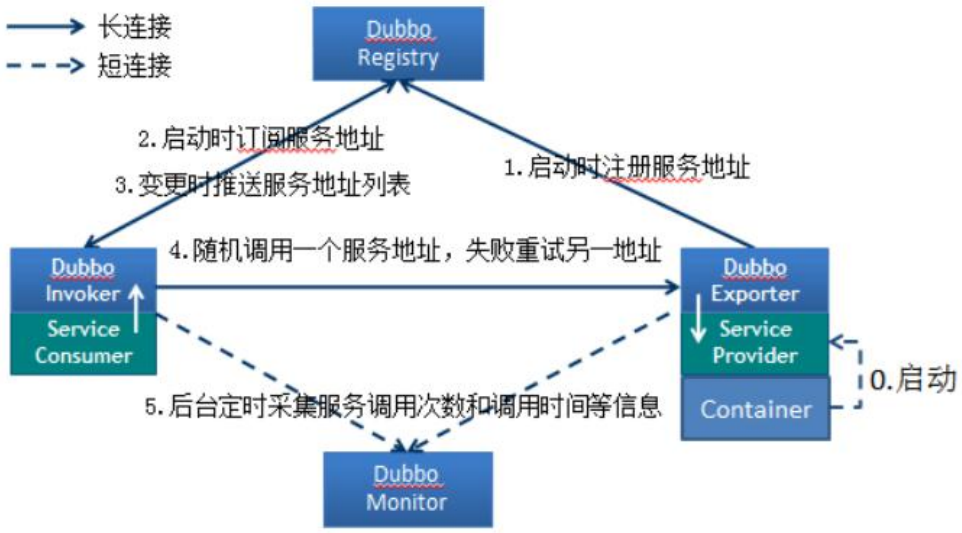
Dubbo 分布式服务框架

Dubbo 架构图解



节点角色说明：

Provider: 暴露服务的服务提供方。

Consumer: 调用远程服务的服务消费方。

Registry: 服务注册与发现的注册中心。

Monitor: 统计服务的调用次调和调用时间的监控中心。

Container: 服务运行容器。

调用关系说明：

0. 服务容器负责启动，加载，运行服务提供者。

1. 服务提供者在启动时，向注册中心注册自己提供的服务。

2. 服务消费者在启动时，向注册中心订阅自己所需的服务。

3. 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。

4. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。

5. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

1. 给我介绍一下 dubbo 吧？

我最近做的这个分布式的项目，项目之间通信就是用的 dubbo 框架，我们用的是 dubbo 改进版，就是当当网后期给升级了一下，叫 dubbox 了，在使用上我觉得都差不多，我们在用的时候是用 zookeeper 做的注册用心我们是结合 maven 一起使用的，我觉的分布式和最早以前传统的单架构项目的区别是，传统的架构对于后期的维护和升级不怎么方便；我们通过 maven 的模块式开发，就可以把一个单架构的工程，拆封成一个一个的小模块，包括（jar 和 war）；jar 包可以被 war 包直接依赖。war 包可以分布式部署，但是这样不同的 war 之间的交互就成为了问题。我们之前的做法是，两个 war 之间进行交互时，一般采用 webservice 或

httpclient。但是在这种分布式架构中，多个 war 之间的交互会特别频繁，如果用 webservice 或 httpclient 架构师很多时候自己都搞不清楚他们之间的依赖关系了。但是我们使用了 dubbo 就完美的解决了这个问题，有了 dubbo 架构以后，多个 war 之间不在直接进行交互，而是统一发布到 dubbo 的注册中心 zookeeper 进行交互。无论是发布服务还是调用服务，都通过 dubbo 实现，并且提供了 web 页面，来监控和管理各个接口直接的调用。

2. 你们在项目中如何使用 dubbo 的?

Dubbo 是依赖 spring 容器的,所以我们在使用的时候是结合 spring 用的,我们在配置服务的提供者的项目中,在 Spring 的 xml 配置文件中写下面标签注册到注册中心,在 dubbo 标签中配置服务提供者端口号,服务提供名称,注册中心的地址和端口号,下面红色代码不用说.<dubbo:protocol name="dubbo" port="20887"></dubbo:protocol>(服务提供者端口号)

```
<dubbo:application name="youlexuan-cart-service"/>
<dubbo:registry address="zookeeper://192.168.177.129:2181"/>
<dubbo:annotation package="com.youlexuan.cart.service.impl" />
```

