Spring Boot JMS ActiveMQ Producer and Consumer Example

By **Rakesh** -September 13, 2020

In this post we will see about Spring Boot JMS ActiveMQ Producer and Consumer Example from scratch.

Table of Contents

Points we are going to learn in this tutorial.

How to install ActiveMQ and login into ActiveMQ console.

How to create a producer endpoint to send the messages.

How to create a consumer (Using @JmsListener and controller class) to receive the messages.

Download and Install ActiveMQ.

Click on below link to download ActiveMQ.

http://activemq.apache.org/components/classic/download/

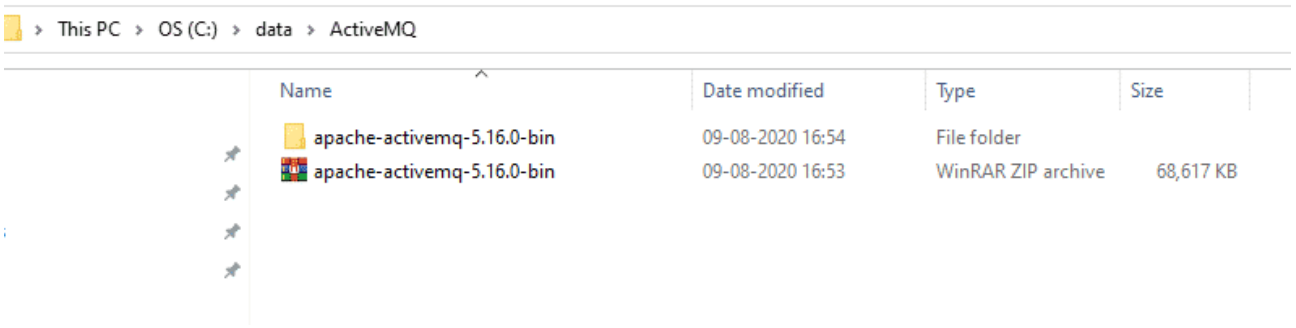
Download the zip file and extarct it.

ActiveMQ 5.16.0 (Jul 1, 2020)

Documentation

Windows	apache-activemq-5.16.0-bin.zip	SHA512	GPG Signature
Unix/Linux/Cygwin	apache-activemq-5.16.0-bin.tar.gz	SHA512	GPG Signature
Source Code Distribution:	activemq-parent-5.16.0-source-release.zip	SHA512	GPG Signature

The keys file for verifying the release can be obtained here

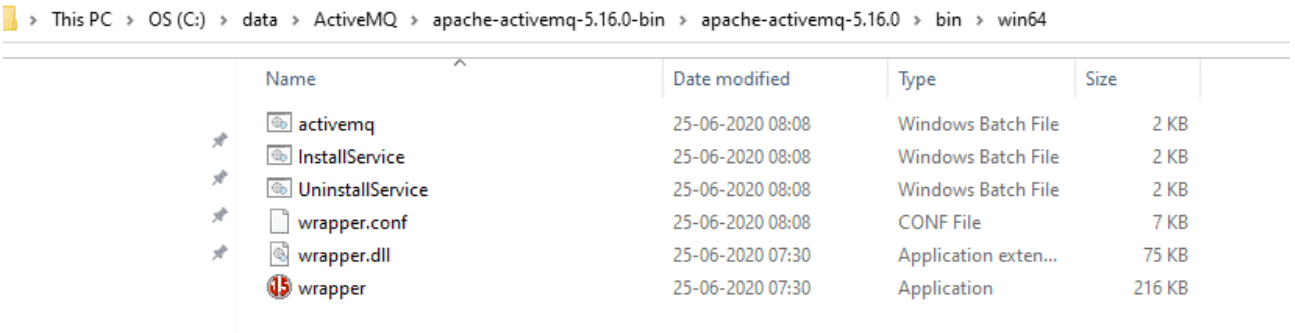


This PC › OS (C:) › data › ActiveMQ

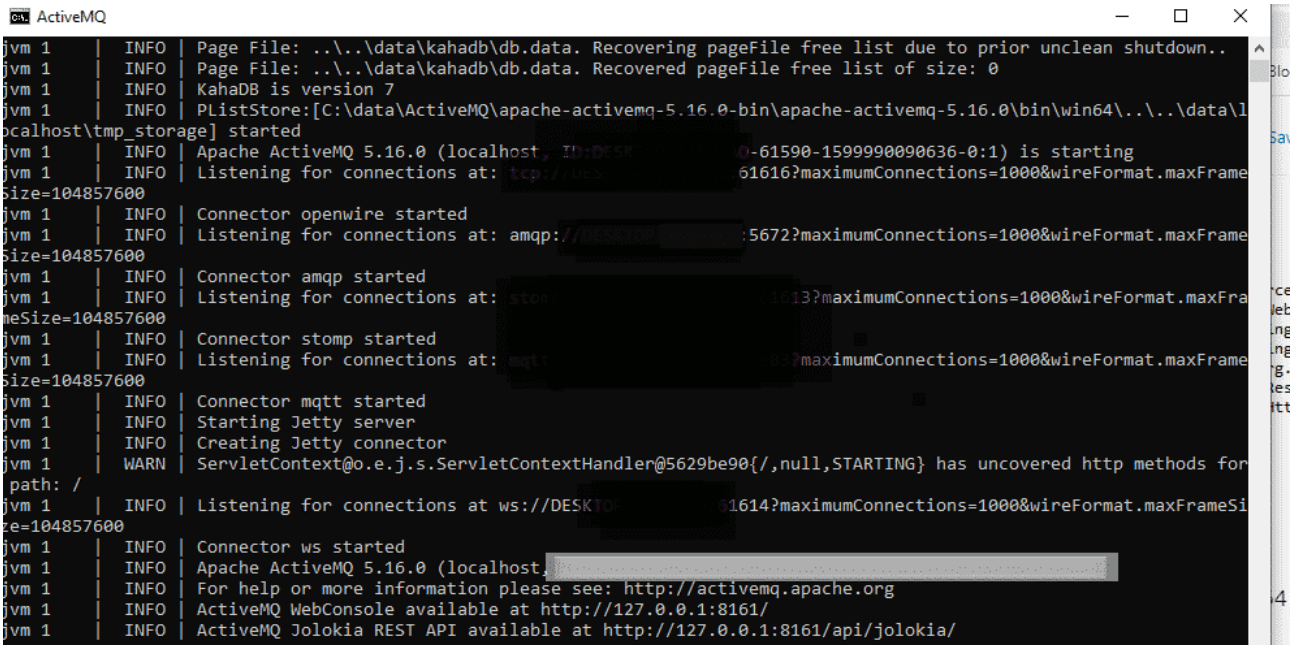
Name	Date modified	Type	Size
apache-activemq-5.16.0-bin	09-08-2020 16:54	File folder	
apache-activemq-5.16.0-bin	09-08-2020 16:53	WinRAR ZIP archive	68,617 KB

Go to win64(or win32 depends on your machine) folder. For example.

C:\data\ActiveMQ\apache-activemq-5.16.0-bin\apache-activemq-5.16.0\bin\win64



Double click on activemq, our ActiveMQ should get started and we should able to see below screen.

