# 一、Lombok框架

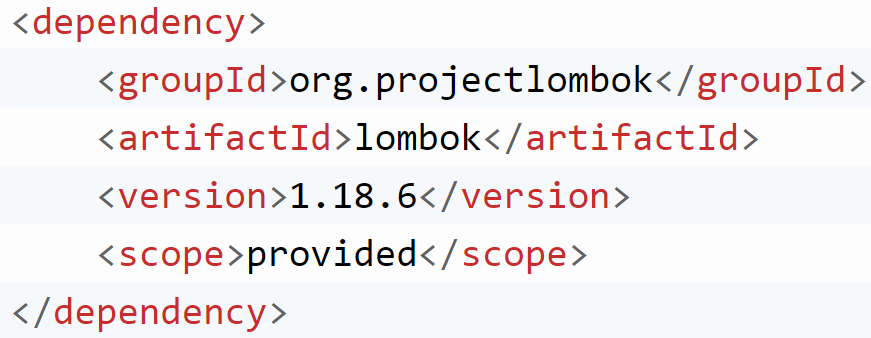
## 1. Lombok的作用

如果用一句话来介绍lombok的话，那么lombok的作用就是通过简单的注解来精简代码以达到消除冗长代码的目的。

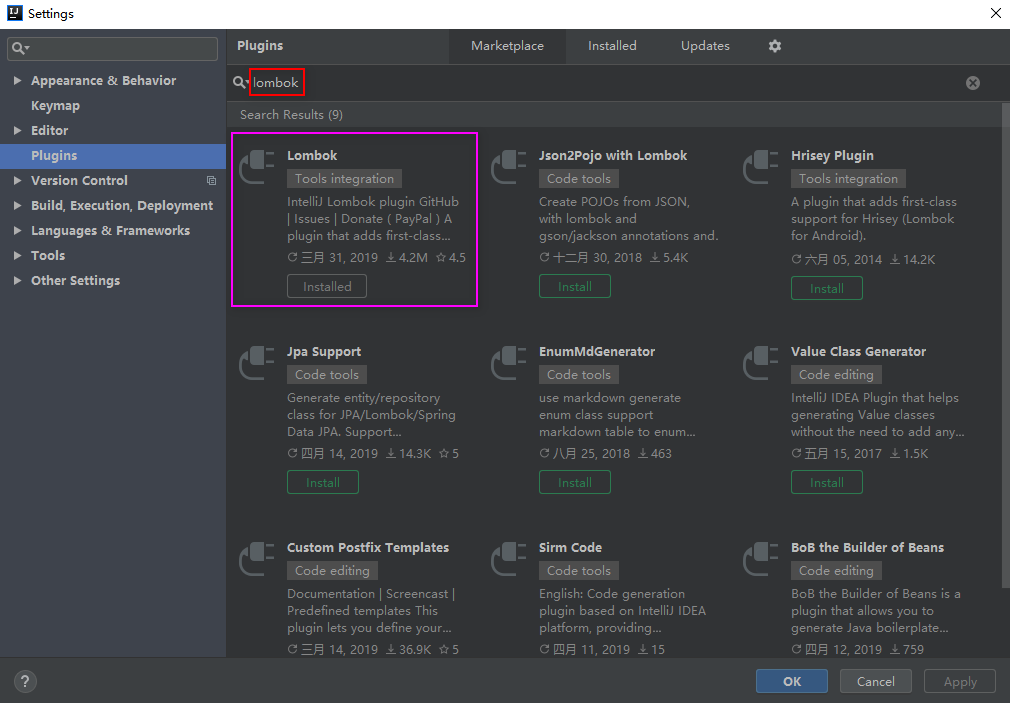
## 2. Lombok的优点

* 提高编码效率；
* 使代码更简洁；
* 消除冗长代码；