调用是通过,在 springmvc.xml 文件中配置<!-- 引用 dubbo 服务 -->

```
<dubbo:application name="youlexuan-cart-web" />
<dubbo:registry address="zookeeper://192.168.177.129:2181"/>
<dubbo:annotation package="com.youlexuan.cart.controller" />
```

用 dubbo 的好处是：把项目拆分成各个独立的小工程，通过接口调用方式，互相交互数据，可以单独进行部署和升级，这样就减少了他们直接的耦合性和代码的复用性。还可以针对不同模块采用不同的部署策略，比如订单模块并发比较高，所以可以把订单模块这个 war 包，单独部署多套，都注册到同一个 zookeeper 中。当客户端调用时，zookeeper 会帮忙进行负载处理。

3. 你们通过什么方式把服务发布到注册中心的?

通过 tomcat 发布到注册中心的和通过 tomcat 从注册中心调用的.

4. Dubbo 容错策略有哪些?

我**大概**记得有那么几种策略失败重试,失败自动恢复， 并行调用,广播调用 具体怎么配置的,**用的话我得去查一下**,我知道就是在那个 dubbo 的标签里配置的,

参考一下链接

https://www.cnblogs.com/xhj123/p/9087532.html

5. DubboDubbo 超时时间设置怎么设置，服务调用超时问题如何解决

Dubbo 超时时间设置有两种方式：

服务提供者端设置超时时间，在 Dubbo 的用户文档中，推荐如果能在服务端多配置就尽量多配置，因为服务提供者比消费者更清楚自己提供的服务特性。

服务消费者端设置超时时间，如果在消费者端设置了超时时间，以消费者端为主，即优先级更高。因为服务调用方设置超时时间控制性更灵活。如果消费方超时，服务端线程不会定制，会产生警告。

6. dubbo 在调用服务不成功时，默认是会重试两次。负载均衡实现原理是什么？

我原来看过点 dubbo 里的源码,知道里面有四种实现负载均衡的方式,
我记得其中一个

Random LoadBalance

随机，按权重设置随机概率。在一个截面上碰撞的概率高，但调用量越大分布越均匀，而且按概率使用权重后也比较均匀，有利于动态调整提供者权重。

第二个是：

RoundRobin LoadBalance(不推荐，基于权重的轮询负载均衡机制)

轮循，按公约后的权重设置轮循比率。
存在慢的提供者累积请求的问题，比如：第二台机器很慢，但没挂，当请求调到第二台时就卡在那，久而久之， 所有请求都卡在调到第二台上。

第三个是：

LeastActive LoadBalance

最少活跃调用数，相同活跃数的随机，活跃数指调用前后计数差。使慢的提供者收到更少请求，因为越慢的提供者的调用前后计数差会越大。

第四个是：

ConsistentHash LoadBalance

1、一致性 Hash，相同参数的请求总是发到同一提供者。(如果你需要的不是随机负载均衡，是要一类请求都到一
个节点，那就走这个一致性 hash 策略。)
2、当某一台提供者挂时，原本发往该提供者的请求，基于虚拟节点，平摊到其它提供者，不会引起剧烈变动。
3、缺省只对第一个参数 Hash，如果要修改，请配置 <dubbo:parameter key="hash.arguments" value="0,1"/>
4、缺省用 160 份虚拟节点，如果要修改，请配置 <dubbo:parameter key="hash.nodes" value="320" />

可以参考链接理解 https://blog.csdn.net/shengqianfeng/article/details/82314269

7. 你们公司的 double 注册中心用的什么？

用的是 zookeeper，你们用了几台 zookeeper?我们开发的时候用的是 1 台服务器，然后上线的时候用的是 3 台，一个主的两个备用的，你知道为什么是 3 台吗？知道啊，因为 zookeeper 集群是投标选举机制，必须是奇数的，超过半数以上，主的挂掉，备用的就自动启动了，咱们互联网思维，集群不就是为了解决服务器不宕机的问题么.

8. 解释下 dubbo,云服务,和 zookeeper?