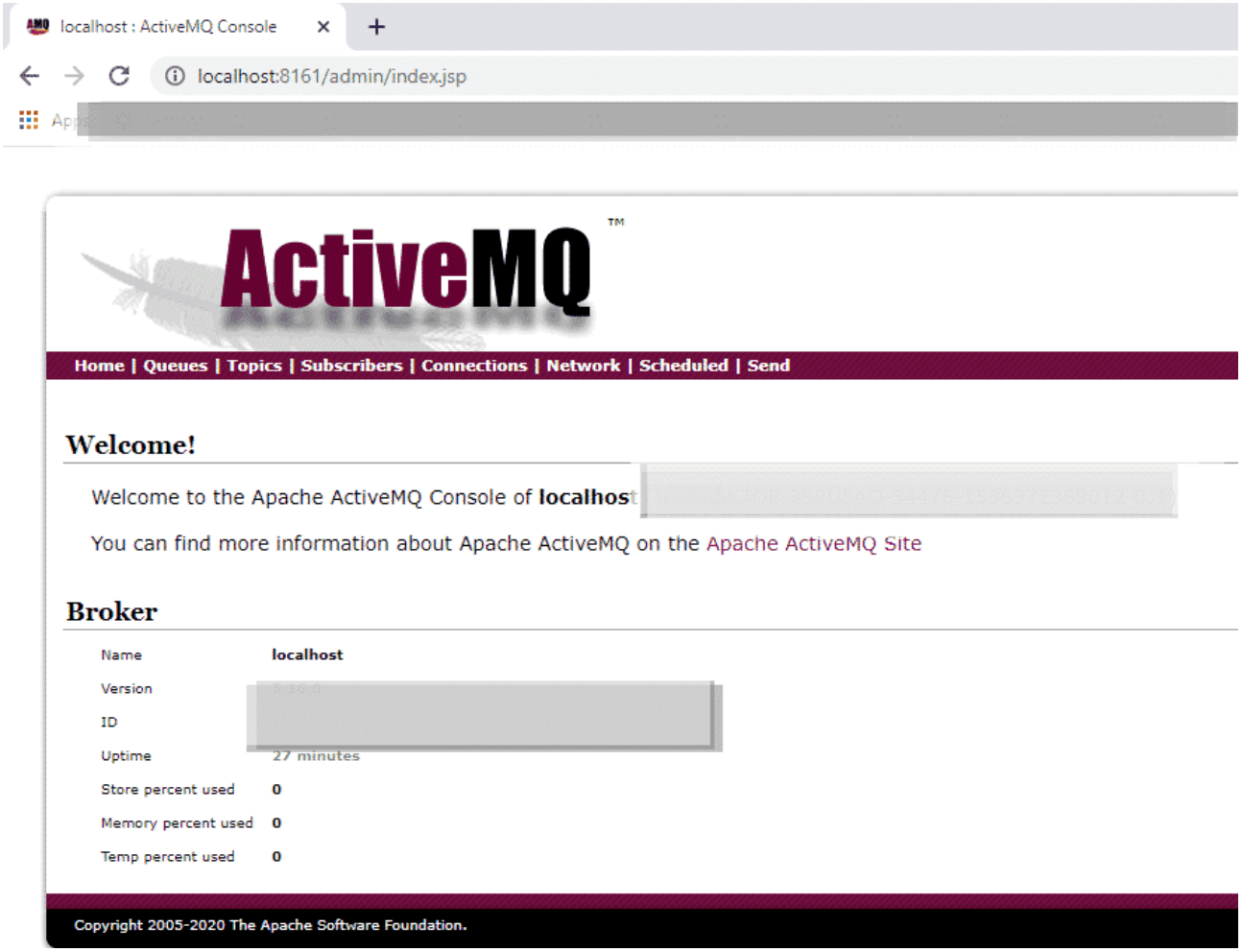


Let’s login to ActiveMQ Console using below URL.

http://localhost:8161/admin/

It will ask for username and password. Default username – admin and password – admin.

Once we provide username and password we should able to see below ActiveMQ console.



Spring Boot JMS ActiveMQ Producer and Consumer Example – Step by step tutorial from scratch.

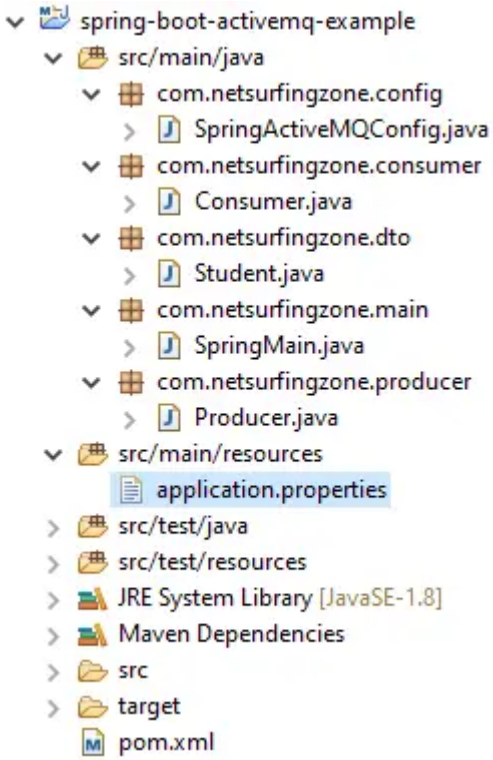
Open eclipse and create a maven project, Don't forget to check to 'create a simple project (skip)' click on next. Fill all details(GroupId – spring-boot-activemq-example , ArtifactId – spring-boot-activemq-example , and name – spring-boot-activemq-example) and click on finish. Keep packaging as the jar.

maven dependency

```
<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>spring-boot-activemq-example</groupId>
      <artifactId>spring-boot-activemq-example</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <name>spring-boot-activemq-example</name>
      <description>spring-boot-activemq-example</description>

      <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>2.0.2.RELEASE</version>
      </parent>
      <dependencies>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-web</artifactId>
            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-activemq</artifactId>
            </dependency>
      </dependencies>


</project>
```

Directory structure



application.properties

```
server.port = 9091
```

```
activemq.broker.url=tcp://localhost:61616
```

Spring Boot ActiveMQ configuration.

SpringActiveMQConfig.java

```
package com.netsurfingzone.config;

import javax.jms.Queue;

import org.apache.activemq.ActiveMQConnectionFactory;
import org.apache.activemq.command.ActiveMQQueue;
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;
```

```
@Configuration
@EnableJms
public class SpringActiveMQConfig {

	@Value("${activemq.broker.url}")
	private String brokerUrl;

	@Bean
	public Queue queue() {
		return new ActiveMQQueue("netsurfingzone-queue");
	}

	@Bean
	public ActiveMQConnectionFactory activeMQConnectionFactory() {
		ActiveMQConnectionFactory activeMQConnectionFactory = new
ActiveMQConnectionFactory();
		activeMQConnectionFactory.setBrokerURL(brokerUrl);
		return activeMQConnectionFactory;
	}

	@Bean
	public JmsTemplate jmsTemplate() {
		return new JmsTemplate(activeMQConnectionFactory());
	}

}
```

Define DTO Student.java

```
package com.netsurfingzone.dto;

import java.io.Serializable;

public class Student implements Serializable {
	private static final long serialVersionUID = 1L;
	private String studentId;
	private String name;
	private String rollNumber;

	public String getStudentId() {
		return studentId;
	}
```

```
	public void setStudentId(String studentId) {
		this.studentId = studentId;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getRollNumber() {
		return rollNumber;
	}

	public void setRollNumber(String rollNumber) {
		this.rollNumber = rollNumber;
	}

}
```

Create producer class.

