图文 100 BrokerController是如何构建出来的，以及他包含了哪些组件?

详情 评论

BrokerController是如何构建出来的，以及他包含了哪些组件?

继《从零开始带你成为JVM实战高手》后，阿里资深技术专家携新作再度出山，重磅推荐：

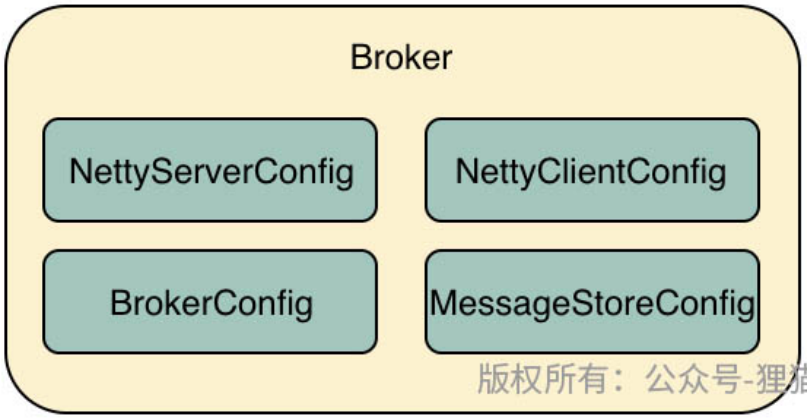
（点击下方蓝字试听）

《从零开始带你成为MySQL实战优化高手》

1、BrokerController是在哪里创建出来的?

上回我们讲到Broker在启动的时候，他首先会执行一个关键的方法，就是createBrokerController()

在这个里面，我们之前分析了他已经初始化以及解析了一些核心配置组件，我们看看下面的图，你会发现Broker里已经有一些核心配置组件了。



上述的核心配置组件，上一次我们也分析过了，其实这类配置初始化的代码就是看着麻烦，并没什么难度的，他本质上就是用默认的配置参数值以及我们配置文件里的配置参数值，包括命令行传递的配置参数值，去填充到这些配置组件中去。

然后后续你Broker运行的过程中，各种行为自然都是根据这些配置组件里的配置参数值来走的，大概就是这个意思，所以大家千万别对源码感到恐慌。

我们接着看，那么在准备好了上述核心配置组件之后，接下来下一步是干什么呢？

其实就是要创建最核心的Broker组件了，这个Broker组件是如何创建的呢？

我们看下面的源码，下面的源码就是在createBrokerController()方法中的。

```
final BrokerController controller = new BrokerController(
  brokerConfig,
  nettyServerConfig,
  nettyClientConfig,
  messageStoreConfig);

controller.getConfiguration().registerConfig(properties);
```

上述代码的意思已经非常明显了，他就是创建了一个核心的BrokerController组件，这个BrokerController组件，你大致可以认为就是代表了Broker他自己好了。我们可以来梳理一下BrokerStartup和BrokerController两个代码组件的关系，以及为什么要这么设计。

2、为什么要叫做“BrokerController”呢？

首先我们来思考一下，Broker启动的时候，他的main class是谁？是不是BrokerStartup这个类？

那么这个类名，顾名思义，就是用来启动Broker的一个类，他里面包含的是把Broker给进行初始化和完成全部启动工作的逻辑。

所以，大家觉得BrokerStartup自己可以代表成是一个Broker吗？明显是不对的。

所以其实最核心的组件，就是BrokerController，这个BrokerController可能有的朋友会以为跟我们平时在Java Web开发中，用Spring MVC框架开发的一系列Controller是一个意思，负责接收和处理请求。

其实这么想，也有一定道理，但是也不完全是对的。

因为毕竟中间件系统的架构设计思想和普通的Java Web业务系统还是不一样的。虽然两种Contorller的含义是相近的，但还是有区别。

BrokerController，如果一定要用中文来表达出这个组件的含义的话，你应该把他叫做是“Broker管理控制组件”

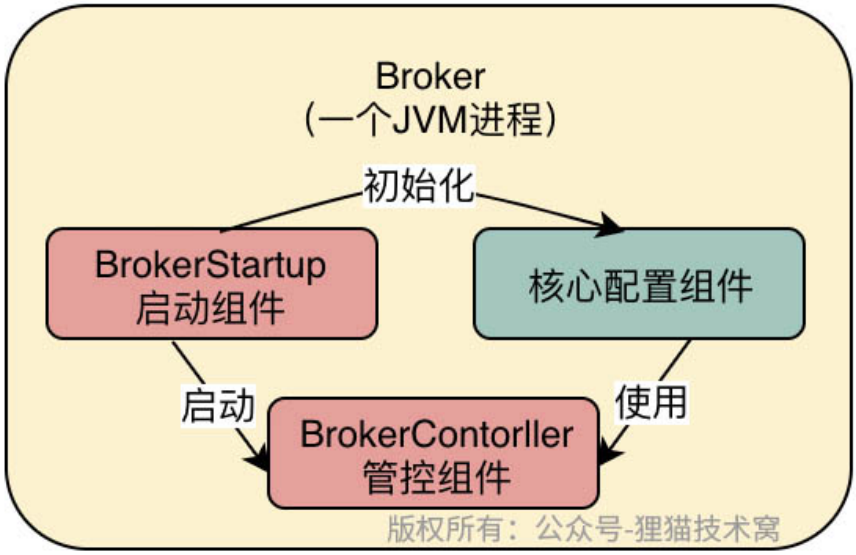
什么意思呢？他这个组件其实被创建出来以及初始化完毕之后，就是用来控制当前在运行的这个Broker的！