* 避免修改字段名字时忘记修改方法名。

## 3. pom.xml中关于Lombok的依赖

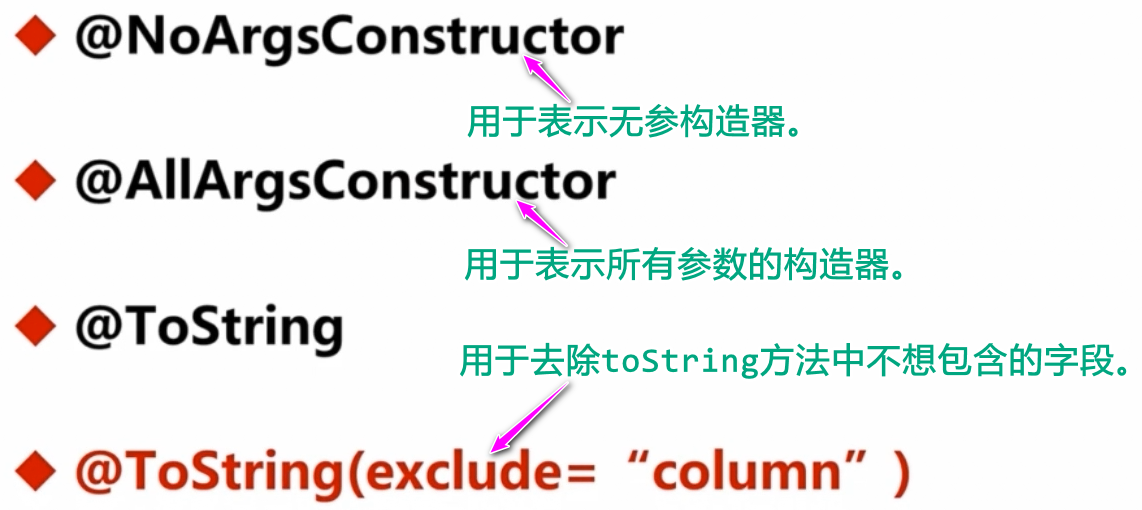


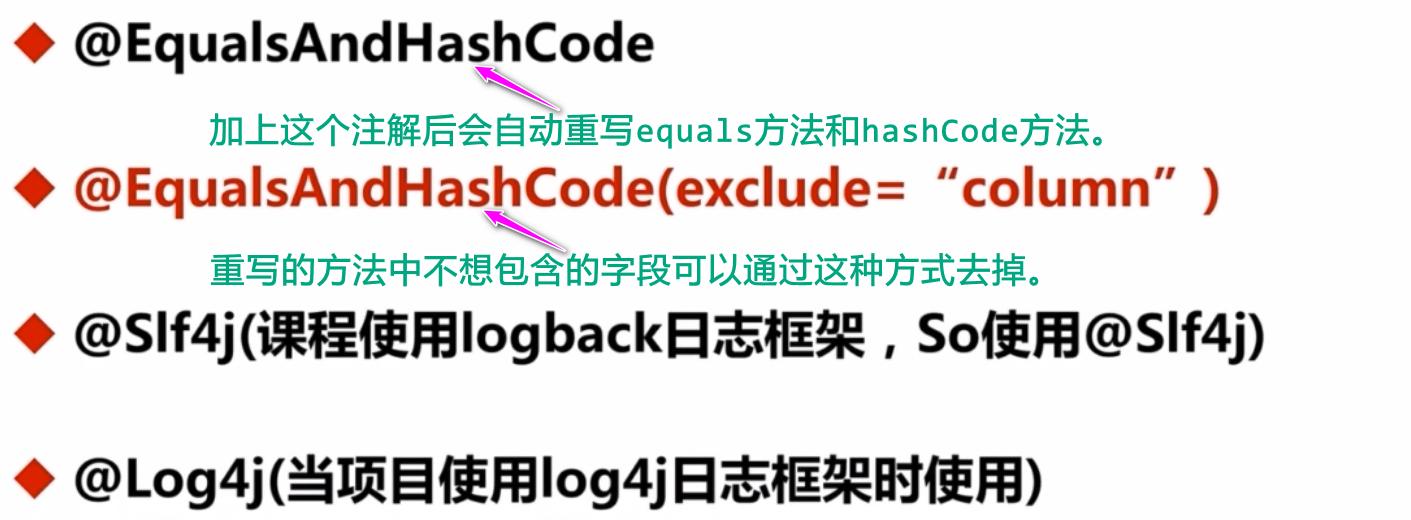
上面的scope标签有没有都可以，都不影响。另外，如果是在IDEA中使用lombok框架的话，对于maven项目，除了要在pom.xml中引入依赖外，还要在IEDA中安装相应的插件，如下图所示。

