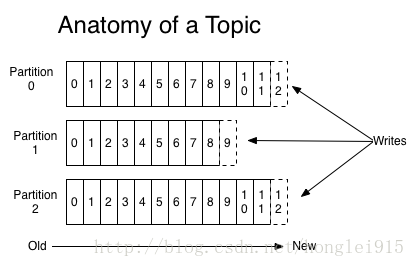
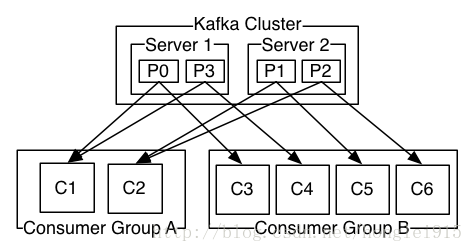
kafka

介绍

Kafka是一个分布式的、可分区的、可复制的消息系统。它提供了普通消息系统的功能，但具有自己独特的设计。

这个独特的设计是什么样的呢？  
 首先让我们看几个基本的消息系统术语：  
Kafka将消息以topic为单位进行归纳。  
将向Kafka topic发布消息的程序成为producers.  
将预订topics并消费消息的程序成为consumer.  
Kafka以集群的方式运行，可以由一个或多个服务组成，每个服务叫做一个broker.  
producers通过网络将消息发送到Kafka集群，集群向消费者提供消息，如下图所示：  
  
   
  
客户端和服务端通过TCP协议通信。Kafka提供了Java客户端，并且对多种语言都提供了支持。

Topics 和Logs  
先来看一下Kafka提供的一个抽象概念:topic.  
一个topic是对一组消息的归纳。对每个topic，Kafka 对它的日志进行了分区，如下图所示：  
   
  
每个分区都由一系列有序的、不可变的消息组成，这些消息被连续的追加到分区中。分区中的每个消息都有一个连续的序列号叫做offset,用来在分区中唯一的标识这个消息。  
在一个可配置的时间段内，Kafka集群保留所有发布的消息，不管这些消息有没有被消费。比如，如果消息的保存策略被设置为2天，那么在一个消息被发布的两天时间内，它都是可以被消费的。之后它将被丢弃以释放空间。Kafka的性能是和数据量无关的常量级的，所以保留太多的数据并不是问题。  
  
实际上每个consumer唯一需要维护的数据是消息在日志中的位置，也就是offset.这个offset有consumer来维护：一般情况下随着consumer不断的读取消息，这offset的值不断增加，但其实consumer可以以任意的顺序读取消息，比如它可以将offset设置成为一个旧的值来重读之前的消息。  
  
以上特点的结合，使Kafka consumers非常的轻量级：它们可以在不对集群和其他consumer造成影响的情况下读取消息。你可以使用命令行来"tail"消息而不会对其他正在消费消息的consumer造成影响。  
  
将日志分区可以达到以下目的：首先这使得每个日志的数量不会太大，可以在单个服务上保存。另外每个分区可以单独发布和消费，为并发操作topic提供了一种可能。  
  
****分布式****  
  
每个分区在Kafka集群的若干服务中都有副本，这样这些持有副本的服务可以共同处理数据和请求，副本数量是可以配置的。副本使Kafka具备了容错能力。  
每个分区都由一个服务器作为“leader”，零或若干服务器作为“followers”,leader负责处理消息的读和写，followers则去复制leader.如果leader down了，followers中的一台则会自动成为leader。集群中的每个服务都会同时扮演两个角色：作为它所持有的一部分分区的leader，同时作为其他分区的followers，这样集群就会据有较好的负载均衡。  
  
****Producers****  
  
Producer将消息发布到它指定的topic中,并负责决定发布到哪个分区。通常简单的由负载均衡机制随机选择分区，但也可以通过特定的分区函数选择分区。使用的更多的是第二种。  
  
  
****Consumers****  
  
发布消息通常有两种模式：队列模式（queuing）和发布-订阅模式(publish-subscribe)。队列模式中，consumers可以同时从服务端读取消息，每个消息只被其中一个consumer读到；发布-订阅模式中消息被广播到所有的consumer中。Consumers可以加入一个consumer 组，共同竞争一个topic，topic中的消息将被分发到组中的一个成员中。同一组中的consumer可以在不同的程序中，也可以在不同的机器上。如果所有的consumer都在一个组中，这就成为了传统的队列模式，在各consumer中实现负载均衡。如果所有的consumer都不在不同的组中，这就成为了发布-订阅模式，所有的消息都被分发到所有的consumer中。更常见的是，每个topic都有若干数量的consumer组，每个组都是一个逻辑上的“订阅者”，为了容错和更好的稳定性，每个组由若干consumer组成。这其实就是一个发布-订阅模式，只不过订阅者是个组而不是单个consumer。  
  
