;
import org.springframework.stereotype.Component;

import com.the.basic.tech.info.activemq.models.Company;

@Component
public class JmsSubcriber {
	private static final Logger logger =
LoggerFactory.getLogger(JmsSubcriber.class);

@JmsListener(destination = "${jsa.activemq.topic}")
public void receive(Company msg){
	logger.info("***  JmsSubcriber Recieved Message:  {}",
msg.toString());
	}
}
```

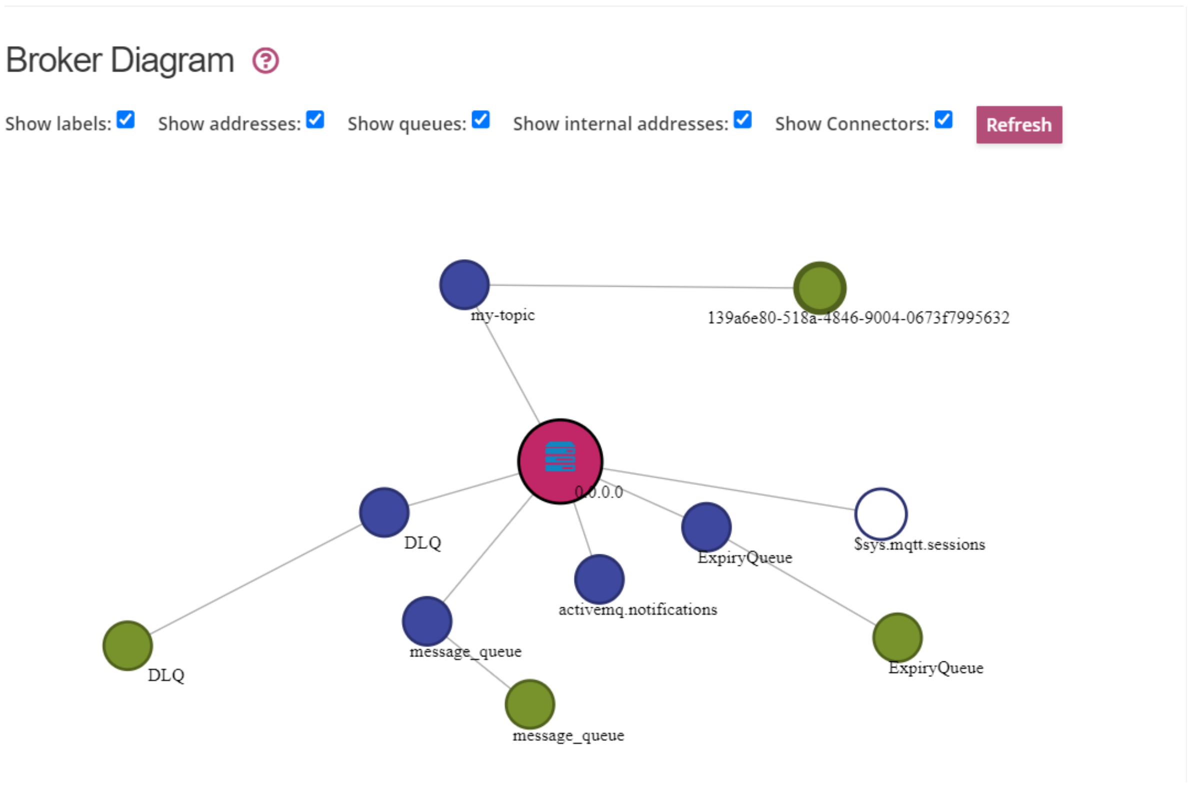
e. Classe principale « SpringActiveMqTopicConsumerApplication»

```
package com.the.basic.tech.info.activemq;