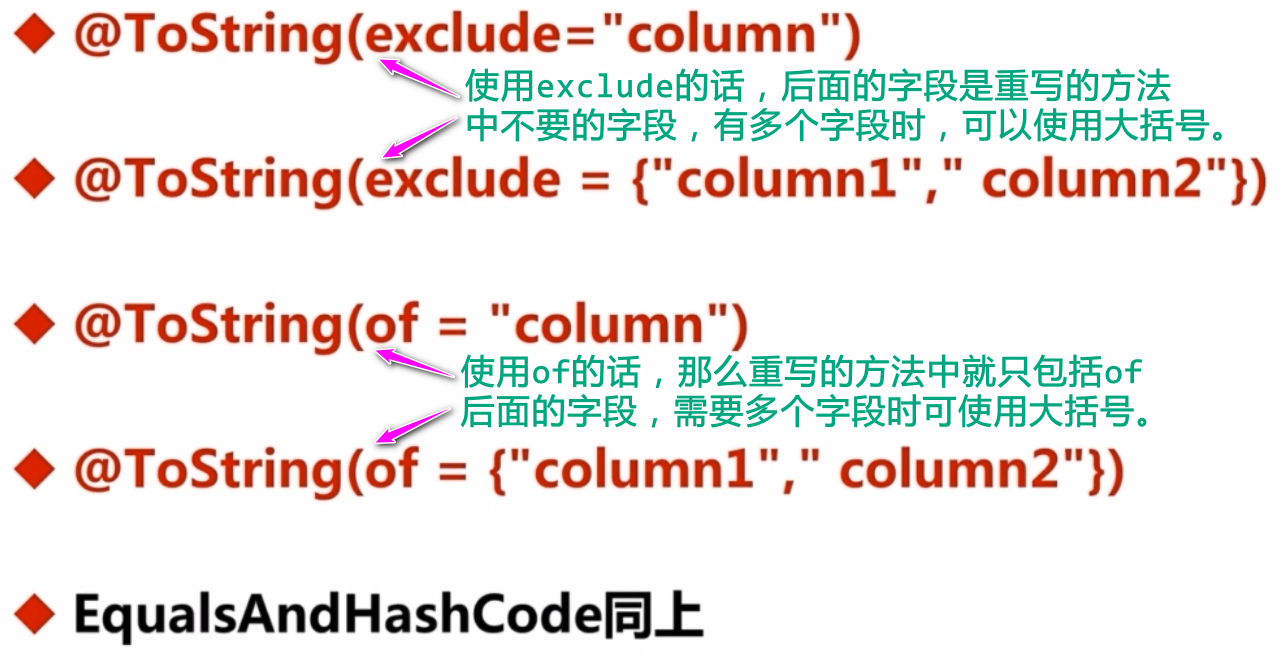


## 4. Lombok的常用注解





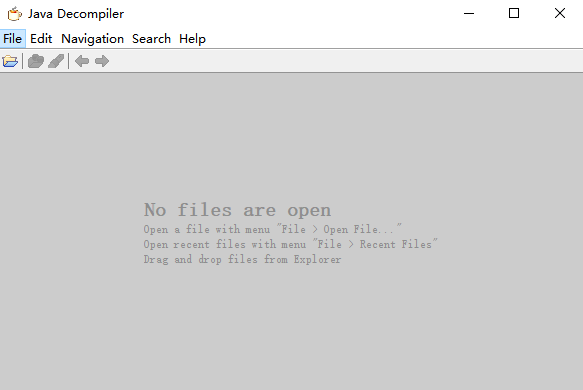






# 二、反编译工具

这里要说的反编译的工具就是java Decompiler，我们可以使用这个工具把class文件进行反编译，从而查看到对应的源码文件，该工具的界面如下。



# 三、maven的环境隔离

## 1. 使用maven环境隔离的原因

自己在本地开发时几乎是不需要使用到环境隔离的，但是在实际的项目开发中进行环境隔离是很有必要的。因为实际的项目开发中是会有多个环境的，不同的环境是会存在一定差异的，为了让这些环境之间互不影响，这时就需要用到Maven的环境隔离了。

