15、Spring Boot与消息

2020年6月24日 8:30

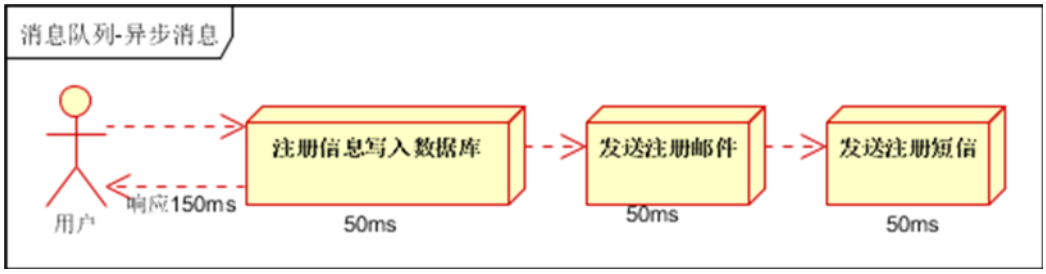
- 1、概述
  a. 大多应用中，可通过消息服务中间件来提升系统异步通信、扩展解耦能力
  b. 消息服务中两个重要概念：

     消息代理（message broker）和目的地（destination）

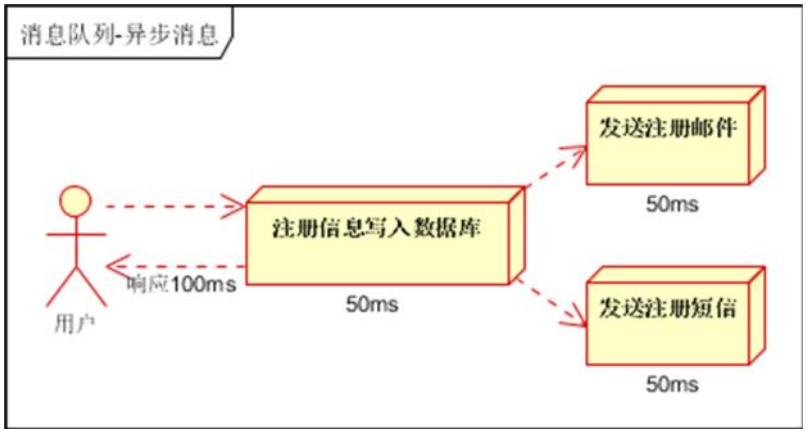
     当消息发送者发送消息以后，将由消息代理接管，消息代理保证消息传递到指定目的地。
  c. 消息队列主要有两种形式的目的地
     1. 队列（queue）：点对点消息通信（point-to-point）
     2. 主题（topic）：发布（publish）/订阅（subscribe）消息通信