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

@SpringBootApplication
public class SpringActiveMqTopicConsumerApplication {

public static void main(String[] args) {
SpringApplication.run(SpringActiveMqTopicConsumerApplication.class,
args);
	}
}
```

➔ Réexécuter le Producer

➔ Exécuter le Subscriber :

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.4)

Starting SpringActiveMqTopicConsumerApplication using
Java 1.8.0_92
INFO 34732 --- [ntContainer#0-1]
c.t.b.t.info.activemq.jms.JmsSubcriber    : ***
JmsSubcriber Recieved Message:  Company [name=Apple,
products=[Product [name=Iphone X, company=null],
Product [name=IPadPro, company=null]]]
INFO 34732 --- [ntContainer#0-1]
c.t.b.t.info.activemq.jms.JmsSubcriber    : ***
JmsSubcriber Recieved Message:  Company [name=Samsung,
products=[Product [name=Galaxy SXX, company=null],
Product [name=Gear YYY, company=null]]]
```

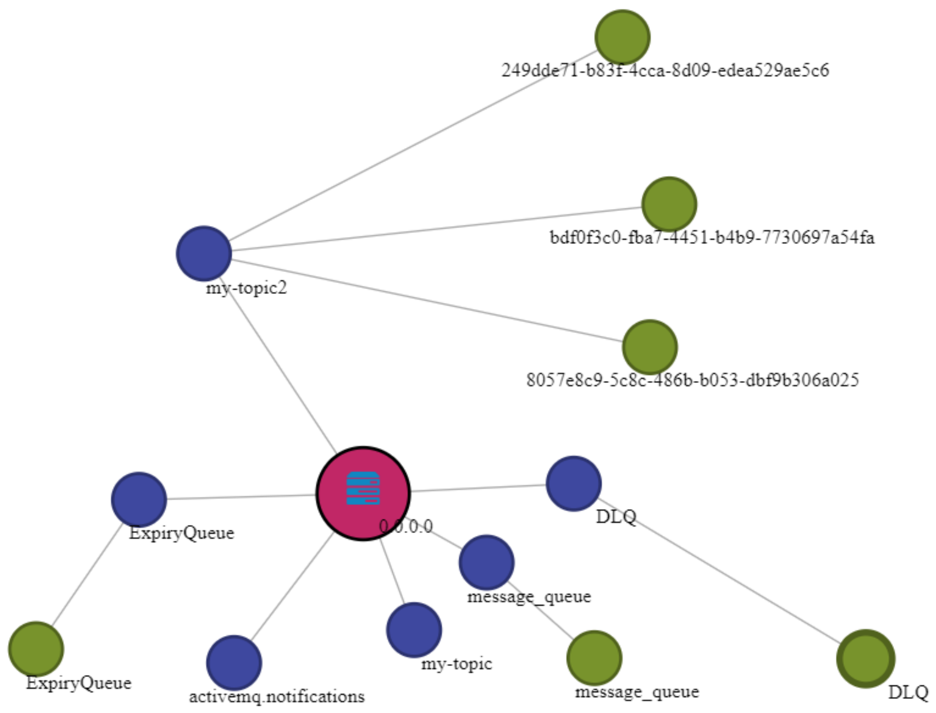


Broker Diagram

Show labels: ✓ Show addresses: ✓ Show queues: ✓ Show internal addresses: ✓ Show Connectors: ✓ Refresh

Suite à l’exécution du Subscriber, une Queue a été crée et été attachée à la Topic correspondante

Si on exécute le Scbscriber 3 fois par exemple, on obtient le diagramme ci-après :


249dde71-b83f-4cca-8d09-edea529ae5c6
bdf0f3c0-fba7-4451-b4b9-7730697a54fa
my-topic2
8057e8c9-5c8c-486b-b053-dbf9b306a025
ExpiryQueue
0.0.0.0
DLQ
message_queue
ExpiryQueue
activemq.notifications
my-topic
message_queue
DLQ

Si on exécute le Publisher à nouveau, on remarque au niveau de la console des 3 Subscribers qu'ils consomment tous le message Pushé par le publisher.