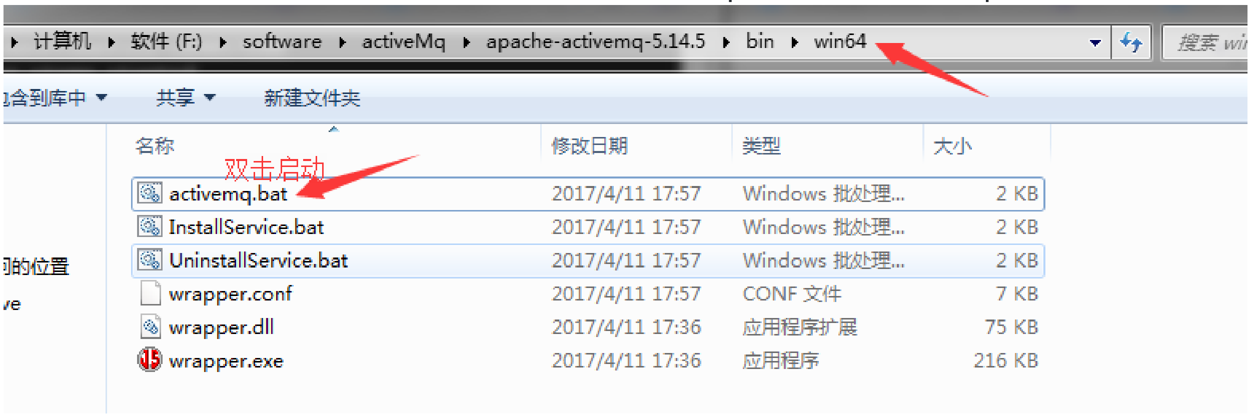
第一步:下载安装ActiveMq

1.下载地址(http://activemq.apache.org/activemq-5145-release)

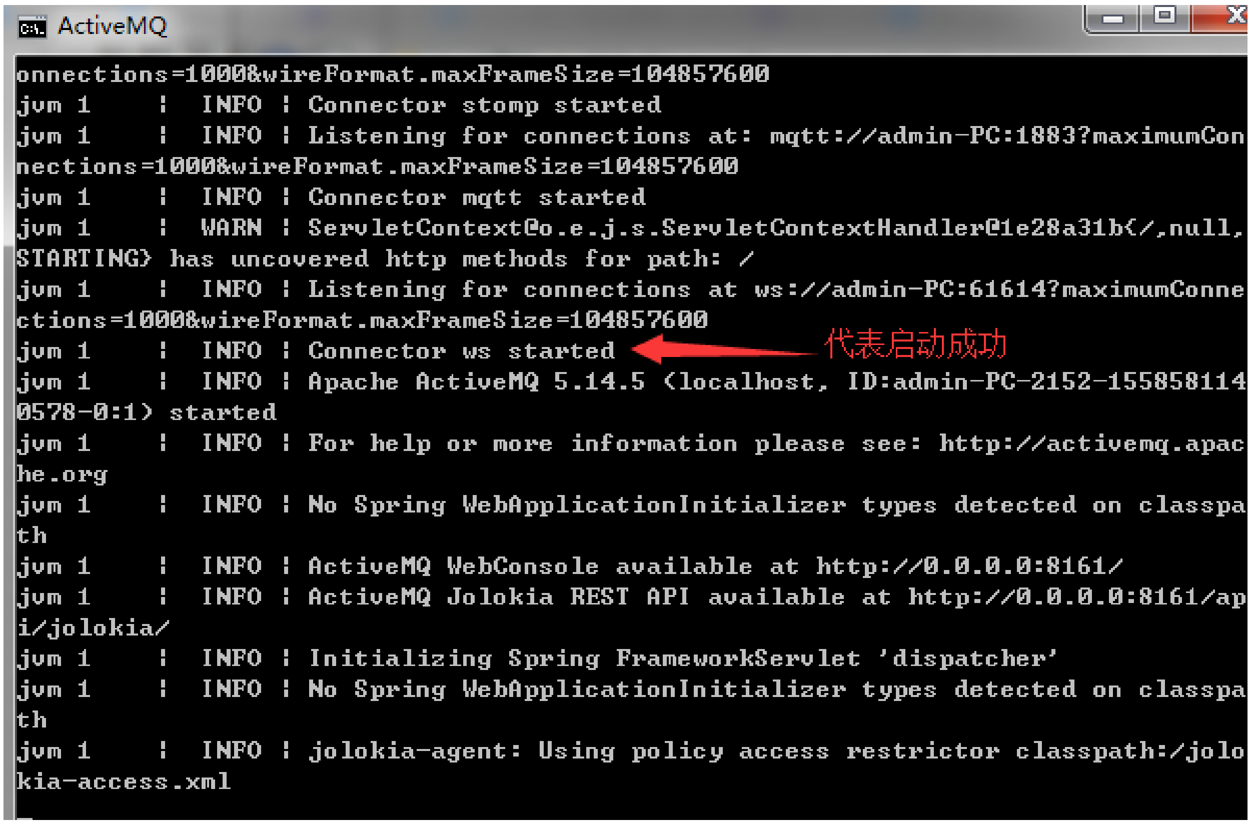
2.配置本地环境变量 path `F:\software\activeMq\apache-activemq-5.14.5\bin`