     异步处理：

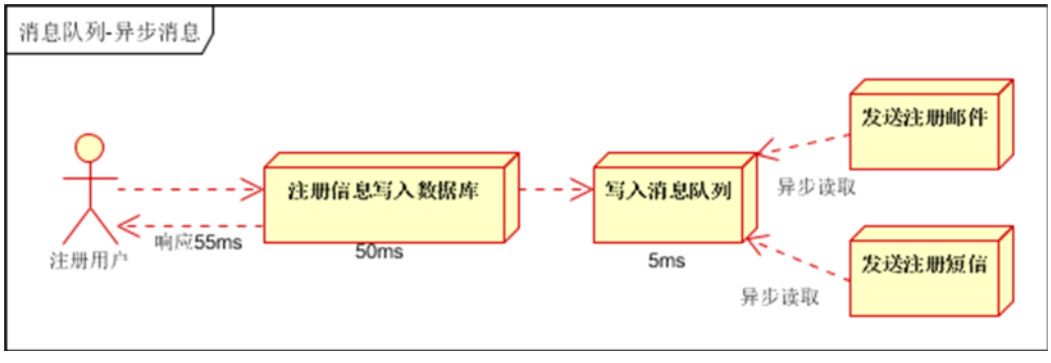
     按逻辑顺序发送，时间过长



     使用多线程时间还是有点长

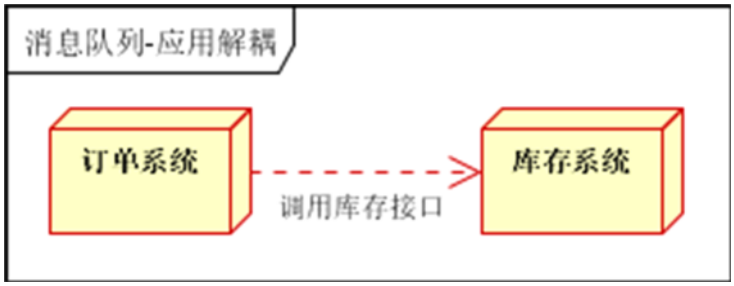


     使用消息队列，当用户数据库逻辑执行完之后直接响应用户并将邮件与短信任务写入消息队列执行异步任务

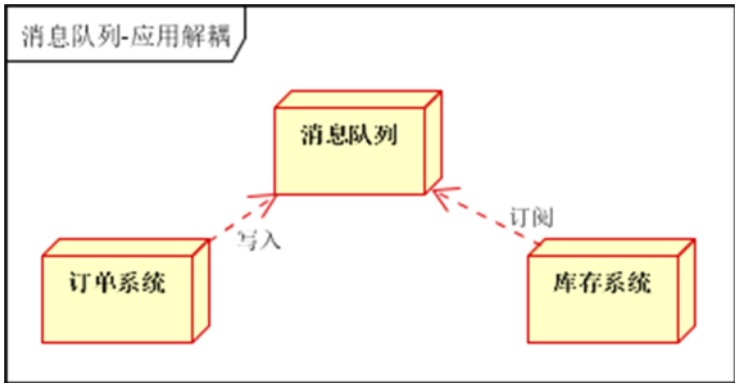


应用解耦：

订单系统直接调用库存接口，耦合性过大



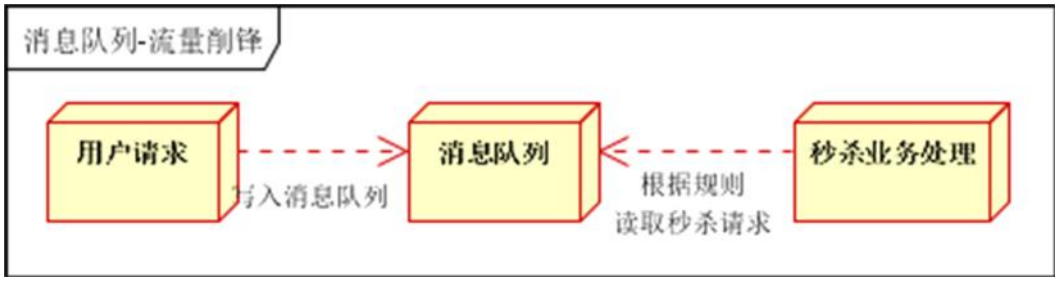
通过消息队列订单系统写入消息队列，库存系统订阅，实现解耦



流量削峰：

秒杀系统瞬间流量巨大，会造成服务器压力过大，可以通过某些规则将用户请求放入消息队列，秒杀业务根据规则读取秒杀请求。

如：只有100个商品，但是请求人数有10000，可以让前一百人直接入队，其余的返回商品已经卖完，秒杀业务只需要处理入队的100个请求就行了。



d. 点对点式：
   - 消息发送者发送消息，消息代理将其放入一个队列中，消息接收者从队列中获取消息内容，消息读取后被移出队列
   - 消息只有唯一的发送者和接受者，但并不是说只能有一个接收者

e. 发布订阅式：
   - 发送者（发布者）发送消息到主题，多个接收者（订阅者）监听（订阅）这个主题，那么就会在消息到达时同时收到消息

f. JMS（Java Message Service）JAVA消息服务：
   - 基于JVM消息代理的规范。ActiveMQ、HornetMQ是JMS实现

g. AMQP（Advanced Message Queuing Protocol）
   - 高级消息队列协议，也是一个消息代理的规范，兼容JMS
   - RabbitMQ是AMQP的实现