Producer.java – This contoller class will be used to send message to activemq queue(netsurfingzone-queue).

```
package com.netsurfingzone.producer;

import javax.jms.Queue;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jms.core.JmsTemplate;
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.fasterxml.jackson.databind.ObjectMapper;
import com.netsurfingzone.dto.Student;

@RestController
```

```
@RequestMapping("/produce")
public class Producer {

	@Autowired
	private JmsTemplate jmsTemplate;

	@Autowired
	private Queue queue;

	@PostMapping("/message")
	public Student sendMessage(@RequestBody Student student) {

		try {
			ObjectMapper mapper = new ObjectMapper();
			String studentAsJson =
mapper.writeValueAsString(student);

			jmsTemplate.convertAndSend(queue, studentAsJson);
		} catch (Exception e) {
			e.printStackTrace();
		}
		return student;
	}
}
```

Create Consumer class.

Consumer.java – This consumer class will be used to receive messages from the ActiveMQ queue. We are using @JmsListener annotation that will be used to read the message from the destination. For example in our case destination would be netsurfingzone-queue. See more details about @JmsListener here.

```
package com.netsurfingzone.consumer;

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

@Component
public class Consumer {
```

```
	private static final Logger logger =
LoggerFactory.getLogger(Consumer.class);

	@JmsListener(destination = "netsurfingzone-queue")
	public void consumeMessage(String message) {
		logger.info("Message received from activemq
queue---"+message);
	}
}
```

The above consumer will read message and print that message to console using logger.

Define SpringMain class.

```
package com.netsurfingzone.main;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan(basePackages = "com.netsurfingzone.*")
public class SpringMain {
	public static void main(String[] args) {
		SpringApplication.run(SpringMain.class, args);
	}

}
```

Deploy the application.

```
SpringMain.java
1  package com.netsurfingzone.main;
2
3  import org.springframework.boot.SpringApplication;
6
7  @SpringBootApplication
8  @ComponentScan(basePackages = "com.netsurfingzone.*")
9  public class SpringMain {
10     public static void main(String[] args) {
11         SpringApplication.run(SpringMain.class, args);
12     }
13
14 }
15
```

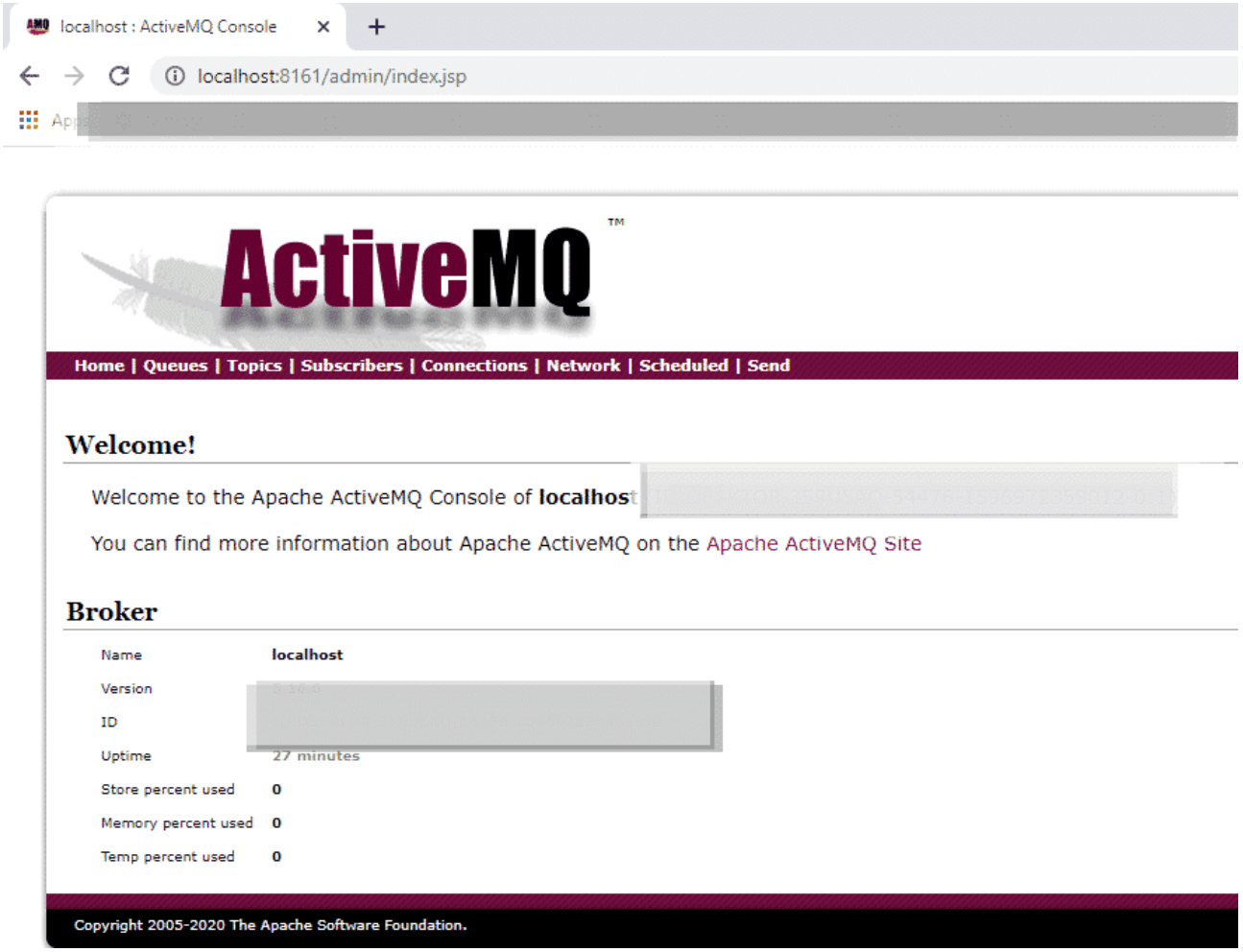
```
Markers  Properties  Servers  Data Source Explorer  Snippets  Problems  Progress  Search  Debug  Console

pringMain (28) [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (13-Sep-2020, 3:49:37 PM)
rvletRegistrationBean  : Servlet dispatcherServlet mapped to [/]
lterRegistrationBean   : Mapping filter: 'characterEncodingFilter' to: [/*]
lterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
lterRegistrationBean   : Mapping filter: 'httpPutFormContentFilter' to: [/*]
lterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
mpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.Resourc
tMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerAp
tMappingHandlerMapping : Mapped "{[/produce/message],methods=[POST]}" onto public com.netsurfingzone.dto.Student com.netsurfingzone.pro
tMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.l
tMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.springfr
mpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHtt
mpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestH
onMBeanExporter        : Registering beans for JMX exposure on startup
ultLifecycleProcessor  : Starting beans in phase 2147483647
omcat.TomcatWebServer  : Tomcat started on port(s): 9091 (http) with context path ''
.main.SpringMain       : Started SpringMain in 3.838 seconds (JVM running for 5.664)
```

