# Spring Boot核心-企业级开发-异步消息

## 异步消息

异步消息主要目的是为了系统与系统之间的通信。所谓异步消息即消息发送者无须等待消息接收者的处理及返回，甚至无须关心消息是否发送成功。在异步消息中有两个很重要的概念，即消息代理（ message broker ）和目的地（ destination ）。

当消息发送者发送消息后，消息将由消息代理接管，消息代理保证消息传递到指定的目的地。异步消息主要有两种形式的目的地：队列（ queue ）和主题（ topic ）。队列用于点对点式(point-to-point ）的消息通信：主题用于发布／订阅式（ publish/subscribe ）的消息通信。

### 点对点式

当消息发送者发送消息，消息代理获得消息后将消息放进一个队列（ queue ）里，当有消息接收者来接收消息的时候，消息将从队列里取出来传递给接收者，这时候队列里就没有了这条消息。

点对点式确保的是每一条消息只有唯一的发送者和接收者，但这并不能说明只有一个接收者可以从队列里接收消息。因为队列里有多个消息，点对点式只保证每一条消息只有唯一的发送者和接收者。

### 发布/订阅式

发布/订阅式是消息发送者发地消息到主题（topic），而多个消息接收者监听这个主题。此时的消息发送者和接收者分别叫做发布者和订阅者。

## 企业级消息代理

JMS( Java Message Service) 即 Java 消息服务，是基于JVM 消息代理的规范，而ActiveMQ，HometQ 是一个JMS 消息代理的实现。

AMQP (Advanced Message Queuing Protocol ）也是一个消息代理的规范，但它不仅兼容JMS ，还支持跨语言和平台。AMQP 的主要实现有RabbitMQ。

## Spring 的支持

Spring 对JMS 和AMQP 的支持分别来自于spring-jms 和Spring-rabbit 。

它们分别需要ConnectionFactory 的实现来连接消息代理，并分别提供了JmsTemplate 、RabbitTemplate 来发送消息。

Spring 为JMS 、AMQP 提供了@JmsListeper、@RabbitListener 注解在方法上监听消息、代理发布的消息。我们需要分别通过@EnableJms 、@EnableRabbit 开启支持。

## Spring Boot 的支持

Spring Boot 对JMS 的自动配置支持位于org.springframework.boot.autoconfigure.jms 下，支持JMS 的实现有ActiveMQ 、HometQ 、Artemis （由HometQ 捐赠给ActiveMQ 的代码库形成的ActiveMQ 的子项目）。这里我们以ActiveMQ 为例， Spring Boot 为我们定义了ActiveMQConnectionFactory的Bean 作为连接，并通过“ spring.activemq ”为前缀的属性米配置ActiveMQ 的连接属性。

|  |
| --- |
| *# ----------------------------------------*  *# INTEGRATION PROPERTIES*  *# ----------------------------------------*  *# ACTIVEMQ (*[ActiveMQProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.6.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/jms/activemq/ActiveMQProperties.java))  spring.activemq.broker-url= *# URL of the ActiveMQ broker. Auto-generated by default.*  spring.activemq.close-timeout=15000 *# Time to wait, in milliseconds, before considering a close complete.*  spring.activemq.in-memory=true *# Specify if the default broker URL should be in memory. Ignored if an explicit broker has been specified.*  spring.activemq.non-blocking-redelivery=false *# Do not stop message delivery before re-delivering messages from a rolled back transaction. This implies that message order will not be preserved when this is enabled.*  spring.activemq.password= *# Login password of the broker.*  spring.activemq.send-timeout=0 *# Time to wait, in milliseconds, on Message sends for a response. Set it to 0 to indicate to wait forever.*  spring.activemq.user= *# Login user of the broker.*  spring.activemq.packages.trust-all= *# Trust all packages.*  spring.activemq.packages.trusted= *# Comma-separated list of specific packages to trust (when not trusting all packages).*  spring.activemq.pool.block-if-full=true *# Block when a connection is requested and the pool is full. Set it to false to throw a "JMSException" instead.*  spring.activemq.pool.block-if-full-timeout=-1 *# Blocking period, in milliseconds, before throwing an exception if the pool is still full.*  spring.activemq.pool.create-connection-on-startup=true *# Create a connection on startup. Can be used to warm-up the pool on startup.*  spring.activemq.pool.enabled=false *# Whether a PooledConnectionFactory should be created instead of a regular ConnectionFactory.*  spring.activemq.pool.expiry-timeout=0 *# Connection expiration timeout in milliseconds.*  spring.activemq.pool.idle-timeout=30000 *# Connection idle timeout in milliseconds.*  spring.activemq.pool.max-connections=1 *# Maximum number of pooled connections.*  spring.activemq.pool.maximum-active-session-per-connection=500 *# Maximum number of active sessions per connection.*  spring.activemq.pool.reconnect-on-exception=true *# Reset the connection when a "JMXException" occurs.*  spring.activemq.pool.time-between-expiration-check=-1 *# Time to sleep, in milliseconds, between runs of the idle connection eviction thread. When negative, no idle connection eviction thread runs.*  spring.activemq.pool.use-anonymous-producers=true *# Use only one anonymous "MessageProducer" instance. Set it to false to create one "MessageProducer" every time one is required.* |