	JMS	AMQP
定义	Java api	网络线级协议
跨语言	否	是
跨平台	否	是
Model	提供两种消息模型： （1）、Peer-2-Peer （2）、Pub/sub	提供了五种消息模型： （1）、direct exchange （2）、fanout exchange （3）、topic change （4）、headers exchange （5）、system exchange 本质来讲，后四种和JMS的pub/sub模型没有太大差别，仅是在路由机制上做了更详细的划分；
支持消息类型	多种消息类型： TextMessage MapMessage BytesMessage StreamMessage ObjectMessage Message （只有消息头和属性）	byte[] 当实际应用时，有复杂的消息，可以将消息序列化后发送。
综合评价	JMS 定义了JAVA API层面的标准；在java体系中，多个client均可以通过JMS进行交互，不需要应用修改代码，但是其对跨平台的支持较差；	AMQP定义了wire-level层的协议标准；天然具有跨平台、跨语言特性。

h. Spring支持
   - spring-jms提供了对JMS的支持
   - spring-rabbit提供了对AMQP的支持
   - 需要ConnectionFactory的实现来连接消息代理
   - 提供JmsTemplate、RabbitTemplate来发送消息
   - @JmsListener（JMS）、@RabbitListener（AMQP）注解在方法上监听消息代理发布的消息
   - @EnableJms、@EnableRabbit开启支持

i. Spring Boot自动配置
   - JmsAutoConfiguration

        - RabbitAutoConfiguration
- 2、RabbitMQ简介
    - RabbitMQ简介：

      RabbitMQ是一个由erlang开发的AMQP(Advanved Message Queue Protocol)的开源实现。
    - 核心概念
    - Message

      消息，消息是不具名的，它由消息头和消息体组成。消息体是不透明的，而消息头则由一系列的可选属性组成，这些属性包括routing-key（路由键）、priority（相对于其他消息的优先权）、delivery-mode（指出该消息可能需要持久性存储）等。
    - Publisher

      消息的生产者，也是一个向交换器发布消息的客户端应用程序。
    - Exchange

      交换器，用来接收生产者发送的消息并将这些消息路由给服务器中的队列。

      Exchange有4种类型：direct(默认)，fanout, topic, 和headers，不同类型的Exchange转发消息的策略有所区别
    - Queue

      消息队列，用来保存消息直到发送给消费者。它是消息的容器，也是消息的终点。一个消息可投入一个或多个队列。消息一直在队列里面，等待消费者连接到这个队列将其取走。
    - Binding

      绑定，用于消息队列和交换器之间的关联。一个绑定就是基于路由键将交换器和消息队列连接起来的路由规则，所以可以将交换器理解成一个由绑定构成的路由表。

      Exchange 和Queue的绑定可以是多对多的关系。
    - Connection

      网络连接，比如一个TCP连接。
    - Channel

      信道，多路复用连接中的一条独立的双向数据流通道。信道是建立在真实的TCP连接内的虚拟连接，AMQP 命令都是通过信道发出去的，不管是发布消息、订阅队列还是接收消息，这些动作都是通过信道完成。因为对于操作系统来说建立和销毁 TCP 都是非常昂贵的开销，所以引入了信道的概念，以复用一条 TCP 连接。
    - Consumer

      消息的消费者，表示一个从消息队列中取得消息的客户端应用程序。
    - Virtual Host

      虚拟主机，表示一批交换器、消息队列和相关对象。虚拟主机是共享相同的身份认证和加密环境的独立服务器域。每个 vhost 本质上就是一个 mini 版的 RabbitMQ 服务器，拥有自己的队列、交换器、绑定和权限机制。vhost 是 AMQP 概念的基础，必须

在连接时指定，RabbitMQ 默认的 vhost 是 / 。

- Broker

  表示消息队列服务器实体

- 3、RabbitMQ运行机制
  - AMQP 中的消息路由

    AMQP 中消息的路由过程和 Java 开发者熟悉的 JMS 存在一些差别，AMQP 中增加了 Exchange 和 Binding 的角色。生产者把消息发布到 Exchange 上，消息最终到达队列并被消费者接收，而 Binding 决定交换器的消息应该发送到那个队列。

