01-RabbitMQ 开篇 Hello World

先决条件

本教程假定 RabbitMQ 已在标准端口（5672）上的 localhost 上安装并运行。如果使用不同的主机，端口或凭据，连接设置将需要调整。

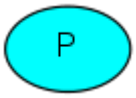
介绍

RabbitMQ 是一个消息代理：它接受并转发消息。您可以将其视为邮局：当您将要发布的邮件放在邮箱中时，您可以确信邮差最终会将邮件发送给收件人。在这个比喻中，RabbitMQ 是一个邮箱，邮局和邮递员。

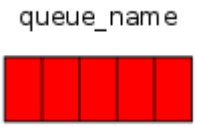
RabbitMQ 和邮局之间的主要区别在于它不处理纸张，而是接受，存储和转发二进制数据块的消息。

RabbitMQ 和消息传递一般使用一些术语。

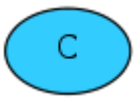
生产意味着不仅仅是发送消息。发送消息的程序是一个生产者：



队列是存在于 RabbitMQ 中的邮箱的名称。虽然消息流过 RabbitMQ 和您的应用程序，但它们只能存储在队列中。一个队列仅由主机的存储器和磁盘限制约束，它本质上是一个大的消息缓冲器。多个生产者可以发送消息到到一个队列，多个消费者可以尝试从一个队列接收数据。下图是代表一个队列的示意图



消费具有与接收相似的含义。一个消费者是一个程序，主要是等待接收信息：



请注意，生产者，消费者和消息代理（Broker）不必驻留在同一个主机上；确实在大多数应用程序中，它们不是。

Hello World

使用 spring-amqp 客户端

在本教程的这一部分中，我们将使用 spring-amqp 库编写两个程序；发送单个消息的生产者，以及接收消息并将其打印出来的消费者。我们将从一个非常简单的例子开始，然后介绍 Spring-amqp API 中的一些细节。下面是一个“Hello World”的消息传递。

在下图中，“P”是我们的生产者，“C”是我们的消费者。中间的框是队列 - RabbitMQ 代表消费者的消息缓冲区。

注意：RabbitMQ 使用了多种协议。本教程使用 AMQP 0-9-1，它是一种开放的通用协议，用于消息传递。有许多不同语言的 RabbitMQ 客户端。我们将使用 RabbitMQ 提供的 Java 客户端。

Spring AMQP 利用 Spring Boot 进行配置和依赖关系管理。Spring 支持 maven 或 gradle，但是在本教程中，我们将使用 Spring Boot 1.5.4 来选择 maven。

配置项目

Spring Boot 提供了许多功能，但我们只会在这里突出一些。首先，Spring Boot 应用程序可以选择通过 application.properties 或 application.yml 文件提供其属性（还有更多的选项，但使我们使用下面这些就足够了）。您将在生成的项目中找到一个 application.properties 文件，其中没有内容。将 application.properties 重命名为具有以下属性的 application.yml 文件：

```
spring:
  profiles:
    active: usage_message
logging:
  level:
    org: ERROR
tutorial:
  client:
    duration: 100000
```

创建一个新的目录，我们可以在其中放置教程代码。我们现在将以下列方式创建一个 JavaConfig 文件（Tut1Config.java）来描述我们的 bean：

```
package com.example.rabbitmq.spring;

import org.springframework.amqp.core.Queue;
import org.springframework.context.annotation.Bean;
```

```java
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Profile;

/**
 * 消息连接配置
 * Author: 王俊超
 * Date: 2017-06-09 07:30
 * All Rights Reserved !!!
 */
@Profile({"tut1", "hello-world"})
@Configuration
public class Tut1Config {
    /**
     * 创建一个消息列队
     * @return
     */
    @Bean
    public Queue hello() {
        return new Queue("hello");
    }

    /**
     * 创建消息接收者
     * @return
     */
    @Profile("receiver")
    @Bean
    public Tut1Receiver receiver() {
        return new Tut1Receiver();
    }

    /**
     * 创建消息发送者
     *
     * @return
     */
    @Profile("sender")
    @Bean
    public Tut1Sender sender() {
        return new Tut1Sender();
    }
}
```