Spring Boot 在JmsAutoConfiguration 还为我们配置好了JmsTemplate ，且为我们开启了注解式消息监听的支持，即自动开启@EnableJms。

Spring Boot 对AMQP 的自动配置支持位于org.springframework. boot.autoconfigure.amqp下，它为我们自己置了连接的ConnectionFactory 和RabbitTemplate，且为我们开启了注解式消息监听，还自动开启@EnableRabbito RabbitMQ 的配置可通过“ spring.rabbitmq ”来配置RabbitMQ，主要包含：

|  |
| --- |
| *# RABBIT (*[RabbitProperties](https://github.com/spring-projects/spring-boot/tree/v1.5.6.RELEASE/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/amqp/RabbitProperties.java))  spring.rabbitmq.addresses= *# Comma-separated list of addresses to which the client should connect.*  spring.rabbitmq.cache.channel.checkout-timeout= *# Number of milliseconds to wait to obtain a channel if the cache size has been reached.*  spring.rabbitmq.cache.channel.size= *# Number of channels to retain in the cache.*  spring.rabbitmq.cache.connection.mode=channel *# Connection factory cache mode.*  spring.rabbitmq.cache.connection.size= *# Number of connections to cache.*  spring.rabbitmq.connection-timeout= *# Connection timeout, in milliseconds; zero for infinite.*  spring.rabbitmq.dynamic=true *# Create an AmqpAdmin bean.*  spring.rabbitmq.host=localhost *# RabbitMQ host.*  spring.rabbitmq.listener.simple.acknowledge-mode= *# Acknowledge mode of container.*  spring.rabbitmq.listener.simple.auto-startup=true *# Start the container automatically on startup.*  spring.rabbitmq.listener.simple.concurrency= *# Minimum number of consumers.*  spring.rabbitmq.listener.simple.default-requeue-rejected= *# Whether or not to requeue delivery failures; default `true`.*  spring.rabbitmq.listener.simple.idle-event-interval= *# How often idle container events should be published in milliseconds.*  spring.rabbitmq.listener.simple.max-concurrency= *# Maximum number of consumers.*  spring.rabbitmq.listener.simple.prefetch= *# Number of messages to be handled in a single request. It should be greater than or equal to the transaction size (if used).*  spring.rabbitmq.listener.simple.retry.enabled=false *# Whether or not publishing retries are enabled.*  spring.rabbitmq.listener.simple.retry.initial-interval=1000 *# Interval between the first and second attempt to deliver a message.*  spring.rabbitmq.listener.simple.retry.max-attempts=3 *# Maximum number of attempts to deliver a message.*  spring.rabbitmq.listener.simple.retry.max-interval=10000 *# Maximum interval between attempts.*  spring.rabbitmq.listener.simple.retry.multiplier=1.0 *# A multiplier to apply to the previous delivery retry interval.*  spring.rabbitmq.listener.simple.retry.stateless=true *# Whether or not retry is stateless or stateful.*  spring.rabbitmq.listener.simple.transaction-size= *# Number of messages to be processed in a transaction. For best results it should be less than or equal to the prefetch count.*  spring.rabbitmq.password= *# Login to authenticate against the broker.*  spring.rabbitmq.port=5672 *# RabbitMQ port.*  spring.rabbitmq.publisher-confirms=false *# Enable publisher confirms.*  spring.rabbitmq.publisher-returns=false *# Enable publisher returns.*  spring.rabbitmq.requested-heartbeat= *# Requested heartbeat timeout, in seconds; zero for none.*  spring.rabbitmq.ssl.enabled=false *# Enable SSL support.*  spring.rabbitmq.ssl.key-store= *# Path to the key store that holds the SSL certificate.*  spring.rabbitmq.ssl.key-store-password= *# Password used to access the key store.*  spring.rabbitmq.ssl.trust-store= *# Trust store that holds SSL certificates.*  spring.rabbitmq.ssl.trust-store-password= *# Password used to access the trust store.*  spring.rabbitmq.ssl.algorithm= *# SSL algorithm to use. By default configure by the rabbit client library.*  spring.rabbitmq.template.mandatory=false *# Enable mandatory messages.*  spring.rabbitmq.template.receive-timeout=0 *# Timeout for `receive()` methods.*  spring.rabbitmq.template.reply-timeout=5000 *# Timeout for `sendAndReceive()` methods.*  spring.rabbitmq.template.retry.enabled=false *# Set to true to enable retries in the `RabbitTemplate`.*  spring.rabbitmq.template.retry.initial-interval=1000 *# Interval between the first and second attempt to publish a message.*  spring.rabbitmq.template.retry.max-attempts=3 *# Maximum number of attempts to publish a message.*  spring.rabbitmq.template.retry.max-interval=10000 *# Maximum number of attempts to publish a message.*  spring.rabbitmq.template.retry.multiplier=1.0 *# A multiplier to apply to the previous publishing retry interval.*  spring.rabbitmq.username= *# Login user to authenticate to the broker.*  spring.rabbitmq.virtual-host= *# Virtual host to use when connecting to the broker.* |