   
  
由两个机器组成的集群拥有4个分区 (P0-P3) 2个consumer组. A组有两个consumerB组有4个  
  
相比传统的消息系统，Kafka可以很好的保证有序性。  
传统的队列在服务器上保存有序的消息，如果多个consumers同时从这个服务器消费消息，服务器就会以消息存储的顺序向consumer分发消息。虽然服务器按顺序发布消息，但是消息是被异步的分发到各consumer上，所以当消息到达时可能已经失去了原来的顺序，这意味着并发消费将导致顺序错乱。为了避免故障，这样的消息系统通常使用“专用consumer”的概念，其实就是只允许一个消费者消费消息，当然这就意味着失去了并发性。  
  
在这方面Kafka做的更好，通过分区的概念，Kafka可以在多个consumer组并发的情况下提供较好的有序性和负载均衡。将每个分区分只分发给一个consumer组，这样一个分区就只被这个组的一个consumer消费，就可以顺序的消费这个分区的消息。因为有多个分区，依然可以在多个consumer组之间进行负载均衡。注意consumer组的数量不能多于分区的数量，也就是有多少分区就允许多少并发消费。  
  
Kafka只能保证一个分区之内消息的有序性，在不同的分区之间是不可以的，这已经可以满足大部分应用的需求。如果需要topic中所有消息的有序性，那就只能让这个topic只有一个分区，当然也就只有一个consumer组消费它。

****二、环境搭建****  
  
****Step 1: 下载Kafka****   
  
下载kafka并解压.



1. > tar -xzf kafka\_2.9.2-0.8.1.1.tgz
2. > cd kafka\_2.9.2-0.8.1.1

****Step 2: 启动服务****  
  
Kafka用到了Zookeeper，所有首先启动Zookper，下面简单的启用一个单实例的Zookkeeper服务。可以在命令的结尾加个&符号，这样就可以启动后离开控制台。

1. > bin/zookeeper-server-start.sh config/zookeeper.properties &
2. [2013-04-22 15:01:37,495] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
3. ...

现在启动Kafka:

1. > bin/kafka-server-start.sh config/server.properties
2. [2013-04-22 15:01:47,028] INFO Verifying properties (kafka.utils.VerifiableProperties)
3. [2013-04-22 15:01:47,051] INFO Property socket.send.buffer.bytes is overridden to 1048576 (kafka.utils.VerifiableProperties)
4. ...

提示:

错误: 找不到或无法加载主类 ，请在kafka主目录下执行

./gradlew jar

****Step 3: 创建 topic****  
  
创建一个叫做“test”的topic，它只有一个分区，一个副本。

1. > bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test

可以通过list命令查看创建的topic:

1. > bin/kafka-topics.sh --list --zookeeper localhost:2181
2. test

除了手动创建topic，还可以配置broker让它自动创建topic.  
  
****Step 4:发送消息.****  
  
Kafka 使用一个简单的命令行producer，从文件中或者从标准输入中读取消息并发送到服务端。默认的每条命令将发送一条消息。  
  
运行producer并在控制台中输一些消息，这些消息将被发送到服务端：

1. > bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test
2. This is a messageThis is another message

ctrl+c可以退出发送。  
  
****Step 5: 启动consumer****  
  
Kafka also has a command line consumer that will dump out messages to standard output.  
Kafka也有一个命令行consumer可以读取消息并输出到标准输出：

1. > bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning
2. This is a message
3. This is another message

你在一个终端中运行consumer命令行，另一个终端中运行producer命令行，就可以在一个终端输入消息，另一个终端读取消息。  
这两个命令都有自己的可选参数，可以在运行的时候不加任何参数可以看到帮助信息。  
  
****Step 6: 搭建一个多个broker的集群****  
  
刚才只是启动了单个broker，现在启动有3个broker组成的集群，这些broker节点也都是在本机上的：  
首先为每个节点编写配置文件：

1. > cp config/server.properties config/server-1.properties
2. > cp config/server.properties config/server-2.properties