Start ActiveMQ – Go to till C:\data\ActiveMQ\apache-activemq-5.16.0-bin\apache-activemq-5.16.0\bin\win64 directory and click on activemq.

```
ActiveMQ
jvm 1    |  INFO | Page File: ..\..\data\kahadb\db.data. Recovering pageFile free list due to prior unclean shutdown..
jvm 1    |  INFO | Page File: ..\..\data\kahadb\db.data. Recovered pageFile free list of size: 0
jvm 1    |  INFO | KahaDB is version 7
jvm 1    |  INFO | PListStore:[C:\data\ActiveMQ\apache-activemq-5.16.0-bin\apache-activemq-5.16.0\bin\win64\..\..\data\l
ocalhost\tmp_storage] started
jvm 1    |  INFO | Apache ActiveMQ 5.16.0 (localhost, ID:DESK[redacted]0-61590-1599990090636-0:1) is starting
jvm 1    |  INFO | Listening for connections at: tcp:[redacted]61616?maximumConnections=1000&wireFormat.maxFrame
Size=104857600
jvm 1    |  INFO | Connector openwire started
jvm 1    |  INFO | Listening for connections at: amqp:[redacted]5672?maximumConnections=1000&wireFormat.maxFrame
Size=104857600
jvm 1    |  INFO | Connector amqp started
jvm 1    |  INFO | Listening for connections at: [redacted]13?maximumConnections=1000&wireFormat.maxFra
meSize=104857600
jvm 1    |  INFO | Connector stomp started
jvm 1    |  INFO | Listening for connections at: [redacted]?maximumConnections=1000&wireFormat.maxFrame
Size=104857600
jvm 1    |  INFO | Connector mqtt started
jvm 1    |  INFO | Starting Jetty server
jvm 1    |  INFO | Creating Jetty connector
jvm 1    |  WARN | ServletContext@o.e.j.s.ServletContextHandler@5629be90{/,null,STARTING} has uncovered http methods for
 path: /
jvm 1    |  INFO | Listening for connections at ws://DESKTO[redacted]61614?maximumConnections=1000&wireFormat.maxFrameSi
ze=104857600
jvm 1    |  INFO | Connector ws started
jvm 1    |  INFO | Apache ActiveMQ 5.16.0 (localhost, [redacted]
jvm 1    |  INFO | For help or more information please see: http://activemq.apache.org
jvm 1    |  INFO | ActiveMQ WebConsole available at http://127.0.0.1:8161/
jvm 1    |  INFO | ActiveMQ Jolokia REST API available at http://127.0.0.1:8161/api/jolokia/
```

Login to ActiveMQ Console using below URL. Username and password is admin & admin.

http://localhost:8161/admin/



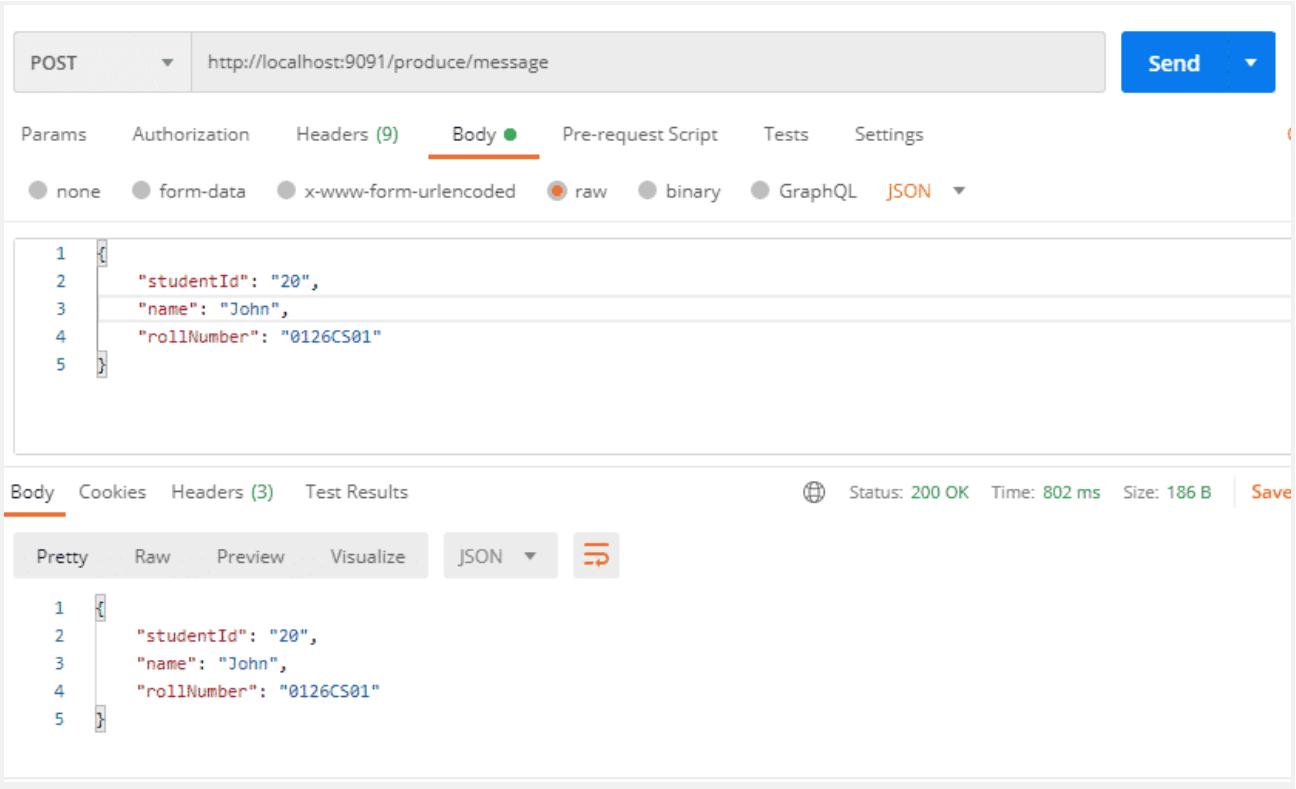
Open postman and use below URL to send the message.