3,启动activeMq
我的电脑是win64位，所以进入win64目录下双击activemq.bat，启动activemq,如下图


计算机 ▸ 软件 (F:) ▸ software ▸ activeMq ▸ apache-activemq-5.14.5 ▸ bin ▸ win64
包含到库中
共享
新建文件夹
双击启动

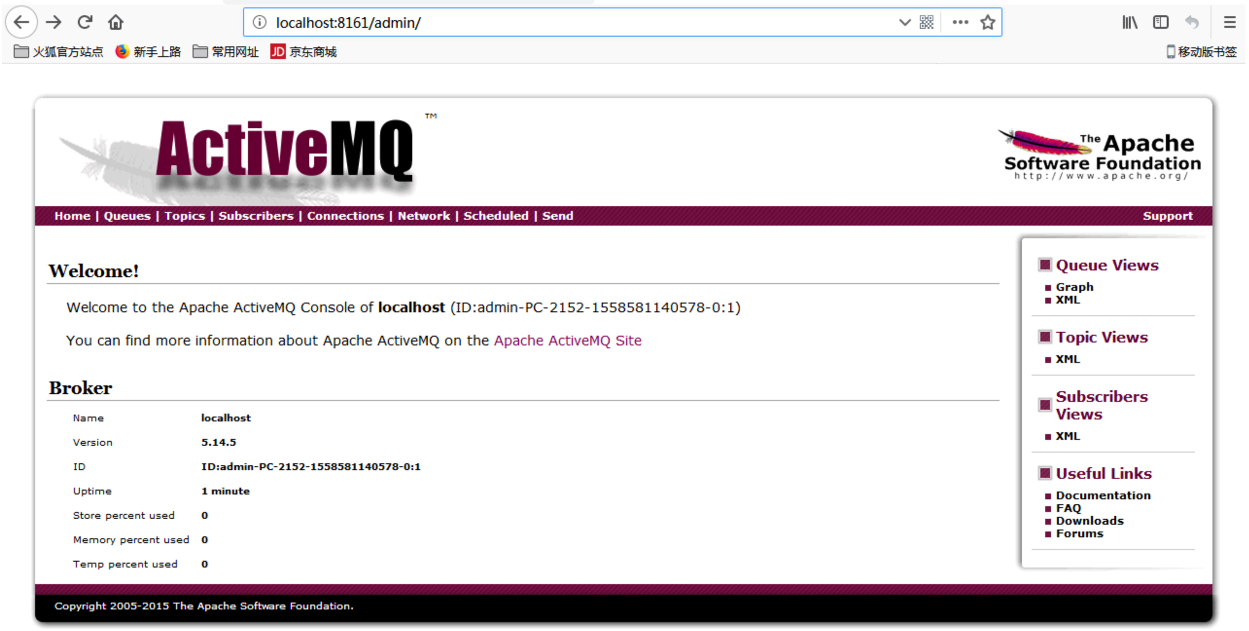
名称	修改日期	类型	大小
activemq.bat	2017/4/11 17:57	Windows 批处理...	2 KB
InstallService.bat	2017/4/11 17:57	Windows 批处理...	2 KB
UninstallService.bat	2017/4/11 17:57	Windows 批处理...	2 KB
wrapper.conf	2017/4/11 17:57	CONF 文件	7 KB
wrapper.dll	2017/4/11 17:36	应用程序扩展	75 KB
wrapper.exe	2017/4/11 17:36	应用程序	216 KB