问：Zookeeper 关掉以后还能使用？还能调用服务？调用服务的时候是不是不经过 zookeeper？

zookeeper 关掉以后还能使用，你先是一个服务，然后注册到 zookeeper，服务消费者第一调用的时候要拿到服务提供者的地址跟端口号。通过地址端口号用 rpc 可以调用。

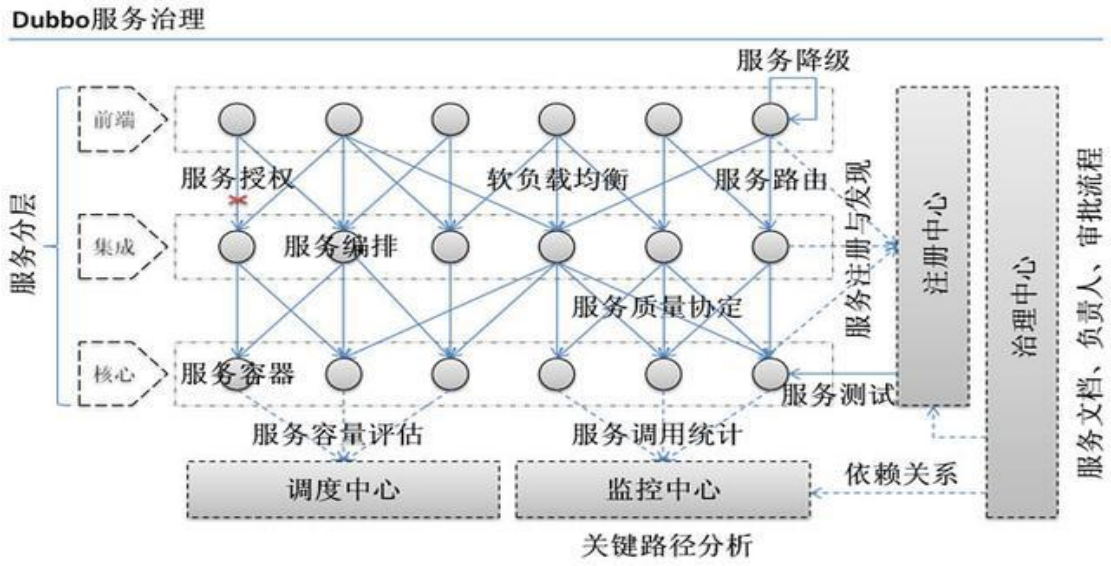
不使用 zookeeper 服务可以调用 dubbo 框架么？

可以，Dubbo 将注册中心进行抽象，它可以外接不同的存储媒介给注册中心提供服务，有 zookeeper，

问：在一个项目里引用多个服务，怎么保证你引用的那个服务就是那个服务？

答：你打开不是有监控中心，你可以看服务后台它注册了多少个服务，消费端有多少个，提供者有多少个。端口号不同，一个服务启动起来会占用一个端口，而且是一个 java jvm 的进程。如果一台服务分配到两三台机器，会给负载均衡策略的。