http://localhost:9091/produce/message

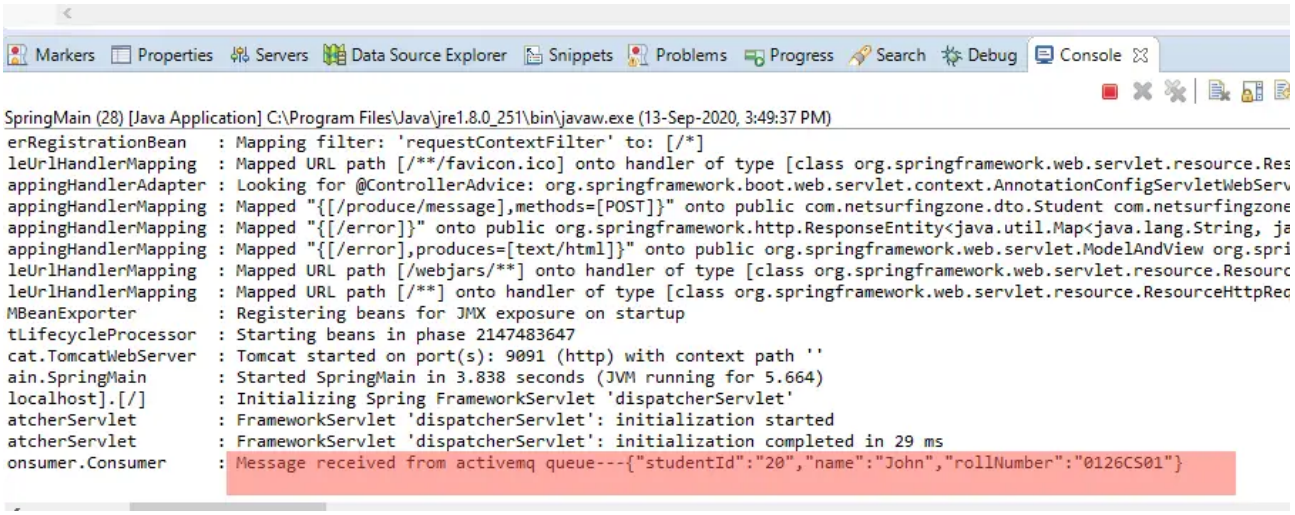
Request Data.

```
{
    "studentId": "20",
    "name": "rakesh",
    "rollNumber": "0126CS01"
```

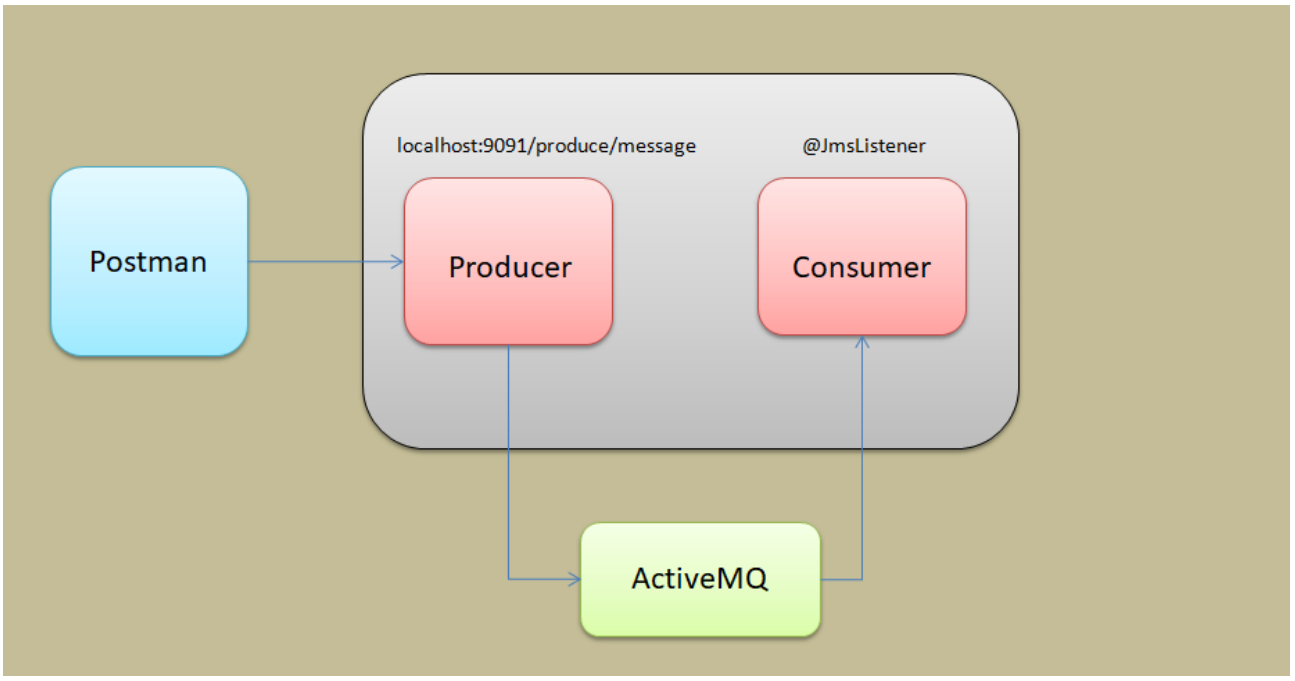
```
}
```



Our consumer should able to read message. Let’s check the console.



Flow diagram of above example.



Spring Boot ActiveMQ Consumer example – Defining Consumer as Rest End point.

We have seen Spring boot ActiveMQ producer and consumer example. We have defined listener which is reading message from queue.

In this section, we will see how to define a consumer a controller class rather than a listener, so that we can read the message even later(for example after some time). We will see our message using the ActiveMQ console(since there is no listener, ,messages would be there in Queue).

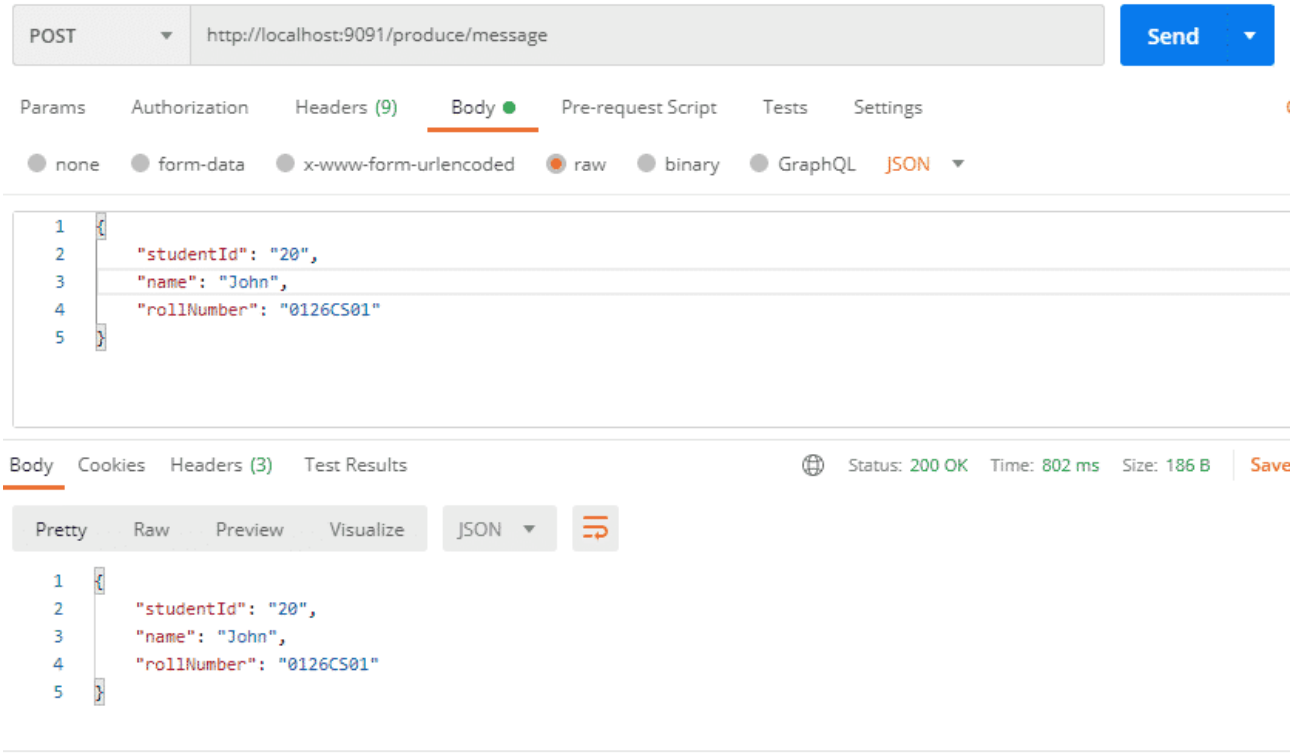
Note – Our producer class and other classes would be same, only we are going to changes Consumer.java.