4.启动成功



```
onnections=1000&wireFormat.maxFrameSize=104857600
jvm 1    |  INFO | Connector stomp started
jvm 1    |  INFO | Listening for connections at: mqtt://admin-PC:1883?maximumCon
nections=1000&wireFormat.maxFrameSize=104857600
jvm 1    |  INFO | Connector mqtt started
jvm 1    |  WARN | ServletContext@o.e.j.s.ServletContextHandler@1e28a31b{/,null,
STARTING} has uncovered http methods for path: /
jvm 1    |  INFO | Listening for connections at ws://admin-PC:61614?maximumConne
ctions=1000&wireFormat.maxFrameSize=104857600
jvm 1    |  INFO | Connector ws started
jvm 1    |  INFO | Apache ActiveMQ 5.14.5 (localhost, ID:admin-PC-2152-155858114
0578-0:1) started
jvm 1    |  INFO | For help or more information please see: http://activemq.apac
he.org
jvm 1    |  INFO | No Spring WebApplicationInitializer types detected on classpa
th
jvm 1    |  INFO | ActiveMQ WebConsole available at http://0.0.0.0:8161/
jvm 1    |  INFO | ActiveMQ Jolokia REST API available at http://0.0.0.0:8161/ap
i/jolokia/
jvm 1    |  INFO | Initializing Spring FrameworkServlet 'dispatcher'
jvm 1    |  INFO | No Spring WebApplicationInitializer types detected on classpa
th
jvm 1    |  INFO | jolokia-agent: Using policy access restrictor classpath:/jolo
kia-access.xml
```

5.在浏览器上访问 `http://localhost:8161/admin` ,会出现如下图



Welcome!

Welcome to the Apache ActiveMQ Console of **localhost** (ID:admin-PC-2152-1558581140578-0:1)

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

Broker

Name	localhost
Version	5.14.5
ID	ID:admin-PC-2152-1558581140578-0:1
Uptime	1 minute
Store percent used	0
Memory percent used	0
Temp percent used	0

Copyright 2005-2015 The Apache Software Foundation.

第二步:写代码，基于spring

1.依赖引入

```
  <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jms</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.8.0</version>
    </dependency>

    <dependency>
      <groupId>org.apache.activemq</groupId>
      <artifactId>activemq-core</artifactId>
      <version>5.7.0</version>
    </dependency>
    <dependency>
      <groupId>org.apache.activemq</groupId>
      <artifactId>activemq-pool</artifactId>
      <version>5.7.0</version>
    </dependency>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-api</artifactId>
      <version>5.0.0</version>
    </dependency>

    <dependency>
      <groupId>org.glassfish.main.javaee-api</groupId>
      <artifactId>javax.jms</artifactId>
```

```
        <version>3.1.2.2</version>
    </dependency>
```

2.jms-connection.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd"
       default-autowire="byName">

    <bean id="activeMQConnectionFactory"
class="org.apache.activemq.ActiveMQConnectionFactory">
        <property name="brokerURL" value="tcp://localhost:61616"/>
    </bean>

    <bean id="pooledConnectionFactory"
class="org.apache.activemq.pool.PooledConnectionFactory">
        <property name="connectionFactory" ref="activeMQConnectionFactory"/>
        <property name="maxConnections" value="10"/>
    </bean>

    <bean id="connectionFactory"
class="org.springframework.jms.connection.SingleConnectionFactory">
        <property name="targetConnectionFactory" ref="pooledConnectionFactory"/>
    </bean>
    <bean id="jmsTemplate" class="org.springframework.jms.core.JmsTemplate">
        <property name="connectionFactory" ref="connectionFactory"/>
    </bean>