在拷贝出的新文件中添加以下参数：

1. config/server-1.properties:
2. broker.id=1
3. port=9093
4. log.dir=/tmp/kafka-logs-1
5. config/server-2.properties:
6. broker.id=2
7. port=9094
8. log.dir=/tmp/kafka-logs-2

broker.id在集群中唯一的标注一个节点，因为在同一个机器上，所以必须制定不同的端口和日志文件，避免数据被覆盖。  
  
We already have Zookeeper and our single node started, so we just need to start the two new nodes:  
刚才已经启动可Zookeeper和一个节点，现在启动另外两个节点：

1. > bin/kafka-server-start.sh config/server-1.properties &
2. ...
3. > bin/kafka-server-start.sh config/server-2.properties &
4. ...

创建一个拥有3个副本的topic:

1. > bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic my-replicated-topic

现在我们搭建了一个集群，怎么知道每个节点的信息呢？运行“"describe topics”命令就可以了：

1. > bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic my-replicated-topic
2. Topic:my-replicated-topic       PartitionCount:1        ReplicationFactor:3     Configs:
3. Topic: my-replicated-topic      Partition: 0    Leader: 1       Replicas: 1,2,0 Isr: 1,2,0

下面解释一下这些输出。第一行是对所有分区的一个描述，然后每个分区都会对应一行，因为我们只有一个分区所以下面就只加了一行。  
leader：负责处理消息的读和写，leader是从所有节点中随机选择的.  
replicas：列出了所有的副本节点，不管节点是否在服务中.  
isr：是正在服务中的节点.  
在我们的例子中，节点1是作为leader运行。  
向topic发送消息：

1. > bin/kafka-console-producer.sh --broker-list localhost:9092 --topic my-replicated-topic
2. ...
3. my test message 1my test message 2^C

消费这些消息：

1. > bin/kafka-console-consumer.sh --zookeeper localhost:2181 --from-beginning --topic my-replicated-topic
2. ...
3. my test message 1
4. my test message 2
5. ^C

测试一下容错能力.Broker 1作为leader运行，现在我们kill掉它：

1. > ps | grep server-1.properties7564 ttys002    0:15.91 /System/Library/Frameworks/JavaVM.framework/Versions/1.6/Home/bin/java...
2. > kill -9 7564

另外一个节点被选做了leader,node 1 不再出现在 in-sync 副本列表中：

1. > bin/kafka-topics.sh --describe --zookeeper localhost:218192 --topic my-replicated-topic
2. Topic:my-replicated-topic       PartitionCount:1        ReplicationFactor:3     Configs:
3. Topic: my-replicated-topic      Partition: 0    Leader: 2       Replicas: 1,2,0 Isr: 2,0

虽然最初负责续写消息的leader down掉了，但之前的消息还是可以消费的：

1. > bin/kafka-console-consumer.sh --zookeeper localhost:2181 --from-beginning --topic my-replicated-topic
2. ...
3. my test message 1
4. my test message 2
5. **搭建Kafka开发环境**  
   我们搭建了kafka的服务器，并可以使用Kafka的命令行工具创建topic，发送和接收消息。下面我们来搭建kafka的开发环境。  
   1.添加依赖

