Spring and JMS Integration

To integrate spring with JMS, you need to create two applications.

1. JMS Receiver Application
2. JMS Sender Application

To create JMS application using spring, we are using **Active MQ Server** of Apache to create the Queue.

Let's see the simple steps to integration spring application with JMS:

Required Jar Files

1) You need to add **spring core**, **spring misc**,**spring aop**, **spring j2ee** and **spring persistence core** jar files.

download the all jar files for spring including aop, mvc, j2ee, remoting, oxm, etc.

2) Add **activemqall5.9.jar** file located inside the activemq directory.

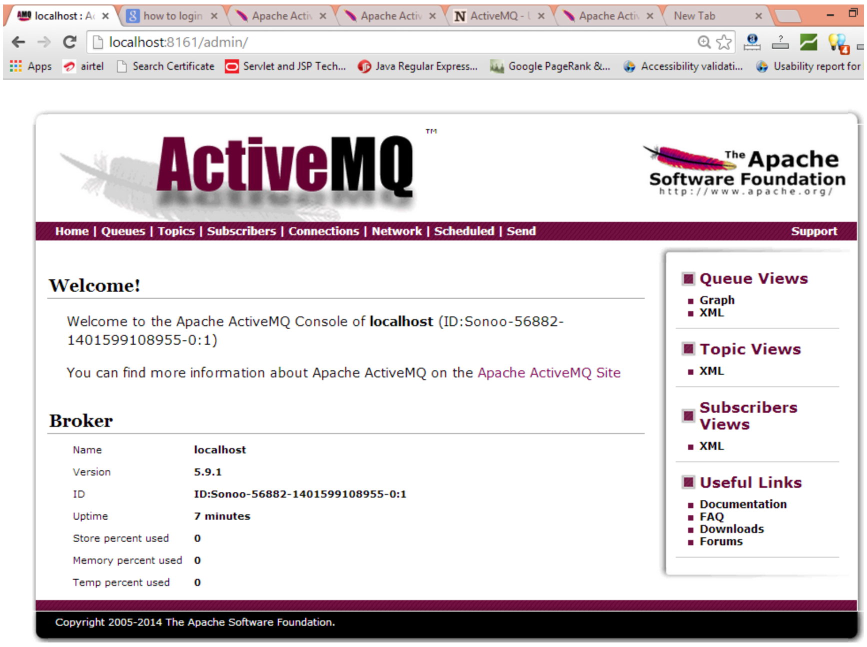
Create a queue in ActiveMQ Server

Download the Active MQ Server Download Active MQ

Double Click on the **activemq.bat** file located inside apache-activemq-5.9.1-bin\apache-activemq-5.9.1\bin\win64 or win32 directory.

Now activemq server console will open.

Access the admin console of activemq server by **http://localhost:8161/admin/** url.



Welcome!

Welcome to the Apache ActiveMQ Console of **localhost** (ID:Sonoo-56882-1401599108955-0:1)

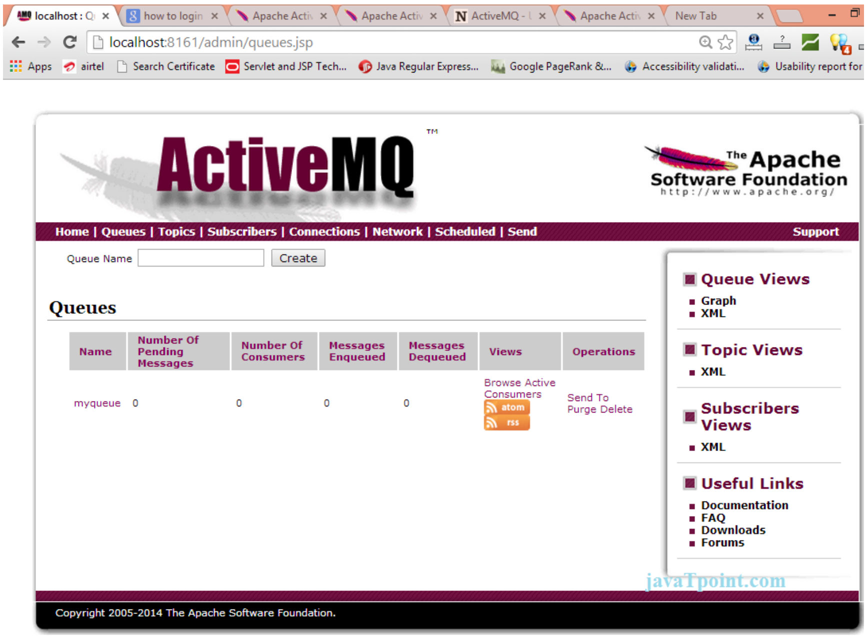
You can find more information about Apache ActiveMQ on the Apache ActiveMQ Site

Broker

Name	localhost
Version	5.9.1
ID	ID:Sonoo-56882-1401599108955-0:1
Uptime	7 minutes
Store percent used	0
Memory percent used	0
Temp percent used	0

Copyright 2005-2014 The Apache Software Foundation.

Now, click on the **Queues link**, write **myqueue** in the textfield and click on the create button.



