**Spring Boot and RabbitMQ Example** to - Configure Listeners to consume messages from the Queue

Spring Boot Application

Queue

1. First, we will be creating a listener class which implements the **AMQP** Message Listener interface. This class is responsible for getting the message from the **RabbitMQ** queue.
2. Next, we will be creating the Spring Boot Configuration class for **RabbitMQ**

“**TEST**”.

1. This will be a non-durable queue.
2. Do not misunderstand a non-durable queue to be a temporary queue.
3. Durability property is related to how long a message will be stored in the queue.

**Example**: - for in RabbitMQ restart messages in non-durable queue will be lost while those in durable queue will not be lost.

1. Create MessageListenerContainer where we configure the RabbitMQConnections.

Spring MessageListenerContainer is a replacement for a **Message-Driven EJB**.

A connection is set up with the **AMQ topic/queue**, it gets messages from that **topic/queue** and feeds them to your MessageListener.

We will be making use of the default connectionfactory.

If we do not wish to use the default connectionfactory we can can create our own CachingConnectionFactory and use it.

1. go to localhost:15672
2. Use the **username** and **password** as guest. If we now go to the queues section, currently there are no **queues**
3. Start the Spring Boot Application. If we now again go to the **RabbitMQ** **Management** **Console** - **Queues** section, we can see that a queue named “**TEST**” queue has been created.
4. Now select the queue and publish a message.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. If we now go to the eclipse console, we can see that our application has consumed the **RabbitMQ** message and printed its content.