<dependency>

<groupId>orj.worf</groupId>

<artifactId>worf-kafka</artifactId>

<version>1.0-SNAPSHOT</version>

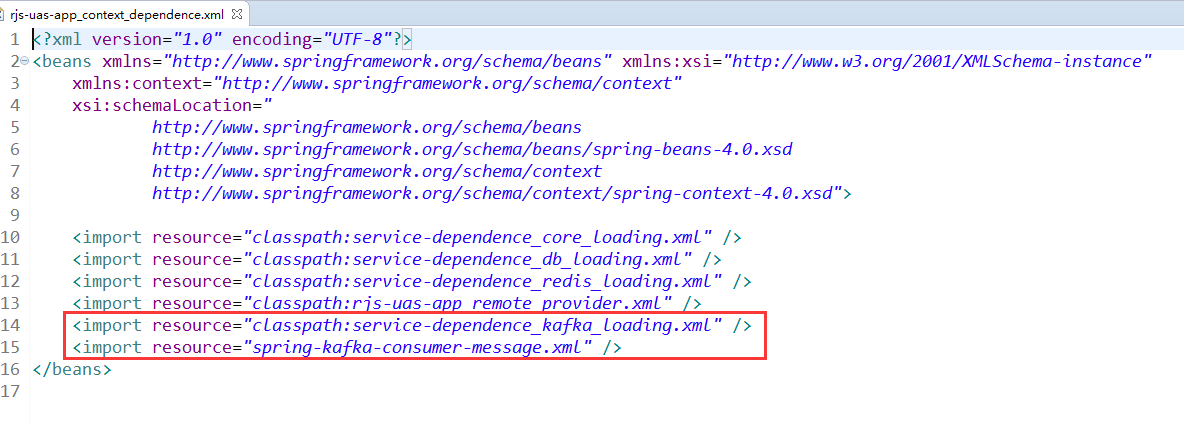
</dependency>

1. 添加配置文件

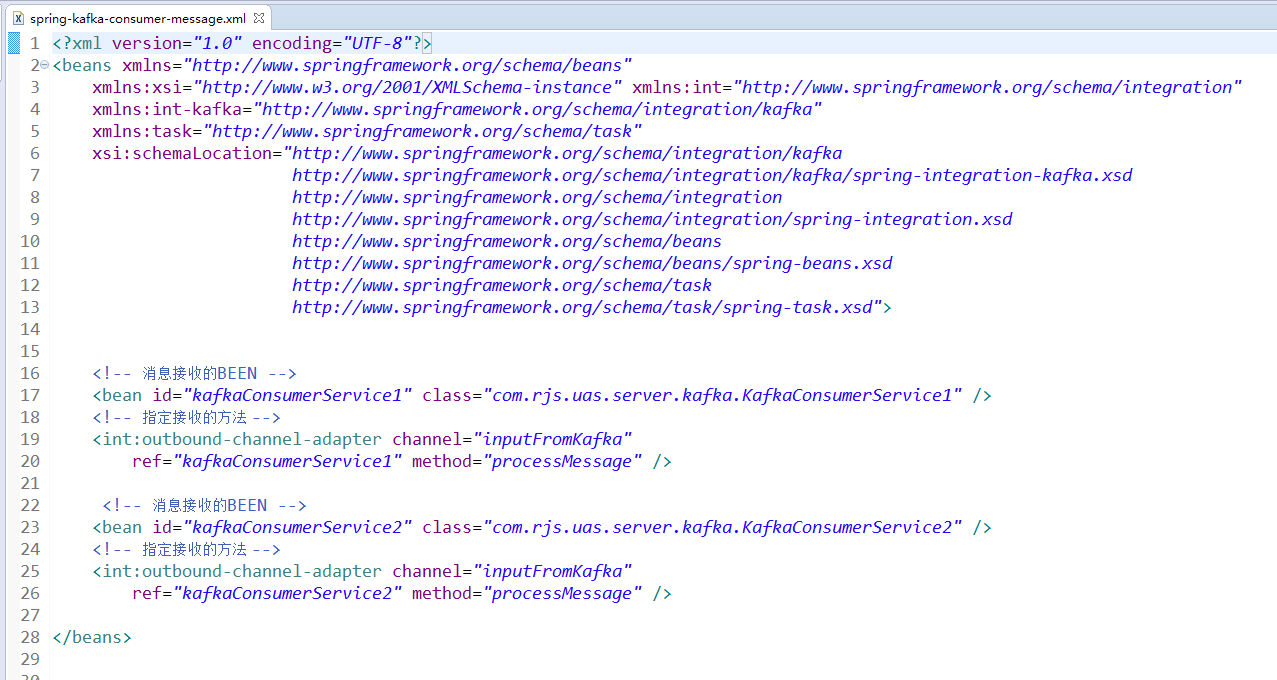
在加载spring配置文件中加入

<import resource=*"classpath:service-dependence\_kafka\_loading.xml"* />

<import resource=*"spring-kafka-consumer-message.xml"* />

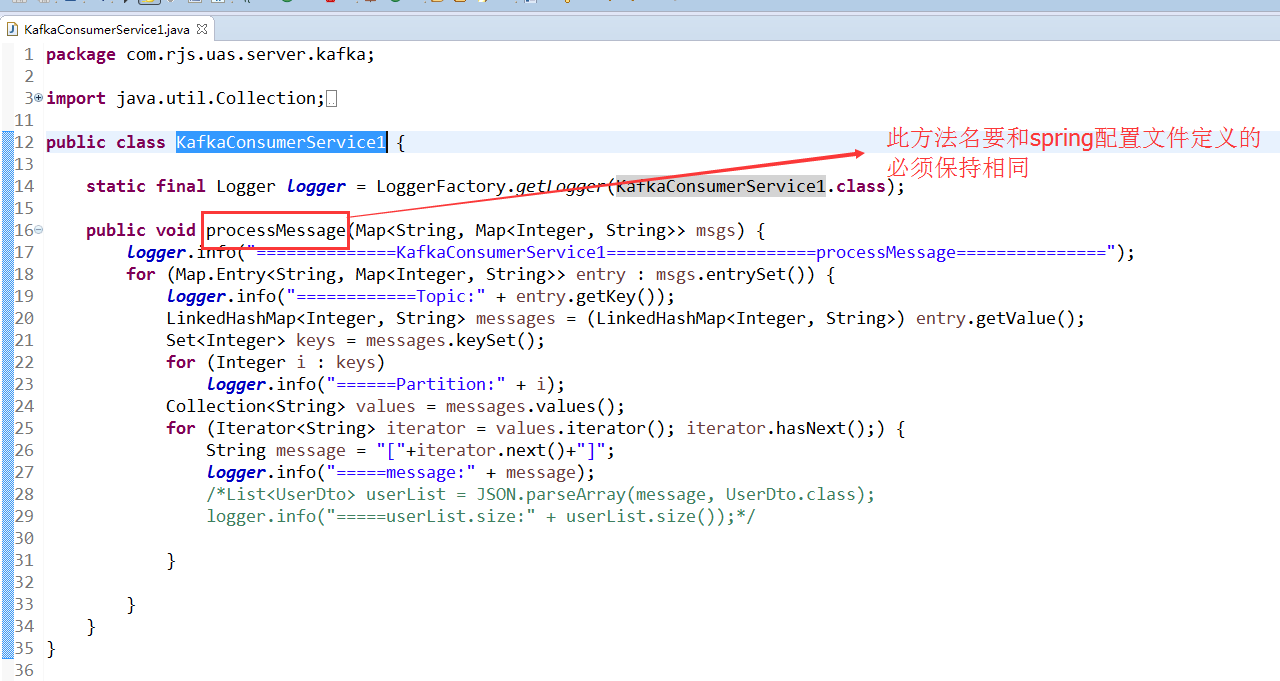


其中 *spring-kafka-consumer-message.xml*  需要自己手动添加，内容如下：





*com.rjs.uas.server.kafka.KafkaConsumerService1中类KafkaConsumerService1内容如下：*