Queues

Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
myqueue	0	0	0	0	Browse Active Consumers atom rss	Send To Purge Delete

1) JMS Receiver Application

Let's see the simple steps to integration spring application with JMS:

1. **MyMessageListener.java**
2. **TestListener.java**
3. **applicationContext.xml**

1) MyMessageListener.java

```
package com.javatpoint;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.TextMessage;
public class MyMessageListener implements
    @Override
    public void onMessage(Message m) {
                                TextMessage    message=
(TextMessage)m;
        try{
            System.out.println(message.getText());
        }catch (Exception e) {e.printStackTrace();
    }
}
```

---

2) TestListener.java

```
package com.javatpoint;
import org.springframework.context.support.G
public class TestListener {
public static void main(String[] args) {
    GenericXmlApplicationContext ctx=new Ger
    ctx.load("classpath:applicationContext.xml")
    ctx.refresh();

    while(true){}
}
}
```

3) applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/sch
    xmlns:xsi="http://www.w3.org/2001/XMLSch
instance"
    xmlns:jms="http://www.springframework.org


    xmlns:p="http://www.springframework.org/s
    xsi:schemaLocation="http://www.springfram
    http://www.springframework.org/schema/bea
beans-3.0.xsd
    http://www.springframework.org/schema/jms
    http://www.springframework.org/schema/jms
jms-3.0.xsd">

<bean id="connectionFactory" class="org.apa
 p:brokerURL="tcp://localhost:61616" />

<bean id="listener" class="com.javatpoint.MyI
</bean>

<jms:listener-container            container-
type="default"                    connection-
factory="connectionFactory"
 acknowledge="auto">
<jms:listener destination="myqueue" ref="list
</jms:listener>
</jms:listener-container>

</beans>
```

2) JMS Sender Application

Let's see the files to create the JMS Sender application:

1. **MyMessageSender.java**
2. **TestJmsSender.java**
3. **applicationContext.xml**

1) MyMessageListener.java

```
package com.javatpoint;
import javax.jms.*;
import org.springframework.beans.factory.ann
import org.springframework.jms.core.JmsTem
import org.springframework.jms.core.Message
import org.springframework.stereotype.Comp

@Component("messageSender")
public class MyMessageSender {
@Autowired
private JmsTemplate jmsTemplate;
public void sendMessage(final String messag
{
    jmsTemplate.send(new MessageCreator(){

        @Override
        public Message createMessage(Session s
            return session.createTextMessage(mes
        }
    });
}
}
```

2) TestJmsSender.java

```
package com.javatpoint;
import org.springframework.context.support.C
public class TestJmsSender {
```

```
public static void main(String[] args) {
    GenericXmlApplicationContext ctx=new Ger
    ctx.load("classpath:applicationContext.xml")
    ctx.refresh();

    MyMessageSender sender=ctx.getBean("me
    sender.sendMessage("hello jms3");

}
}
```

3) applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/sch
    xmlns:xsi="http://www.w3.org/2001/XMLSch
instance"
    xmlns:context="http://www.springframework
    xmlns:jms="http://www.springframework.org

    xmlns:p="http://www.springframework.org/s
    xsi:schemaLocation="http://www.springfram
    http://www.springframework.org/schema/bea
beans-3.0.xsd
    http://www.springframework.org/schema/cor
    http://www.springframework.org/schema/cor
context-3.0.xsd
    http://www.springframework.org/schema/jms
    http://www.springframework.org/schema/jms
```

```
jms-3.0.xsd">

<bean id="connectionFactory" class="org.apa
 p:brokerURL="tcp://localhost:61616" />

<bean id="jmsTemplate" class="org.springfrar
<constructor-
arg name="connectionFactory" ref="connectic
</constructor-arg>
<property name="defaultDestinationName" va
</property>
</bean>

<context:component-scan            base-
package="com.javatpoint">
</context:component-scan>

</beans>
```

download JMS Receiver example (developed using Myeclipse IDE)

download JMS Sender example(developed using Myeclipse IDE)

<<Prev

Next>>

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

Learn Latest Tutorials

Splunk tutorial

Splunk

SPSS tutorial

SPSS

Swagger

T-SQL tutorial

Transact-SQL

Tumblr tutorial

Tumblr

React tutorial

ReactJS

Regex tutorial

Regex

Reinforcement Learning

RxJS tutorial

R Programming

RxJS

React Native

Python Design Patterns

Python Pillow

Python Turtle

Keras tutorial

Keras

Preparation

Aptitude

Aptitude

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence

AWS Tutorial

AWS

Selenium

Cloud Computing

Hadoop

ReactJS

Data Science

Angular 7

Blockchain

Git Tutorial

Git

Machine Learning

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures

DAA tutorial

DAA

Operating System

Computer Network

Compiler Design

Computer Organization

Discrete Mathematics

Ethical Hacking

Computer Graphics

Software Engineering

html tutorial

Web Technology

Cyber Security

Automata

C++ tutorial

C Programming

C++

Java tutorial

Java

.Net

Python tutorial

Python

Programs

Control System

Data Mining

Data Warehouse