实际的项目环境一般包括本地开发环境（Local），开发环境（Dev），测试环境（Beta）以及线上环境（Prod）。由于我是在本地开发，本地开发环境和开发环境是一样的，所以下面在配置的时候，本地开发环境就不配置了。

|  |
| --- |
| 隔离的各个环境之间的各种配置可能存在的一些常见差异：   * FTP服务器相关的配置信息不一样； * 数据库配置不一样； * 支付接口不一样，例如支付宝分沙箱环境和真实环境，我们在开发的时候可以用沙箱环境，但是上线后就要用真实环境了； * 框架的配置文件不一样，例如日志的输出路径等等。 |

以上提到的只是一部分的差异，我们在本地开发时，明明项目是可以正常运行的，但是一到线上就是各种报错，这就是线下的开发环境和线上环境不同导致的。由于环境不同，所以在本地开发的项目，即便在本地能正常运行，但是在发布到线上后，很多配置的内容其实还是需要进行修改的，否则就会报错。

|  |
| --- |
| maven环境隔离解决的实际问题：   * 避免人工修改的弊端，因为人为修改容易犯错； * 轻松分环境编译、打包、部署等等； |

我们把由于环境不同而可能内容不同的配置文件拷贝成多份，比如开发环境一份、测试环境一份、线上环境一份。然后在开发环境下需要进行打包等操作时，就把开发环境中的那份打包进来；如果是线上环境需要进行打包等操作时，就把线上环境的那份打包进来，这样一来就完成了分环境进行打包等操作。如果不是拷贝多份的话，就需要根据环境的不同来修改配置文件的内容，这样就会很容易出错。

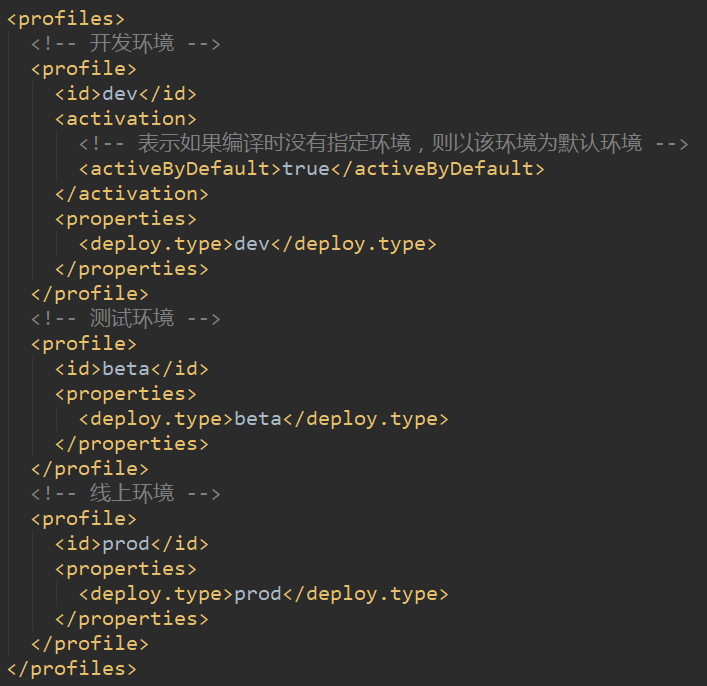
## 2. 配置maven环境隔离的步骤

### 2.1 pom.xml中的配置

①首先在build节点中增加如下配置：

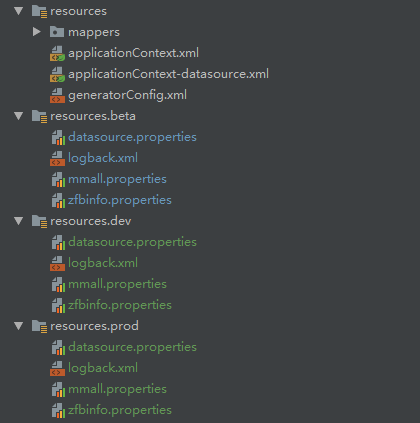


②然后在pom.xml中增加profiles节点，并作如下配置：

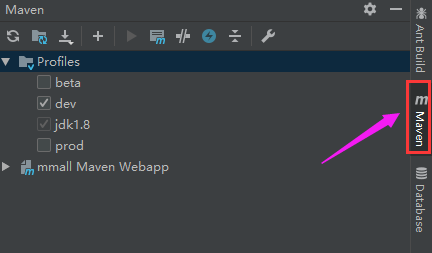


### 2.2 新建针对不同环境的文件夹

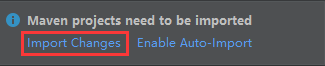
这里把可能会因为环境不同而内容不同的配置文件分别拷贝到不同环境的文件夹下，如下图所示。拷贝好之后，再根据环境来修改对应该环境的目录下相应的配置文件的内容。