Consumer.java

```
package com.netsurfingzone.consumer;

import javax.jms.Queue;
```

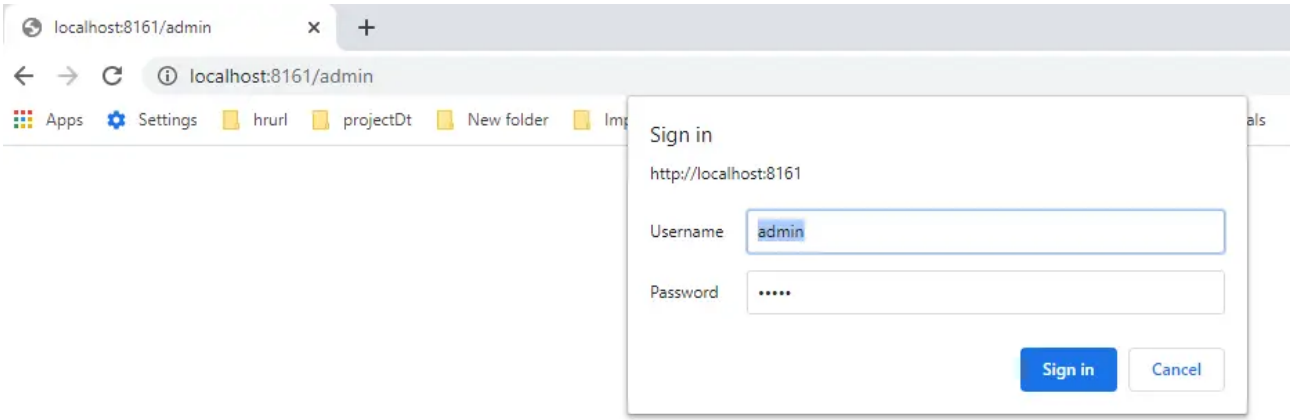
```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jms.core.JmsTemplate;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.netsurfingzone.dto.Student;

@RestController
@RequestMapping("/consume")
public class Consumer {

	@Autowired
	private JmsTemplate jmsTemplate;

	@Autowired
	private Queue queue;

	@GetMapping("/message")
	public Student consumeMessage() {

		Student student = null;
		try {
			ObjectMapper mapper = new ObjectMapper();
			String jsonMessage = (String)
jmsTemplate.receiveAndConvert(queue);
			student = mapper.readValue(jsonMessage, Student.class);

		} catch (Exception e) {
			e.printStackTrace();
		}
		return student;
	}
}
```

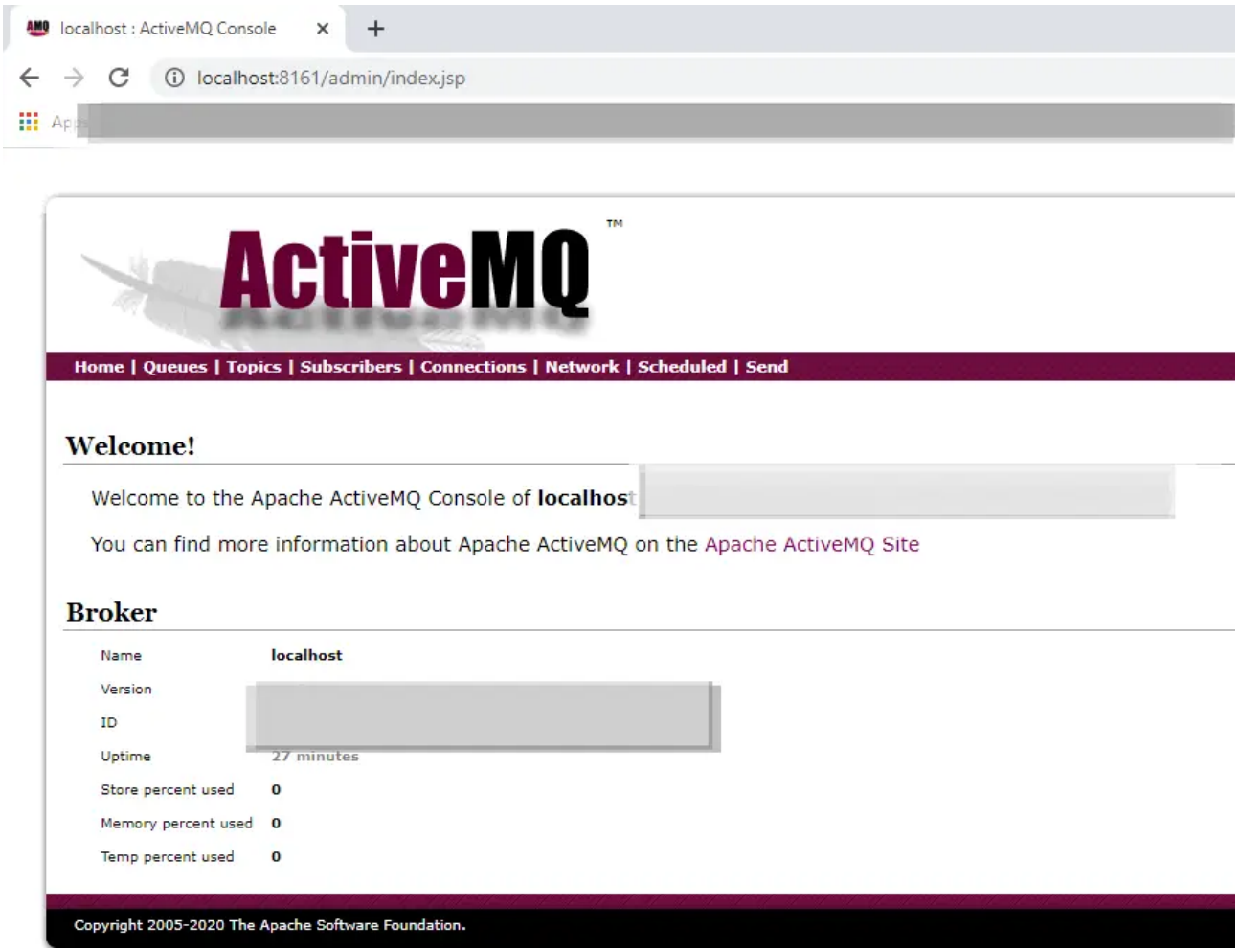
Restart the application and send the message using postman.