请注意，我们将第一个教程配置文件定义为 tut1，包名称或 com.exapmle.rabbitmq。我们使用 @Configuration 来让 Spring 知道这是一个 Java 配置，在这个类中我们定义了我们的 Queue（“hello”）队列，并定义了我们的 Sender 和 Receiver beans。

我们现在将通过引导应用程序来运行我们所有的教程，只需传递我们使用的配置文件即可。为了实现这一点，我们将使用以下内容修改生成的 RabbitAmqpTutorialsApplication.java：

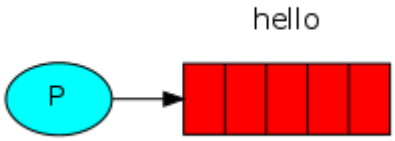
```
package com.example.rabbitmq.spring;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableScheduling;

/**
 * 应用入口类
 *
 * Author: 王俊超
 * Date: 2017-06-09 07:34
 * All Rights Reserved !!!
 */
@SpringBootApplication
@EnableScheduling
public class RabbitAmqpTutorialsApplication {
    public static void main(String[] args) throws Exception {
        SpringApplication.run(RabbitAmqpTutorialsApplication.class, args);
    }
}
```

并添加 RabbitAmqpTutorialsRunner.java 代码如下：

```
package com.example.rabbitmq.spring;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.CommandLineRunner;
import org.springframework.context.ConfigurableApplicationContext;

/**
 * 应用监控器，运行指定时间停止应用的运行
 * Author: 王俊超
 * Date: 2017-06-09 07:37
 * All Rights Reserved !!!
 */
public class RabbitAmqpTutorialsRunner implements CommandLineRunner {

    @Value("${tutorial.client.duration:100000}")
    private int duration;

    @Autowired
    private ConfigurableApplicationContext ctx;
```

```
    @Override
    public void run(String... arg0) throws Exception {
        System.out.println("Ready ... running for " + duration + "ms");
        Thread.sleep(duration);
        ctx.close();
    }
}
```

消息发送



现在发送者和接收者类只需要写少量的代码。我们称之为 Tut1Receiver 和 Tut1Sender。发件人利用我们的配置和 RabbitTemplate 来发送消息。

```
package com.example.rabbitmq.spring;

import org.springframework.amqp.core.Queue;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Scheduled;

/**
 * Author: 王俊超
 * Date: 2017-06-09 07:33
 * All Rights Reserved !!!
 */
public class Tut1Sender {
    @Autowired
    private RabbitTemplate template;

    @Autowired
    private Queue queue;

    @Scheduled(fixedDelay = 1000, initialDelay = 500)
    public void send() {
        String message = "Hello World!";
        this.template.convertAndSend(queue.getName(), message);
        System.out.println(" [x] Sent '" + message + "'");
    }
}
```

你会注意到，spring-amqp 删除了样板代码，只留下了消息传递的逻辑代码。在 Tut1Config 类

中，我们配置的 bean 自动连接消息服务器，并且像许多 spring 连接抽象一样，我们使用可以自动连接到发件人的 RabbitTemplate 来包装样板 rabbitmq 客户端类。剩下的只是创建一个消息，并调用模板的 convertAndSend 方法发送消息，发送消息使用了我们定义的 bean 和刚刚创建的消息队列名称。

接收消息

接收消息同样简单。我们用@RabbitListener 注解我们的 Receiver 类，并传入队列的名称。然后，我们用@RabbitHandler 注释我们的 receive 方法，传递已被推送到队列的有效内容。

```
package com.example.rabbitmq.spring;

import org.springframework.amqp.rabbit.annotation.RabbitHandler;
import org.springframework.amqp.rabbit.annotation.RabbitListener;

/**
 * 消息接收者类，使用@RabbitListener 指定监听的消息通道
 *
 * Author: 王俊超
 * Date: 2017-06-09 07:32
 * All Rights Reserved !!!
 */
@RabbitListener(queues = "hello")
public class Tut1Receiver {
    /**
     * 消息处理方法，使用@RabbitHandler 进行标记
     * @param in
     */
    @RabbitHandler
    public void receive(String in) {
        System.out.println(" [x] Received '" + in + "'");
    }
}
```

运行

先运行接收者，需要添加运行参数：--spring.profiles.active=hello-world,receiver

再运行发送者，需要添加运行参数：--spring.profiles.active=hello-world,sender