### 2.3 在IDEA中设置默认环境



目前使用的环境是开发环境（dev），需要注意的是，这里的对勾分实心的对勾，还有半实心的对勾，像下面jdk1.8前面的对勾就是半实心的，而我们在设置默认环境的时候一定要打实心的对勾。如果我们想要更改默认环境的话，就在这里更改就可以了。每次更改的时候，上图的下面会出现下图中所示的内容。这时要点击我标记的地点，以便能够保存更改。



### 2.4 执行打包等命令

如果要对项目执行打包操作的话，就在命令提示符中输入以下命令：

|  |
| --- |
| mvn clean package -Dmaven.test.skip=true -Pdev |

这个命令是针对开发环境的，如果是测试环境的话，以上命令其他部分不变，最后面的部分变成-Pbeta，同理，线上环境的话，就变成-Pprod。如果是使用以下命令的话，就代表使用默认环境进行打包，而之前在pom.xml中配置的默认环境是开发环境（Dev）。

|  |
| --- |
| mvn clean package -Dmaven.test.skip=true |

# 四、Tomcat集群与nginx负载均衡

## 1. 基本概念

我之前在作总结的时候，有专门讲过集群、分布式以及负载均衡的区别。我对nginx负载均衡的理解就是，当浏览器中的请求到达nginx的时候，由于与nginx相关联的tomcat服务器有很多，所以nginx会把用户的请求随机分发给其中的某个服务器来处理，由于请求是随机分发给任意一个服务器来处理的，所以不会出现过多请求都交给某一个服务器来处理的情况，这一行为就是nginx的负载均衡。这样做的好处是不会使某一个服务器的负载过重，从而就达到了负载均衡的目的。不过由于每个服务器的性能可能不一样，所以我们也可以通过权重的方式，把更多的请求分发给性能较好的服务器来处理。

与nginx相关联的多个tomcat服务器就叫做tomcat集群，其实集群可以理解成是很多人在干同一件事。就是由于请求太多了，一个服务器不够用，所以才会用多个服务器一起来处理请求，这其实就是集群。下面就主要讲一些关于集群相关的知识。