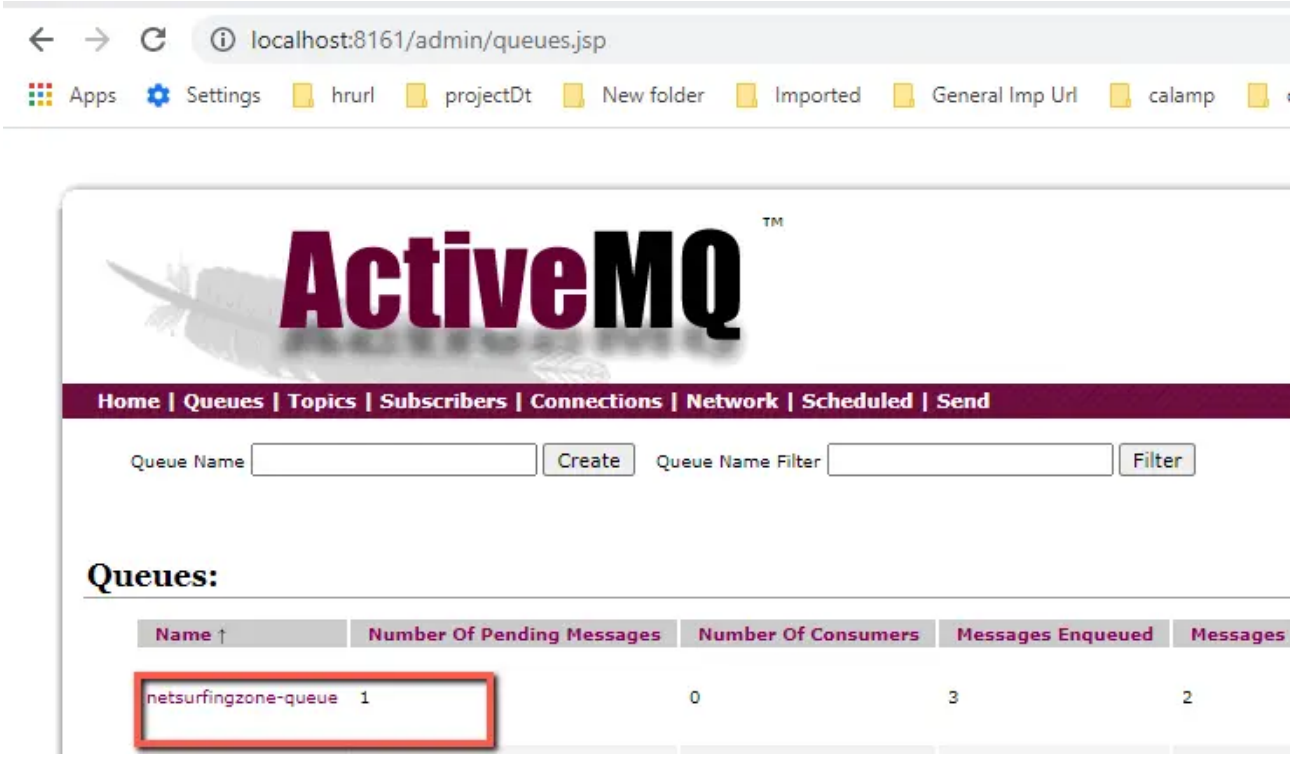
Let’s login to ActiveMQ console.

http://localhost:8161/admin/





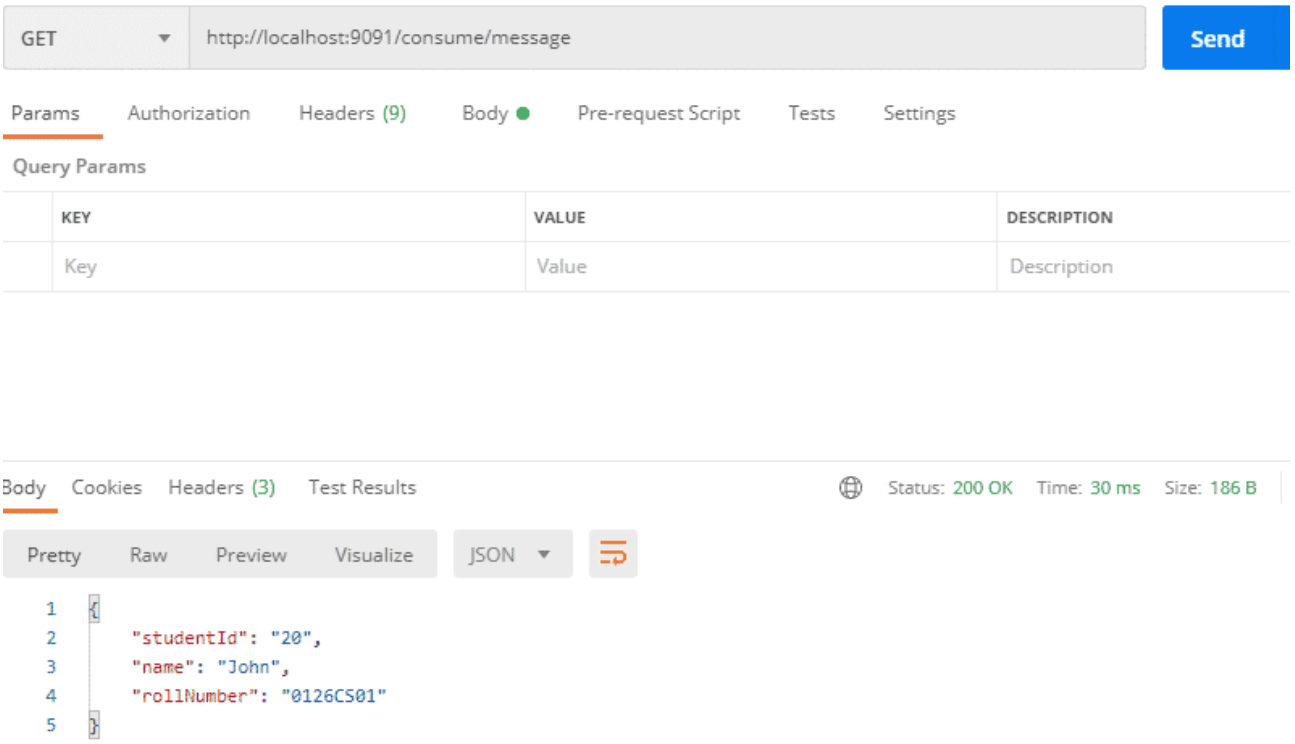
Click on queue.