在 src/mian/resources/conf 目录下添加 kafka.properties 属性文件

kafka.properties 内容如下：

#支持多个zk

kafka.zookeeper.registry=172.16.88.49:2181

#topic1 如果只有一个topic，两个topic的内容请保持一致，默认读取kafka.topic1 的topic

kafka.topic1=my-replicated-topic

kafka.broker.list1=172.16.88.49:9091,172.16.88.49:9092,172.16.88.49:9093

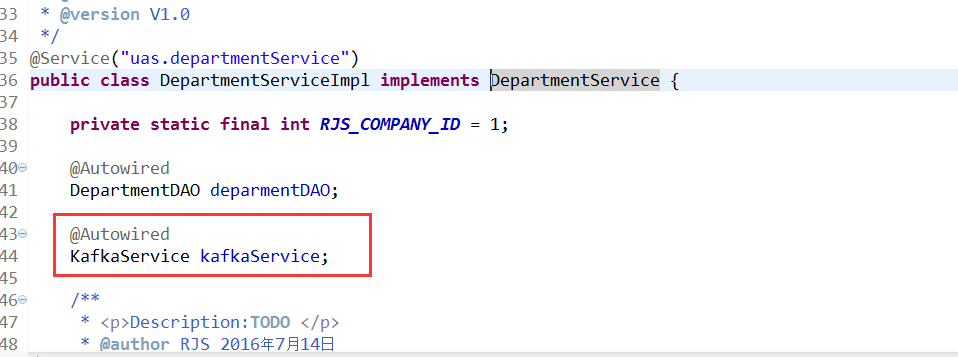
#topic2

kafka.topic2=my-replicated-topic

kafka.broker.list2=172.16.88.49:9091,172.16.88.49:9092,172.16.88.49:9093

这样，服务启动的时候，kafka就会在后台保持运行，如果生产者发送消息，后台就会进入到上述的 processManage 方法中，接收消息。

那么，我们如何让生产者发送消息呢？



通过Autowired 注入 kafkaService ，然后调用 sendUserInfo 方法