## 2. Tomcat集群的作用和实现原理

### 2.1 Tomcat集群的作用

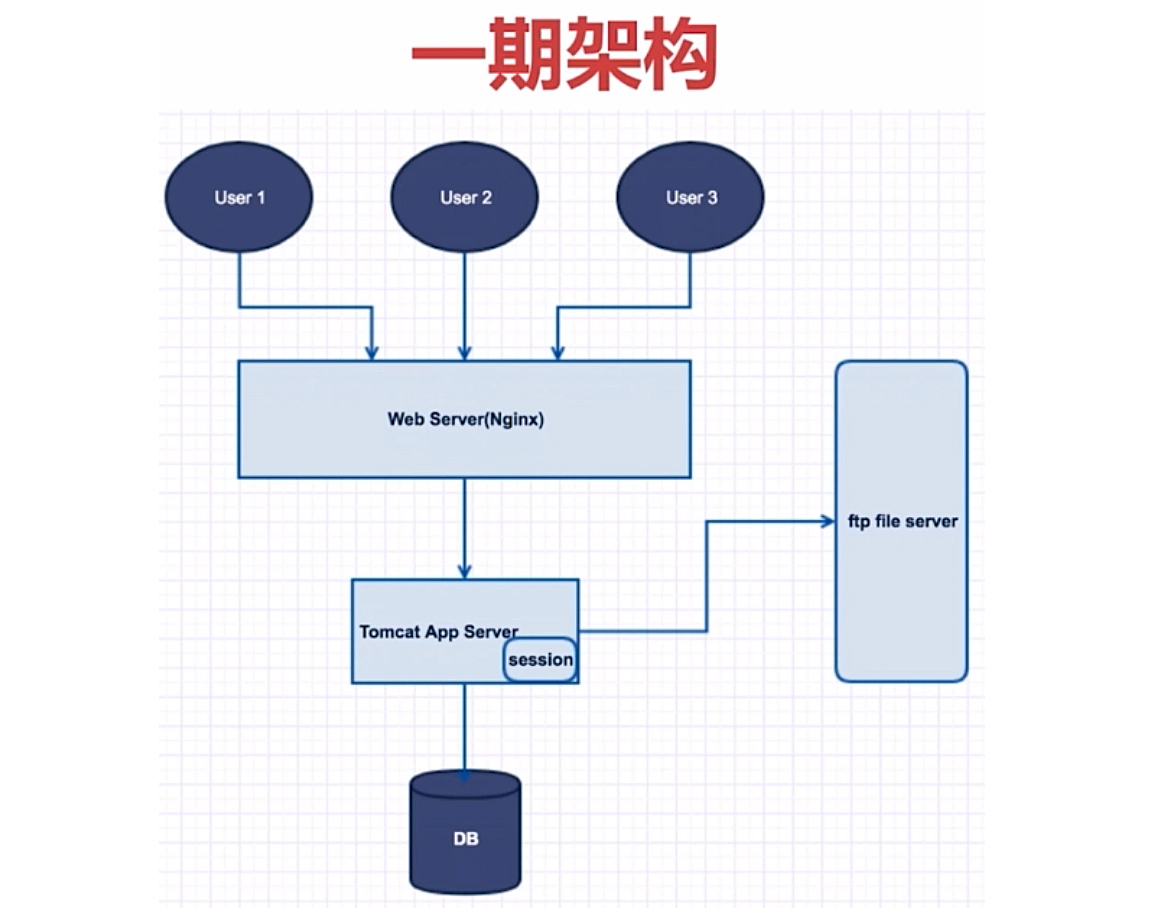
* 提高服务的性能，并发能力以及高可用性；
* 提供项目架构的横向扩展能力。

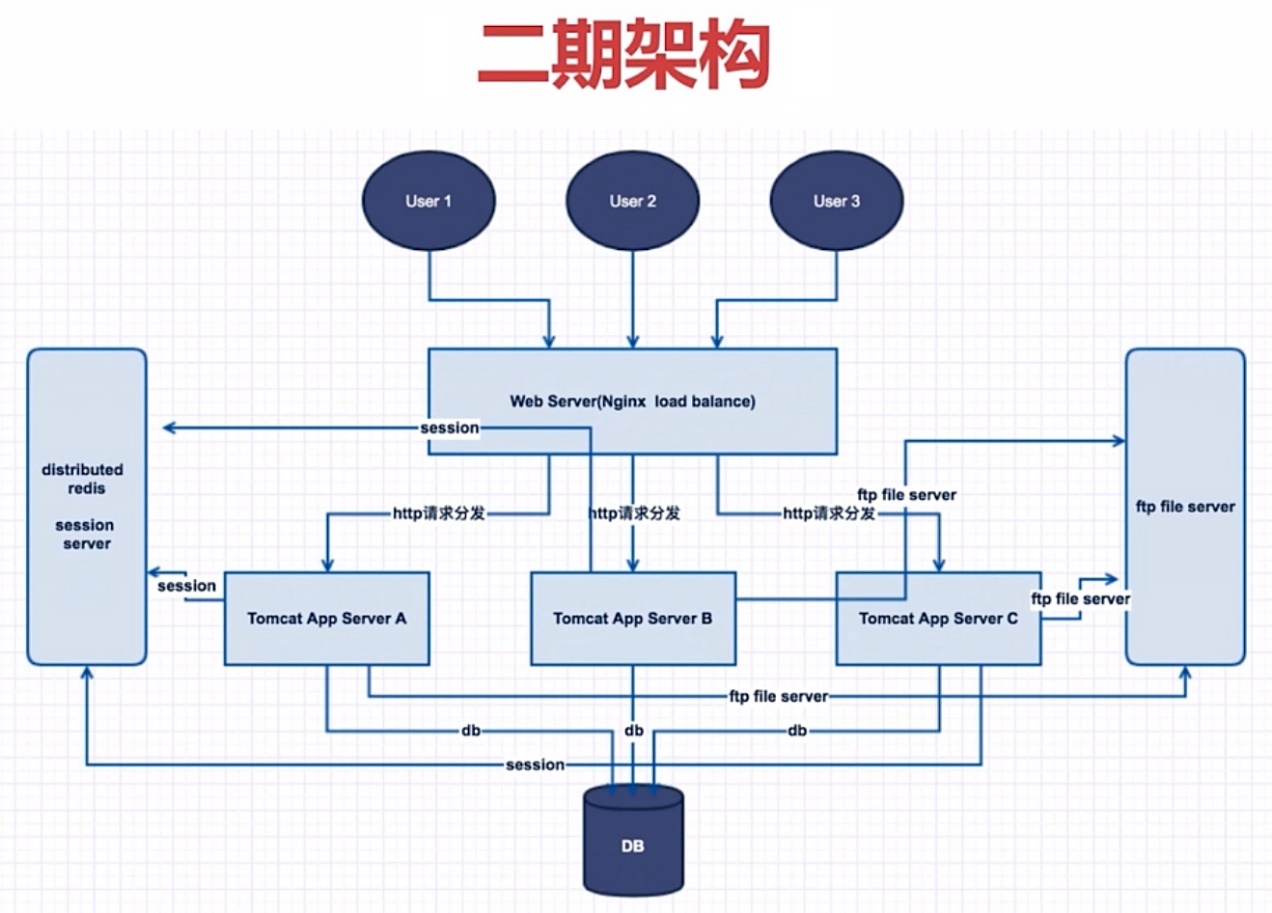
### 2.2 Tomcat集群的实现原理

* 通过nginx负载均衡进行请求转发。

注：tomcat集群不一样就只有以上提到的两个作用，也不一定就上面提到的一个实现原理，我这里记录的都只是老师在课上提到的。

## 3. 一期架构和二期架构对比





## 4. nginx负载均衡的常见策略







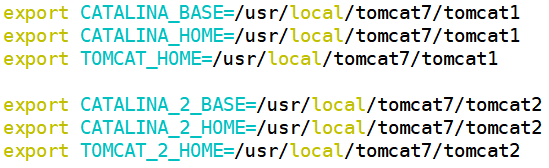
以上关于权重这一策略，我这里特别讲一下，该项目中使用的就是这种策略。上图中在使用weight的时候，分别使其等于15和10。这里需要说明的是，权重表示的只是概率问题，weight=15表示请求分到该服务器上的概率是weight=10时的1.5倍。并不是说有25个请求就一定有15个被分到weight=15所在服务器上。

## 5. 集群和负载均衡的具体步骤

在叙述详细步骤前先说一下，我下面步骤中演示的tomcat和nginx都是在linux环境下，不过在访问的时候，是在windows下的浏览器中进行访问的。

