# 2. Introduction

参考文档的第一部分是Spring AMQP的高级概述，其基本概念和一些代码片段将尽快启动并运行。

## 2.1 快速浏览不厌烦

### 2.1.1 介绍

这是Spring MAQP开始的5分钟的旅行。

先决条件：安装并运行([http://www.rabbitmq.com/download.html](https://www.rabbitmq.com/download.html))。然后抓住spring-rabbit jar及其所有的依赖关系-最简单的方法就是在构建工具中声明依赖关系，对于Maven。

<dependency>

<groupId>org.springframework.amqp</groupId>

<artifactId>spring-rabbit</artifactId>

<version>2.0.2.RELEASE</version>

</dependency>

对于gradle:

compile 'org.springframework.amqp:spring-rabbit:2.0.2.RELEASE'

#### Compatibility(兼容性)

最小的Spring Framework版本依赖是5.0.x。

最小的amqp-client java 客户端库版本是5.0.0。

#### 非常, 非常快

使用简单的命令式Java来发送和接收消息：

ConnectionFactory connectionFactory = **new** CachingConnectionFactory();

AmqpAdmin admin = **new** RabbitAdmin(connectionFactory);

admin.declareQueue(**new** Queue("myqueue"));

AmqpTemplate template = **new** RabbitTemplate(connectionFactory);

template.convertAndSend("myqueue", "foo");

String foo = (String) template.receiveAndConvert("myqueue");

请注意，本地Java Rabbit客户端中也有一个ConnectionFactory 。我们在上面的代码中使用Spring抽象，我们 依靠broker中的默认exchange(因为没有在发送中指定)以及所有队列默认绑定到默认exchange name(因此我们可以使用队列名称作为发送中的路由密钥)。这些行为在AMQP规范中定义。

#### With XML Configuration

The same example as above, but externalizing the resource configuration to XML:

ApplicationContext context =

**new** GenericXmlApplicationContext("classpath:/rabbit-context.xml");

AmqpTemplate template = context.getBean(AmqpTemplate.**class**);

template.convertAndSend("myqueue", "foo");

String foo = (String) template.receiveAndConvert("myqueue");

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:rabbit="http://www.springframework.org/schema/rabbit"

xsi:schemaLocation="http://www.springframework.org/schema/rabbit

http://www.springframework.org/schema/rabbit/spring-rabbit.xsd

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<rabbit:connection-factory id="connectionFactory"/>

<rabbit:template id="amqpTemplate" connection-factory="connectionFactory"/>

<rabbit:admin connection-factory="connectionFactory"/>

<rabbit:queue name="myqueue"/>

</beans>

The <rabbit:admin/> declaration by default automatically looks for beans of type Queue, Exchange and Binding and declares them to the broker on behalf of the user, hence there is no need to use that bean explicitly in the simple Java driver. There are plenty of options to configure the properties of the components in the XML schema - you can use auto-complete features of your XML editor to explore them and look at their documentation.

#### With Java Configuration

The same example again with the external configuration in Java:

ApplicationContext context =

**new** AnnotationConfigApplicationContext(RabbitConfiguration.**class**);

AmqpTemplate template = context.getBean(AmqpTemplate.**class**);

template.convertAndSend("myqueue", "foo");

String foo = (String) template.receiveAndConvert("myqueue");

........

*@Configuration*

**public** **class** RabbitConfiguration {

*@Bean*

**public** ConnectionFactory connectionFactory() {

**return** **new** CachingConnectionFactory("localhost");

}

*@Bean*

**public** AmqpAdmin amqpAdmin() {

**return** **new** RabbitAdmin(connectionFactory());

}

*@Bean*

**public** RabbitTemplate rabbitTemplate() {

**return** **new** RabbitTemplate(connectionFactory());

}

*@Bean*

**public** Queue myQueue() {

**return** **new** Queue("myqueue");

}

}

## 2.2 What’s New

### 2.2.1 Changes in 2.0 Since 1.7

#### CachingConnectionFactory

Starting with version 2.0.2, the RabbitTemplate can be configured to use a different connection to that used by listener containers. This is to avoid deadlocked consumers when producers are blocked for any reason. See [the section called “Using a Separate Connection”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#separate-connection) for more information.