9. 为什么需要服务治理，注册中心挂掉 发布者和调用者还能通信吗？



过多的服务 URL 配置困难；

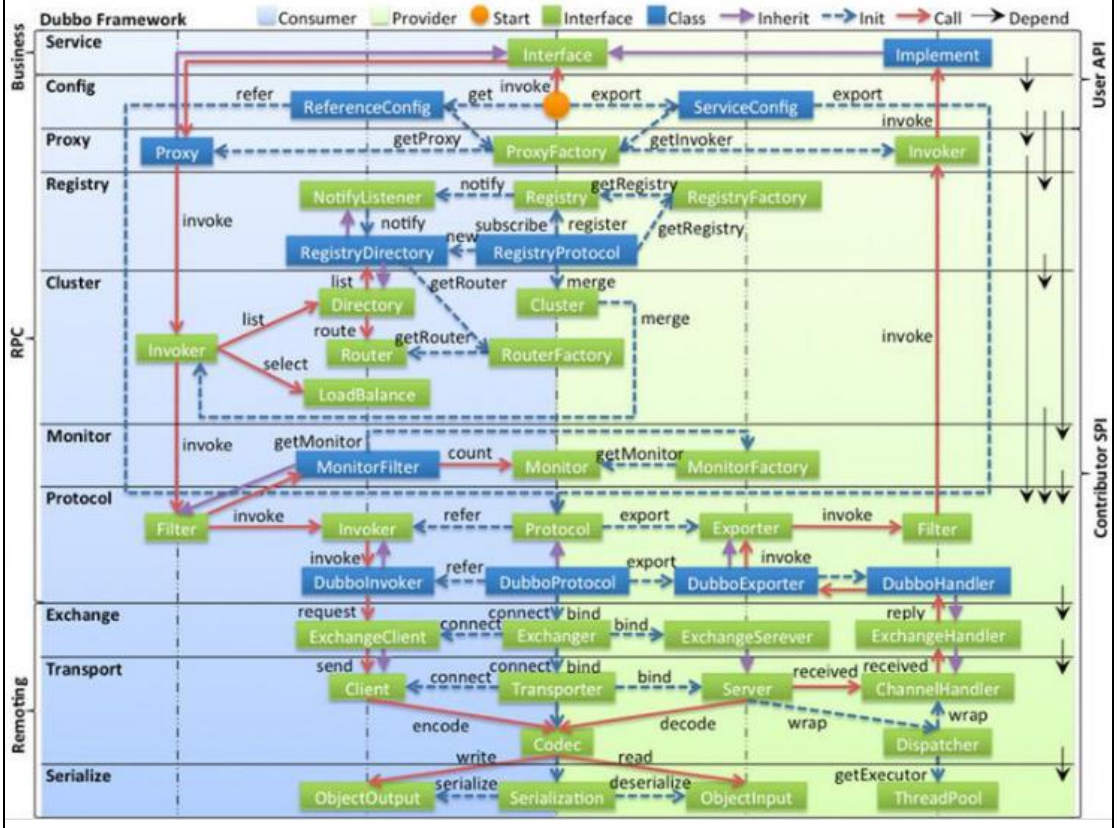
服务依赖混乱，启动顺序不清晰，给开发和测试带来很多麻烦；

过多服务导致性能指标分析难度较大，需要监控；

注册中心挂掉 发布者和调用者可以通信的，启动 dubbo 时，消费者会从 zookeeper 拉取注册的生产者的地址接口等数据，缓存在本地。每次调用时，按照本地存储的地址进行调用。

10. 介绍下 Dubbo 的原理？

关于 dubbo 的原理，我也是稍微研究了一下，可能不是太透哈，dubbo 的框架设计的话一共分为 10 层，我就说一下比较我记住的那些吧。
最外层的话，是接口，或者服务层，就是我们配置文件中所暴露出的接口，这个是我们能看到的然后再往下边，是配置层，主要负责读取我们的配置文件，把这些配置信息读取出来还有一个服务代理层，也就是负责代理我们所写的服务执行对应功能，就像 Spring 中 JDK 代理类似。再往下的话，是服务注册层，读取完配置文件信息，需要将 dubbo 的配置信息注册到注册中心再走的话，有一个监控层，可以检测 rpc 协议执行过程中的调用次数及调用时间再有信息交互，网络传输层，这里才是进行网络的传输，进行的是 socket 传输
还要一个数据序列化层，这一层就是讲要传输的数据对象进行序列化，所以我们在写代码时需要让对象实现序列化接口



图中从下至上分为十层，各层均为单向依赖，右边的黑色箭头代表层之间的依赖关系，每一层都可以剥离上层被复用，其中，Service 和 Config 层为 API，其它各层均为 SPI。各层说明：

第一层：service 层，接口层，给服务提供者和消费者来实现的

第二层：config 层，配置层，主要是对 dubbo 进行各种配置的

第三层：proxy 层，服务接口透明代理，生成服务的客户端 Stub 和服务器端 Skeleton
第四层：registry 层，服务注册层，负责服务的注册与发现
第五层：cluster 层，集群层，封装多个服务提供者的路由以及负载均衡，将多个实例组合成一个服务
第六层：monitor 层，监控层，对 rpc 接口的调用次数和调用时间进行监控
第七层：protocol 层，远程调用层，封装 rpc 调用
第八层：exchange 层，信息交换层，封装请求响应模式，同步转异步
第九层：transport 层，网络传输层，抽象 mina 和 netty 为统一接口
第十层：serialize 层，数据序列化层。网络传输需要

11.谈谈对 Zookeeper 的理解

ZooKeeper 是一个集中式服务，用于维护配置信息，命名，提供分布式同步和提供组服务，在这个项目 zookeeper 主要也是作为注册中心来使用。其实 dubbo 的注册中心也可以使用 Redis 来实现的。
至于怎么实现。主要用到了 Redis 两点功能
1. key-value 的存储
2. 发布和订阅的消息机制

具体的配置可以跟您简单说一下
连接 redis 服务器；
服务消费者启动后从 key 值中获取服务提供者地址；
服务消费者订阅服务提供者启动的消息；
服务消费者设置地址；
当有新的服务提供者上线时，重新获取服务提供者地址列表。