    <!--发送消息配置-->
    <bean id="checkMessage" class="org.apache.activemq.command.ActiveMQQueue">
        <constructor-arg index="0" value="huahua.check.message.check"/>
    </bean>
</beans>
```

3.application-jms-mgr.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd"
       default-autowire="byName">
    <import resource="jms-connection.xml"/>

    <!--接受消息配置-->
    <bean id="mgrCheckTopicDestination"
class="org.apache.activemq.command.ActiveMQQueue">
        <constructor-arg index="0" value="huahua.check.message.check"/>
    </bean>
    <bean id="mgrCheckMessageListener"
class="com.hh.receiveactivemq.ReceiveActiveMqMessageListener">
```

```
        <property name="taskExecutor" ref="mgrCheckTaskExecutor"/>
    </bean>

    <!--connectionFactory这个是获取连接池，然后判断连接池里面有没有mgrCheckTopicDestination配置的huahua消息，如果有再进mgrCheckMessageListener类里进行业务-->
    <bean id="mgrCheckJmsContainer"
class="org.springframework.jms.listener.DefaultMessageListenerContainer">
        <property name="connectionFactory" ref="connectionFactory"/>
        <property name="destination" ref="mgrCheckTopicDestination"/>
        <property name="messageListener" ref="mgrCheckMessageListener"/>
    </bean>

    <bean id="mgrCheckTaskExecutor"
class="org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor">
        <!--线程池维护线程的最小数量-->
        <property name="corePoolSize" value="5"/>
        <!--线程池维护线程所允许的空闲时间-->
        <property name="keepAliveSeconds" value="60"/>
        <!--线程池维护线程的最大数量-->
        <property name="maxPoolSize" value="1000"/>
        <!--线程池所使用的缓冲队列-->
        <property name="queueCapacity" value="200"/>
        <property name="rejectedExecutionHandler">
            <bean class="java.util.concurrent.ThreadPoolExecutor$AbortPolicy"/>
        </property>
    </bean>
</beans>
```

4.写一个JmsUtil类

```
package com.hh.sendactivemq;
import org.springframework.jms.core.JmsTemplate;
import org.springframework.jms.core.MessageCreator;
import javax.jms.*;

public class JmsUtil {
    public static void sendMessage(JmsTemplate jmsTemplate, Destination destination,
final String message) {

        jmsTemplate.send(destination, new MessageCreator() {
            public Message createMessage(Session session) throws JMSException {
                return session.createTextMessage(message);
            }
        });
    }
}
```

5.写发送消息的类

```
@Controller
@RequestMapping("user")
@ResponseBody
```

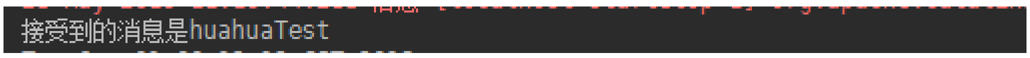
```
public class HelloCtrl{
    @Autowired
    private  JmsTemplate jmsTemplate;
    @Autowired
    private javax.jms.Destination checkMessage;
    @RequestMapping("/sendActiveMq")
    public String sendActiveMq(){
        String str = "huahuaTest";
        JmsUtil.sendMessage(jmsTemplate, checkMessage, str);
        return "success";
    }
}
```

6.写接受消息的类

```
package com.hh.receiveactivemq;

import org.springframework.core.task.TaskExecutor;
import org.springframework.util.StringUtils;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.TextMessage;

public class ReceiveActiveMqMessageListener implements MessageListener {
    private TaskExecutor taskExecutor;
    @Override
    public void onMessage(Message message) {
        if(message instanceof TextMessage){
            TextMessage tm = (TextMessage) message;

            String text = null;
            try {
                text = tm.getText();
                if(!StringUtils.isEmpty(text)){
                    System.out.println("接受到的消息是"+text);
                }
            } catch (JMSException e) {
                e.printStackTrace();
            }

        }
    }
    public void setTaskExecutor(TaskExecutor taskExecutor){
        this.taskExecutor = taskExecutor;
    }
}
```

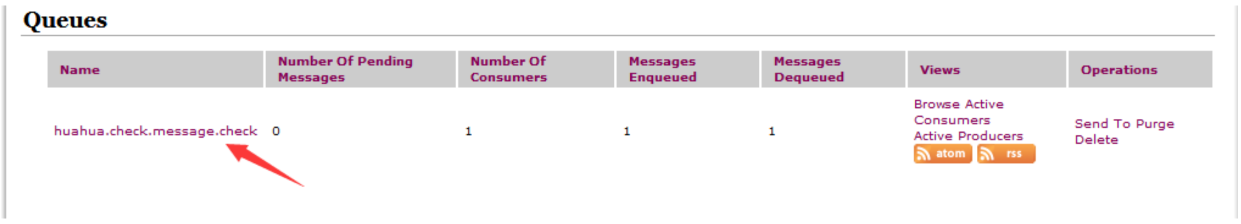
第三步：启动项目

1.运行结果
这句话是在接受消息代码里打印的，启动项目打印了这句话说明发送消息成功，接受消息成功



```
接受到的消息是huahuaTest
```

2.查看queue 发现出现了如下图，则代表接受消息成功



Queues

Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
huahua.check.message.check	0	1	1	1	Browse Active Consumers Active Producers atom rss	Send To Purge Delete