#### AMQP Client library

Spring AMQP now uses the new 5.0.x version of the amqp-client library provided by the RabbitMQ team. This client has auto recovery configured by default; see [the section called “RabbitMQ Automatic Connection/Topology recovery”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#auto-recovery).

|  |
| --- |
| [Note] |
| As of version 4.0, the client enables automatic recovery by default; while compatible with this feature, Spring AMQP has its own recovery mechanisms and the client recovery feature generally isn’t needed. It is recommended to disable amqp-client automatic recovery, to avoid getting AutoRecoverConnectionNotCurrentlyOpenException s when the broker is available, but the connection has not yet recovered. Starting with version 1.7.1, Spring AMQP disables it unless you explicitly create your own RabbitMQ connection factory and provide it to the CachingConnectionFactory. RabbitMQ ConnectionFactory instances created by the RabbitConnectionFactoryBean will also have the option disabled by default. |

#### General Changes

The ExchangeBuilder now builds durable exchanges by default. The @Exchange annotation used within a @QeueueBinding also declares durable exchanges by default. The @Queue annotation used within a @RabbitListener by default declares durable queues if named and non-durable if anonymous. See [the section called “Builder API for Queues and Exchanges”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#builder-api) and [the section called “Annotation-driven Listener Endpoints”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#async-annotation-driven) for more information.

#### Deleted classes

UniquelyNameQueue is no longer provided. It is unusual to create a durable non auto-delete queue with a unique name. This class has been deleted; if you require its functionality, use new Queue(UUID.randomUUID().toString()).

#### New Listener Container

The DirectMessageListenerContainer has been added alongside the existing SimpleMessageListenerContainer. See [the section called “Choosing a Container”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#choose-container)and [Section 3.1.15, “Message Listener Container Configuration”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#containerAttributes) for information about choosing which container to use as well as how to configure them.

#### Log4j Appender

This appender is no longer available due to the end-of-life of log4j. See [Section 3.2, “Logging Subsystem AMQP Appenders”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#logging) for information about the available log appenders.

#### Logback Appender

This appender no longer captures caller data (method, line number) by default; it can be re-enabled by setting the includeCallerData configuration option. See [Section 3.2, “Logging Subsystem AMQP Appenders”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#logging) for information about the available log appenders.

#### RabbitTemplate Changes

|  |  |
| --- | --- |
| [Important] | **Important** |
| Previously, a non-transactional RabbitTemplate participated in an existing transaction if it ran on a transactional listener container thread. This was a serious bug; however, users might have relied on this behavior. Starting with version 1.6.2, you must set the channelTransacted boolean on the template for it to participate in the container transaction. |

The RabbitTemplate now uses a DirectReplyToMessageListenerContainer (by default) instead of creating a new consumer for each request. See [the section called “RabbitMQ Direct reply-to”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#direct-reply-to) for more information.

The AsyncRabbitTemplate now supports Direct reply-to; see [the section called “AsyncRabbitTemplate”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#async-template) for more information.

The RabbitTemplate and AsyncRabbitTemplate now have receiveAndConvert and convertSendAndReceiveAsType methods that take a ParameterizedTypeReference<T> argument, allowing the caller to specify the type to convert the result to. This is particularly useful for complex types or when type information is not conveyed in message headers. Requires a SmartMessageConverter such as the Jackson2JsonMessageConverter. See [Section 3.1.6, “Receiving messages”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#receiving-messages), [Section 3.1.9, “Request/Reply Messaging”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#request-reply), [the section called “AsyncRabbitTemplate”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#async-template), and [the section called “Converting From a Message With RabbitTemplate”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#json-complex) for more information.

You can now use a RabbitTemplate to perform multiple operations on a dedicated channel. See [the section called “Scoped Operations”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#scoped-operations) for more information.

#### Listener Adapter

A convenient FunctionalInterface is available for using lambdas with the MessageListenerAdapter. See [the section called “MessageListenerAdapter”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#message-listener-adapter) for more information.

#### Listener Container Changes

##### Prefetch default value

The prefetch default value used to be 1, which could lead to under-utilization of efficient consumers. The default prefetch value is now 250, which should keep consumers busy in most common scenarios and thus improve throughput.

|  |  |
| --- | --- |
| [Important] | **Important** |
| There are nevertheless scenarios where the prefetch value should be low: for example, with large messages, especially if the processing is slow (messages could add up to a large amount of memory in the client process), and if strict message ordering is necessary (the prefetch value should be set back to 1 in this case). Also, with low-volume messaging and multiple consumers (including concurrency within a single listener container instance), you may wish to reduce the prefetch to get a more even distribution of messages across consumers. |

For more background about prefetch, see this post about [consumer utilization in RabbitMQ](https://www.rabbitmq.com/blog/2014/04/14/finding-bottlenecks-with-rabbitmq-3-3/) and this post about [queuing theory](https://www.rabbitmq.com/blog/2012/05/11/some-queuing-theory-throughput-latency-and-bandwidth/).