  - Exchange分发消息时根据类型的不同分发策略有区别，目前共四种类型：direct、fanout、topic、headers 。headers 匹配 AMQP 消息的 header 而不是路由键， headers 交换器和 direct 交换器完全一致，但性能差很多，目前几乎用不到了，所以直接看另外三种类型：
    - 消息中的路由键（routing key）如果和 Binding 中的 binding key 一致， 交换器就将消息发到对应的队列中。路由键与队列名完全匹配，如果一个队列绑定到交换机要求路由键为“dog”，则只转发 routing key 标记为“dog”的消息，不会转发“dog.puppy”，也不会转发“dog.guard”等等。它是完全匹配、单播的模式。
    - 每个发到 fanout 类型交换器的消息都会分到所有绑定的队列上去。fanout 交换器不处理路由键，只是简单的将队列绑定到交换器上，每个发送到交换器的消息都会被转发到与该交换器绑定的所有队列上。很像子网广播，每台子网内的主机都获得了一份复制的消息。fanout 类型转发消息是最快的。
    - topic 交换器通过模式匹配分配消息的路由键属性，将路由键和某个模式进行匹配，此时队列需要绑定到一个模式上。它将路由键和绑定键的字符串切分成单词，这些单词之间用点隔开。它同样也会识别两个通配符：符号“#”和符号“*”。#匹配0个或多个单词，*匹配一个单词。
- 4、docker安装RabbitMQ
  - 1、拉取镜像

    docker pull rabbitmq:3.7.26-management
  - 2、端口映射，启动

    docker run -d -p 5672:5672 -p 15672:15672 --name myrabbitmq f6ea23d4f7e4
  - 3、访问15672端口进入管理页面
- 5、RabbitMQ整合
  - 1、引入 spring-boot-starter-amqp

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

  - 2、application.properties配置

```
spring.rabbitmq.host=39.101.170.233
spring.rabbitmq.username=guest
spring.rabbitmq.password=guest
```

#spring.rabbitmq.virtual-host=

- 3、自动配置
  - 1、RabbitAutoConfiguration
  - 2、自动配置了连接工厂ConnectionFactory
  - 3、RabbitProperties封装了RabbitMQ的配置
  - 4、RabbitTemplate：给RabbitMQ发送和接受消息
  - 5、AmqpAdmin：RabbitMQ系统管理功能组件
  - 6、@EnableRabbit+@RabbitListener监听消息队列的内容

```
@EnableRabbit//开启基于注解的RabbitMQ模式
@SpringBootApplication
public class Springboot02AmqpApplication {
        public static void main(String[] args) {
                SpringApplication.run(Springboot02AmqpApplication.class,
args);
        }
}
service层
@Service
public class BookService {
        @RabbitListener(queues = "atguigu.news")
        public void receive(Book book){
                System.out.println("收到消息"+book);
        }
        @RabbitListener(queues = "atguigu")
        public void receive02(Message message){
                System.out.println(message.getBody());
                System.out.println(message.getMessageProperties());
        }
}
```

- 4、测试RabbitMQ
  - 1、单播（点对点）

```
@Test
void contextLoads() {
        //Message需要自己构造一个；定义消息体内容和消息头
        //rabbitTemplate.send(exchange,routeKey,message);
        //只需要传入要发送的对象，自动序列化发送给rabbitmq
        //rabbitTemplate.convertAndSend(exchange,routeKey,object);
        //Map<String,Object> map = new HashMap<>();
        //map.put("msg","这是第一个消息");
        //map.put("data", Arrays.asList("helloworld",123,true));
        //对象被默认序列化以后发送出去
```

```
        rabbitTemplate.
                convertAndSend("exchange.direct","atguigu.news",new
                Book("西游记","吴承恩"));
    }
    //接受数据，如何将数据转为json
    @Test
    public void receive(){
        Book book = (Book) rabbitTemplate.
                                receiveAndConvert("atguigu.news");
        System.out.println(book);
    }
```

编写MyAMQPConfig类，返回Jackson2JsonMessageConverter即可返回json

```
    @Configuration
    public class MyAMQPConfig {
        @Bean
        public MessageConverter messageConverter(){
            return new Jackson2JsonMessageConverter();
        }
    }
```

- 2、广播

```
    /*
    * 广播
    * */
    @Test
    public void sendMsg(){
        rabbitTemplate.
                convertAndSend("exchange.fanout","",
                new Book("红楼梦","曹雪芹"));
    }
```