12、dubbo 的健壮性表现

1. 监控中心宕掉不影响使用，只是丢失部分采样数据
2. 数据库宕掉后，注册中心仍能通过缓存提供服务列表查询，但不能注册新服务
3. 注册中心对等集群，任意一台宕掉后，将自动切换到另一台
4. 注册中心全部宕掉后，服务提供者和服务消费者仍能通过本地缓存通讯
5. 服务提供者无状态，任意一台宕掉后，不影响使用
6. 服务提供者全部宕掉后，服务消费者应用将无法使用，并无限次重连等待服务提供者恢复我们前面提到过：注册中心负责服务地址的注册与查找，相当于目录服务，服务提供者和消费者只在启动时与注册中心交互，注册中心不转发请求，压力较小。

13、采用 dubbo 直连

即在服务消费方配置服务提供方的位置信息。

- xml 配置方式：

<dubbo:reference id="userService" interface="com.kaola.service.UserService"
url="dubbo://localhost:20880" />

- 注解方式：

@Reference(url = "127.0.0.1:20880")
HelloService helloService;

14、Dubbox 直连

```
<dubbo:protocol name="dubbo" port="20882"></dubbo:protocol>
<dubbo:application name="kaola-content-service"/>
<dubbo:registry address="zookeeper://192.168.25.134:2181"/>
<dubbo:annotation package="com.kaola.content.service.impl" />
```

15、服务降级

场景一、
比如说服务 A 调用服务 B，结果服务 B 挂掉了，服务 A 重试几次调用服务 B，还是不行，直接降级，走一个备用
的逻辑，给用户返回响应
场景二、
在开发自测，联调过程中，经常碰到一些下游服务调用不通的场景，这个时候我们如何不依赖于下游系统，就业务
系统独立完成自测？
dubbo 自身是支持 mock 服务的，在 reference 标签里，有一个参数 mock，该参数有四个值，false,default,true,或者
Mock 类的类名。分别代表如下含义:

- false，不调用 mock 服务。
- true，当服务调用失败时，使用 mock 服务。
- default，当服务调用失败时，使用 mock 服务。
- force，强制使用 Mock 服务(不管服务能否调用成功)。(使用 xml 配置不生效,使用 ReferenceConfigAPI 可以生效)
- 使用方法:将 mock 参数启用，在<dubbo:reference>中添加参数项 mock=true。实现需要调用的服务接口上游系统

需要调用下游系统，则下游需要提供 jar 包给上有系统，该 jar 包只有接口，没有实现。我们需要实现该接口，
且 命名 必须 是该 接口 名 +Mock， 例 如 原 接口是 com.alibaba.dubbo.demo.DemoService, 则 实 现 类必 须 是 com.alibaba.dubbo.demo.DemoServiceMock。

16、zookeeper 宕机与 dubbo 直连的情况

在实际生产中，假如 zookeeper 注册中心宕掉，一段时间内服务消费方还是能够调用提供方的服务的，实际上它使 用的本地缓存进行通讯，这只是 dubbo 健壮性的一种体现。

17、了解 Ngnix 吗?

Nginx 是一款轻量级的 Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器。 Nginx 主要提供反向
代理、负载均衡、动静分离(静态资源服务)等服务

我们项目上线之后用的是一个负载均衡，现在用的比较多了，就我了解的像新浪，搜狐，快手都有用到，其实就是一个负载均衡服务器，反向代理，我们在访问项目的时候是不需要直接访问服务器的，而是访问 ngnix 服务器，由 ngnix 来决定访问哪一个服务器，在 ngnix.config 配置文件里面配置代理服务器，想代理几台就代理几台，就那台服务器配置高点的话，可以配置一个 weight,就是配置一个权重，把权重配置的稍微大点，如果是服务器的配置低的话呢，我们也可以配置 weight 低一点，
像项目后期的测试这块，我也都有跟进过。我们用的是 windows 版本的 ngnix,包括在 linux 上的也都有实现过，最终项目上线的时候，是上线到 linux 上的，ngnix 用的是两台服务器，其实还有一台是三台，一主一备，还有一个高可用的一个服务，让 ngnix 的主节点和备用服务器呢，来传输一个心跳协议，来检测状态。如果主节点挂掉的话，就是从的，那个备用服务器会立马启动起来工作，然后运维人员对主节点进行相应的维修，再次重新启动，再转回主节点。

18、Nginx 的四个主要组成部分了解吗?