所以他英文叫做BrokerController，中文叫做“Broker管理控制组件”。

也正是因为如此，你大致可以理解一下他们这里的关系，我们用mqbroker脚本启动的JVM进程，实际上你可以认为就是一个Broker，这里Broker实际上应该是代表了一个JVM进程的概念，而不是任何一个代码组件！

然后BrokerStartup作为一个main class，其实是属于一个代码组件，他的作用是准备好核心配置组件，然后就是创建、初始化以及启动BrokerController这个核心组件，也就是启动一个Broker管理控制组件，让BrokerController去控制和管理Broker这个JVM进程运行过程中的一切行为，包括接收网络请求、包括管理磁盘上的消息数据，以及一大堆的后台线程的运行。

所以讲到这里，我又完善了一下下面的图，大家在里面可以清晰的看到Broker、BrokerStartup、BrokerController之间的关系。



上面那个图里，其实就把这几者之间的关系，说的很清晰了，Broker这个概念本身代表的不是一个代码组件，他就是你用mqbroker脚本启动的JVM进程。然后JVM进程的main class是BrokerStartup，他是一个启动组件，负责初始化核心配置组件，然后基于核心配置组件去启动BrokerControler这个管控组件。

然后在Broker这个JVM进程运行期间，都是由BrokerController这个管控组件去管理Broker的请求处理、后台线程以及磁盘数据。

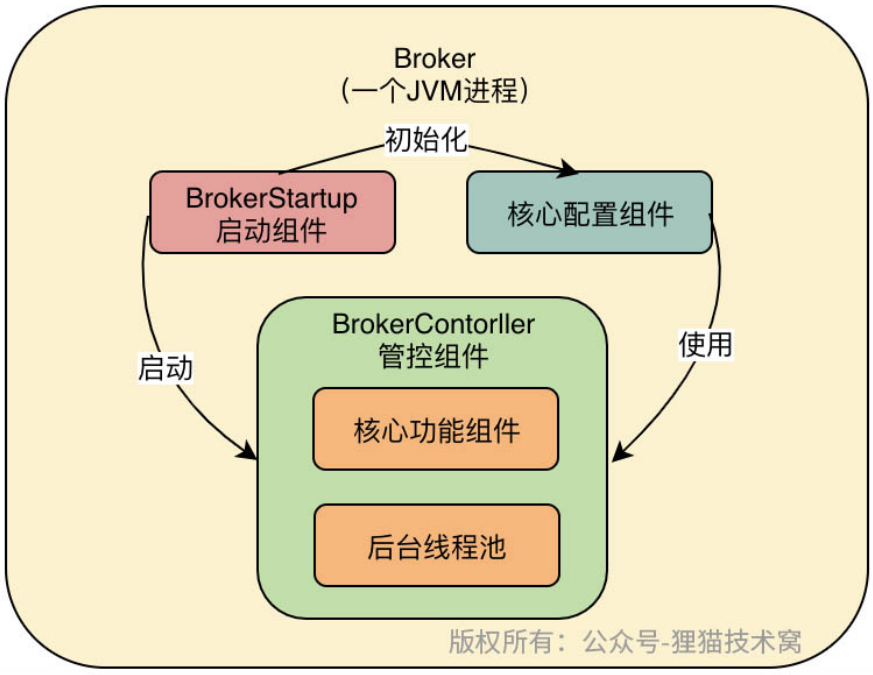
3、初步看一下BrokerController的构造函数

接着我们初步的看一下BrokerController的构造函数，如下所示，我在里面简单给大家写了一些注释。

```
// 下面这4行代码很简单，把4个核心配置组件自己保存了一下
this.brokerConfig = brokerConfig;
this.nettyServerConfig = nettyServerConfig;
this.nettyClientConfig = nettyClientConfig;
this.messageStoreConfig = messageStoreConfig;

// 下面这一大堆，其实不用多说，也大概能看出来都是Broker的各种功能对应的组件
// 比如说这个ConsumerOffsetManager，一看就知道是专门管理consumer消费offset的
// 还有TopicConfigManager，一看就知道是管理Topic配置的
// 还有PullMessageProcessor，一看就知道是处理Consumer发送请求过来拉取消息的
// 所以你大概可以认为，Broker会有很多很多的功能，每个功能都是一个组件来负责的
// 于是乎，Broker在初始化的时候，内部就会有一大堆的代码组件
this.consumerOffsetManager = new ConsumerOffsetManager(this);
this.topicConfigManager = new TopicConfigManager(this);
this.pullMessageProcessor = new PullMessageProcessor(this);
this.pullRequestHoldService = new PullRequestHoldService(this);
this.messageArrivingListener = new NotifyMessageArrivingListener(
                                    this.pullRequestHoldService);

this.consumerIdsChangeListener = new DefaultConsumerIdsChangeListener(this);
this.consumerManager = new ConsumerManager(this.consumerIdsChangeListener);

this.consumerFilterManager = new ConsumerFilterManager(this);
this.producerManager = new ProducerManager();
this.clientHousekeepingService = new ClientHousekeepingService(this);
this.broker2Client = new Broker2Client(this);
this.subscriptionGroupManager = new SubscriptionGroupManager(this);
this.brokerOuterAPI = new BrokerOuterAPI(nettyClientConfig);
this.filterServerManager = new FilterServerManager(this);
this.slaveSynchronize = new SlaveSynchronize(this);

// 下面这里是一大堆的线程池的队列
// 说白了，就是用来实现某些功能的后台线程池的队列
// 不同的后台线程和处理请求的线程放在不同的线程池里去执行，大概就是这个意思
// 因为有些Broker的功能是你发送请求给他，他来进行处理的，会用到上面的一些组件
// 有些Broker的功能，实际上是自己的后台线程去执行的
