**What is AMQP?**

We have already discussed some [JMS](https://www.journaldev.com/9776/jms1-and-jms2-api-overview) concepts and examples in my previous posts. In this post, we are going to discuss AMQP protocol and Spring AMQP Messaging.

AMQP stands for Advanced Message Queuing Protocol. AMQP is an open standard protocol for implementing MOMs (Message Oriented Middleware).

**Why we need AMQP?**

We have JMS API to develop Enterprise Messaging systems but why we need another Messaging standard.

The main drawback or limitation of JMS API is interoperability that means we can develop Messaging systems that will work only in Java-based applications. It does not support other languages.

AMQP solves the JMS API problem. The major advantage of AMQP is that it supports interoperability between heterogeneous platforms and messaging brokers. We can develop our Messaging systems in any language (Java, C++, C#, Ruby etc.) and in any operating system; still, they can communicate with each other by using AMQP based message brokers.

### How Spring AMQP works?

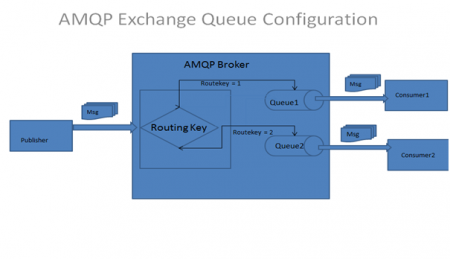
[Spring Framework](https://www.journaldev.com/16922/spring-framework) provides Spring AMQP API to integrate AMQP Message brokers with Spring applications to develop Enterprise Messaging Systems.

In AMQP Messaging systems, Message Publisher sends messages to Exchange. Publisher does not know which queue is configured to this Exchange and which Consumer is associated with this Queue.

While configuring Exchange, we will map it to one or more Queues by specifying some routing key.

AMQP Consumer is connected to Queue and listening to messages with some routing key.

When Publisher sends a message into Exchange, then Exchange uses this routing key and send those messages to associated Queue. When the Queue receives messages, then associated Consumer receives those messages automatically.



For example, we have implemented our AMQP Messaging system by following above architecture.

If Publisher sends a message with Route key = 1, then this message is routed to Queue1, then to Consumer1.

If Publisher sends a message with Route key = 2, then this message is routed to Queue2, then to Consumer2.

**NOTE:** Most popular AMQP protocol implemented Message brokers or JMS Servers are:

1. Apache Active MQ server
2. Spring Rabbit MQ server

### Spring AMQP Modules

Spring Framework provides two set of APIs to deal with AMQP protocol servers. Spring AMQP Projects are located at the following website:

<https://spring.io/projects/spring-amqp>

1. [Spring RabbitMQ](https://www.journaldev.com/11655/spring-rabbitmq) AMQP API

Spring RabbitMQ AMQP API is used to implement messaging applications with Spring Rabbit MQ server. This API is developed in two Spring modules:

* + spring-amqp:

It is base abstraction for AMQP Protocol implementation.

* + spring-rabbit:

It is the RabbitMQ implementation.

1. Spring ActiveMQ AMQP API

This API is used to implement messaging applications with Apache Active MQ server. This API uses Spring AMQP abstraction module with ActiveMQ API:

* + spring-amqp:

It is base abstraction for AMQP Protocol implementation.

* + activemq-spring:

It is the ActiveMQ implementation.