##### Message Count

Previously, MessageProperties.getMessageCount() returned 0 for messages emitted by the container. This property only applies when using basicGet (e.g. from RabbitTemplate.receive() methods) and is now initialized to null for container messages.

##### Transaction Rollback behavior

Message requeue on transaction rollback is now consistent, regardless of whether or not a transaction manager is configured. See [the section called “A note on Rollback of Received Messages”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#transaction-rollback) for more information.

##### Shutdown Behavior

If the container threads do not respond to a shutdown within shutdownTimeout, the channel(s) will be forced closed, by default. See [Section 3.1.15, “Message Listener Container Configuration”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#containerAttributes) for more information.

##### After Receive Message Post Processors

If a MessagePostProcessor in the afterReceiveMessagePostProcessors property returns null, the message is discarded (and acknowledged if appropriate).

#### Connection Factory Changes

The connection and channel listener interfaces now provide a mechanism to obtain information about exceptions. See [the section called “Connection and Channel Listeners”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#connection-channel-listeners) and [the section called “Publishing is Asynchronous - How to Detect Success and Failures”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#publishing-is-async) for more information.

A new ConnectionNameStrategy is now provided to populate the application-specific identification of the target RabbitMQ connection from the AbstractConnectionFactory. See [Section 3.1.2, “Connection and Resource Management”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#connections) for more information.

#### Retry Changes

The MissingMessageIdAdvice is no longer provided; it’s functionality is now built-in; see [the section called “Failures in Synchronous Operations and Options for Retry”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#retry)for more information.

#### Anonymous Queue Naming

By default, AnonymousQueues are now named with the default Base64UrlNamingStrategy instead of a simple UUID string. See [the section called “AnonymousQueue”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#anonymous-queue) for more information.

#### @RabbitListener Changes

You can now provide simple queue declarations (only bound to the default exchange) in @RabbitListener annotations. See [the section called “Annotation-driven Listener Endpoints”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#async-annotation-driven) for more information.

You can now configure @RabbitListener annotations so that any exceptions thrown will be returned to the sender. You can also configure a RabbitListenerErrorHandler to handle exceptions. See [the section called “Handling Exceptions”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#annotation-error-handling) for more information.

You can now bind a queue with multiple routing keys when using the @QueueBinding annotation. Also @QueueBinding.exchange() now supports custom exchange types and declares durable exchanges by default.

You can now set the concurrency of the listener container at the annotation level rather than having to configure a different container factory for different concurrency settings.

You can now set the autoStartup property of the listener container at the annotation level, overriding the default setting in the container factory.

See [the section called “Annotation-driven Listener Endpoints”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#async-annotation-driven) for more information.

#### Container Conditional Rollback

When using an external transaction manager (e.g. JDBC), rule-based rollback is now supported when providing the container with a transaction attribute. It is also now more flexible when using a transaction advice. See [the section called “Conditional Rollback”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#conditional-rollback) for more information.

#### Remove Jackson 1.x support

Deprecated in previous versions, Jackson 1.x converters and related components have now been deleted; use similar components based on Jackson 2.x. See [the section called “Jackson2JsonMessageConverter”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#json-message-converter) for more information.

#### JSON Message Converter

When the \_\_TypeId\_\_ is set to Hashtable for an inbound JSON message, the default conversion type is now LinkedHashMap; previously it was Hashtable. To revert to a Hashtable use setDefaultMapType on the DefaultClassMapper.

#### XML Parsers

When parsing Queue and Exchange XML components, the parsers no longer register the name attribute value as a bean alias if an id attribute is present. See [the section called “A Note On "id" and "name" Attributes”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#note-id-name) for more information.

#### Blocked Connection

The com.rabbitmq.client.BlockedListener can now be injected into the org.springframework.amqp.rabbit.connection.Connection object. Also the ConnectionBlockedEvent and ConnectionUnblockedEvent events are emitted by the ConnectionFactory, when the connection is blocked or unblocked by the Broker.

See [Section 3.1.2, “Connection and Resource Management”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html#connections) for more information.

### 2.2.2 Earlier Releases

See [Section A.2, “Previous Releases”](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_change_history.html#previous-whats-new) for changes in previous versions.

|  |  |  |
| --- | --- | --- |
| [Prev](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_preface.html) |  | [Next](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/_reference.html) |
| 1. Preface | [Home](https://docs.spring.io/spring-amqp/docs/2.0.2.RELEASE/reference/html/index.html) | 3. Reference |