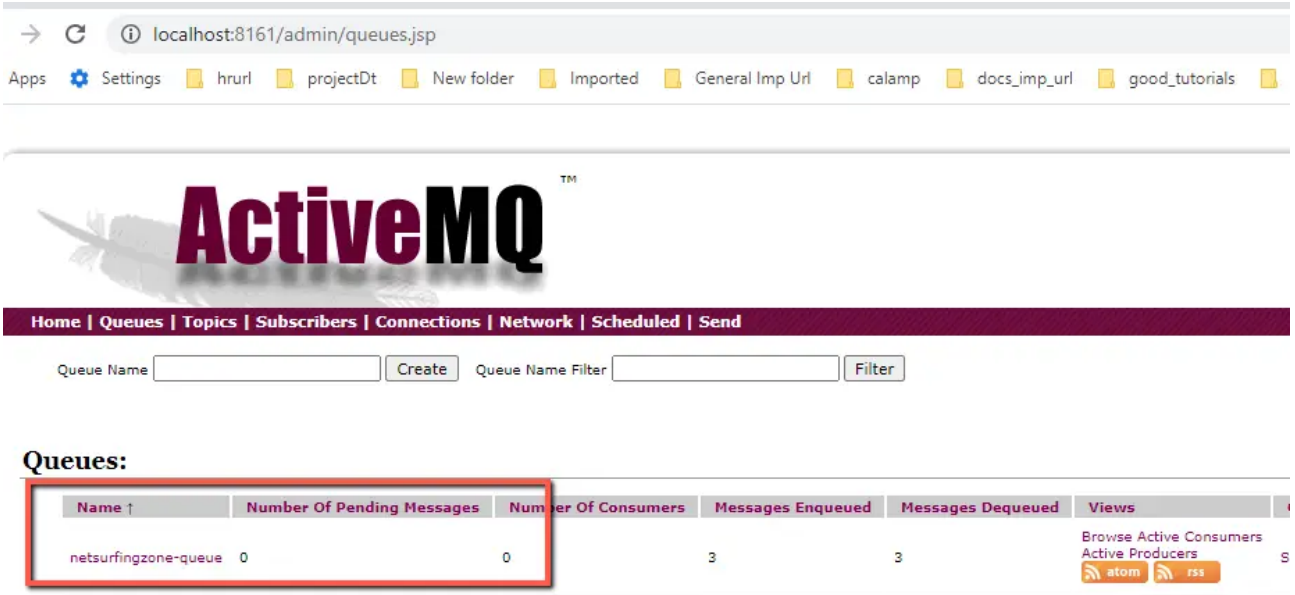
We should able to see message.

Let’s consume/read this using below endpoint.

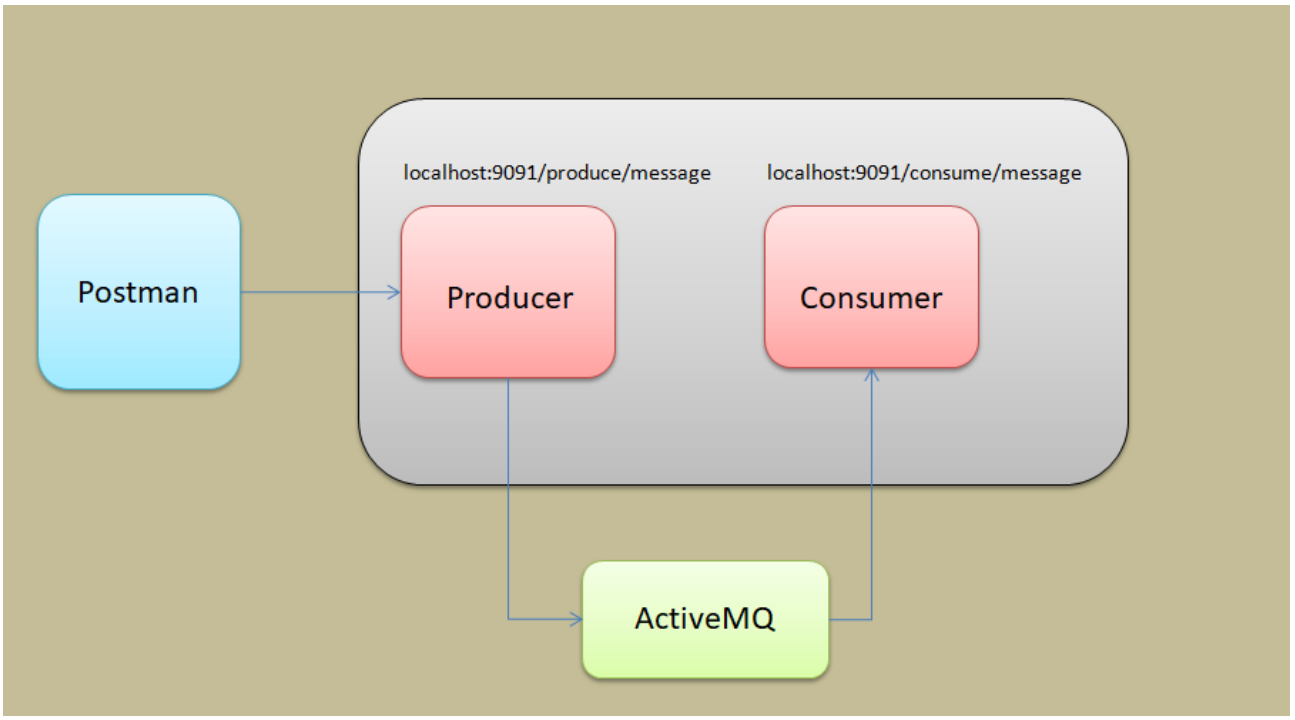
http://localhost:9091/consume/message



Verify the ActiveMQ console. Number of pending message should be zero.



Flow diagram of above example.



That's all about Spring Boot JMS ActiveMQ Producer and Consumer Example.

See Spring JMS example Docs.

See other messaging example using spring boot.

- Spring Boot Kafka Producer and Consumer Example – Step By Step Guide.
- Spring Boot AWS SQS Listener Example.

Some more example.

- Get Session From EntityManager in Spring Boot
- Spring Boot CRUD Example With MySQL/PostgreSQL
- Hazelcast Cache Spring Boot Example
- How to get ApplicationContext in Spring Boot.
- Spring Data JPA @Modifying Annotation Example.
- Hibernate/JPA EhCache Configuration Example.
- OneToMany Mapping using @JoinTable in Hibernate/JPA.
- @OneToMany orphanRemoval true example in Hibernate/JPA
- How to get JPA EntityManager in Spring Boot
- Hibernate Lazy vs Eager loading Example

- JPA and Hibernate Cascade Types example with Spring Boot
- Failed to lazily initialize a collection of role could not initialize proxy – no Session
- JPA EntityManager persist() and merge() method.
- @ElementCollection Example in Hibernate/JPA Using Spring Boot.
- JPA EntityManager CRUD example Using Spring Boot.

Summary – We have seen Spring Boot JMS ActiveMQ Producer and Consumer Example from scratch. We created producer and consumers(using listener and rest end point).