this.sendThreadPoolQueue = new LinkedBlockingQueue<Runnable>(
    this.brokerConfig.getSendThreadPoolQueueCapacity());

this.pullThreadPoolQueue = new LinkedBlockingQueue<Runnable>(
    this.brokerConfig.getPullThreadPoolQueueCapacity());

this.replyThreadPoolQueue = new LinkedBlockingQueue<Runnable>(
    this.brokerConfig.getReplyThreadPoolQueueCapacity());

this.queryThreadPoolQueue = new LinkedBlockingQueue<Runnable>(
    this.brokerConfig.getQueryThreadPoolQueueCapacity());

this.clientManagerThreadPoolQueue = new LinkedBlockingQueue<Runnable>(
    this.brokerConfig.getClientManagerThreadPoolQueueCapacity());

this.consumerManagerThreadPoolQueue = new LinkedBlockingQueue<Runnable>(
    this.brokerConfig.getConsumerManagerThreadPoolQueueCapacity());

this.heartbeatThreadPoolQueue = new LinkedBlockingQueue<Runnable>(
    this.brokerConfig.getHeartbeatThreadPoolQueueCapacity());

this.endTransactionThreadPoolQueue = new LinkedBlockingQueue<Runnable>(
    this.brokerConfig.getEndTransactionPoolQueueCapacity());
```

```
60
61 // 下面这些同样也是Broker的一些功能性组件，比如StatsManager一看就是
62 // metric统计组件，就是对Broker内部进行统计的
63 // 还有比如BrokerFastFailure一看就是用于处理Broker故障的组件
64 this.brokerStatsManager = new BrokerStatsManager(
65     this.brokerConfig.getBrokerClusterName());
66
67 this.setStoreHost(new InetSocketAddress(
68     this.getBrokerConfig().getBrokerIP1(),
69     this.getNettyServerConfig().getListenPort()));
70
71 this.brokerFastFailure = new BrokerFastFailure(this);
72 this.configuration = new Configuration(
73     log,
74     BrokerPathConfigHelper.getBrokerConfigPath(),
75     this.brokerConfig,
76     this.nettyServerConfig,
77     this.nettyClientConfig,
78     this.messageStoreConfig
79 );
```

简单来说，看完了上述的源码之后，其实大家没必要立马搞定每一个代码组件的含义，以及他们是做什么的，主要知道BrokerController内部是有一系列的功能性组件的，还有一大堆的后台线程池，知道这两点就可以了，如下图。



大家在上图可以看到，我又在图里进行了完善，让你能感受到，BrokerController里包含了一大堆核心功能组件和后台线程池，至于这些功能组件和后台线程池都是干什么的，不是你现在就要去探究的，后续我们要基于一个一个的场景出发，去研发他里面的各种代码组件是如何工作的。

4、今日源码分析作业

今天给大家留一个源码分析的小作业，希望每个人都可以去看一下BrokerController的构造函数里的内容，自己感受一下，这个里面包含了很多功能组件以及后台线程池，感受一下他的源码是怎么来写的，另外去感受一下Broker这个JVM进程，BrokerStartup启动组件，BrokerController管控组件之间的关系。

如果在分析源码的时候有什么心得，可以发布在评论区跟大家一起交流。

End

狸猫技术窝精品专栏及课程推荐：

- 《从零开始带你成为JVM实战高手》
- 《21天互联网Java进阶面试训练营》（分布式篇）
- 《互联网Java工程师面试突击》（第1季）
- 《互联网Java工程师面试突击》（第3季）

重要说明：

如何提问：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑

如何加群：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

具体加群方式，请参见目录菜单下的文档：《付费用户如何加群》（**购买后可见**）