- Nginx 二进制可执行文件：由各模块源码编译出一个文件
- Nginx.conf 配置文件：控制 Nginx 行为
- acess.log 访问日志： 记录每一条 HTTP 请求信息
- error.log 错误日志:

19、Nginx 的特点

1、高并发，高性能
2. 可扩展性好（模块化设计，第三方插件生态圈丰富）
3. 高可靠性（可以在服务器行持续不间断的运行数年）
4. 热部署（这个功能对于 Nginx 来说特别重要，热部署指可以在不停止 Nginx 服务的情况下升级 Nginx）
5. BSD 许可证（意味着我们可以将源代码下载下来进行修改然后使用自己的版本）

20、介绍一下 Nginx 反向代理

他就是一个反向代理服务器,其实它的优点就是内存占用少,并发访问能力强,我知道的新浪,斗鱼,好这些大公司都使用这个技术,它底层实现是 c 语言实现的.我们主要用到 nginx 两大主要的功能吧,一个是反向代理,一个是负载均衡,先说一下这个反向代理,咱们还得先说一下正向代理,其实咱们平时调试开发都是正向代理,只不过我们不说这个词.比如吧,我们访问一台 tomcat,默认端口号是 8080,那我们访问的时候可能就是 localhost:8080,这样顺着来呢,就可以理解成一个正向代理,就这样理解哈,这个时候,如果我们想要再来一台服务器呢,我们可以配一下,把端口号改成 8081,通过访问不同的端口号来访问, 但是当我们项目

要上线的时候，如果需要把一个项目如果部署到两台服务器上，比如淘宝，这么大，它的主界面不可能是在一台服务器上放着，就不能是访问 8080 或者 8081 这些端口了，这个时候，需要有一个代理的服务器，能够给这两台服务器做一个代理，直接不需要进行标明，就可以访问到任意一台服务器，找到这个主界面。这里呢，这个代理就可以代理这些服务器了，这个时候这个代理，我们可以理解成反向代理。反向代理严格的概念是通过代理服务器来接收网路上的请求，然后将请求转发给内部网路的服务器。而 nginx 可以干这个活，做这个代理，我们可以在 nginx 中配置端口，ip 或者域名指向这些不同端口，甚至不同 ip 的服务器。这就是反向代理这个概念。

21、说下 nginx 负载均衡

Nginx 还有一个重要的功能叫做负载均衡，我们做服务器的集群，怎样保证集群中服务器被均等的进行访问呢，不能说我们认为搭建好了服务器的集群它就会均衡的去访问，这个时候我们可以统一的去访问 nginx 这个服务器，在 nginx 的配置信息中，去配置好这些服务器，它配置文件是这样的，只要你配上，默认访问的比率就是一样的，这个就是负载均衡，当然 nginx 更厉害的是可以配置权重，比如说哈，我两台服务器，其中一台性能比另外一个性能好 2 倍，那我是不是应该访问性能好的服务器频率更高一些，咱们就可以在 nginx 的配置文件中配置一个 weight 属性，指定权重。当然还有其他一些配置的，比如有些服务器需要整修，那咱们就可以配置某台服务器暂时 down 掉，这样用户访问的时候，就不会访问到这台服务器，当修好之后，我们在把这个配置信息干掉就行了。

22、如何配置 Nginx?

我们配置 nginx 是在 nginx 安装目录下的 conf 下有一个 nginx.conf 文件，主要是修改这个配置文件，比如咱们配置反向代理和负载均衡服务器，配置一个 proxy_pass 指向代理服务器，配一下 upstream server 指向要访问的 ip 和端口，这个可以配置多个 ip，可以设置 weight 权重什么的?

23、Nginx 使用方法?

其实 nginx 服务器搭建这一块是由公司的运维去做的，我原来在自己的虚拟机里也配置过，可以搭建两台 nginx 服务器，主备关系，用那个 keepalived 管理主备服务器，keepalived 其实就是让主备服务器之间发送心跳协议，保证 nginx 永不宕机.

24、动静分离

动静分离是让动态网站里的动态网页根据一定规则把不变的资源和经常变的资源区分开来，动静资源做好了拆分以
后，我们就可以根据静态资源的特点将其做缓存操作，这就是网站静态化处理的核心思路。

20、工作中遇到的问题：

1、内存溢出，可以调节 tomcat 内存解决。还要找出代码中内存溢出的漏洞，一般是流没有关闭或者死循环；
2、数据库连接超出最大连接量：修改 mysql 配置文件的最大连接数；
3、代码编译失败、缓存：清理 Eclipse 编译缓存、清理 tomcat 缓存、清理浏览器缓存。
4、tomcat 启动端口号被占用：打开任务管理器，杀进程；
5、tomcat 启动超时：修改启动时间；