### 5.1 配置环境变量

首先在linux的/usr/local/tomcat7目录下增加两个tomcat，分别命名为tomcat1和tomcat2，然后执行vim /etc/profile打开profile文件，打开后在该文件中增加下图中这些内容。



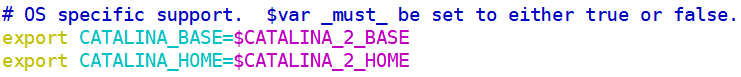
以上配置的意思是在集群中设置两个tomcat服务器，如果还想配置更多的话，就按照图中的方式依次加上就可以了。保存退出后执行source /etc/profile使配置生效。如果不执行这个命令，而是重启虚拟机的话，也可以生效。

注：如果之前在profile文件中配置过tomcat，记得给注释掉，以免对这里配置tomcat集群产生影响。

### 5.2 修改tomcat2中的catalina.sh文件

这里tomcat1的catalina.sh文件不进行修改，只修改tomcat2的catalina.sh文件，具体位置为：/usr/local/tomcat7/tomcat2/bin/catalina.sh。

打开tomcat2的bin目录下的catalina.sh文件后，在下图的注释下面新增两行内容，新增的具体内容如下图所示。新增成功后，保存退出。

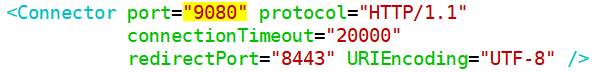


### 5.3 修改tomcat2中的server.xml文件

修改完catalina.sh后，来到tomcat2的conf目录下，修改该目录下的server.xml文件，主要是修改三个端口。首先是把下图中的8005端口改成9005。



然后是把下图中的8080端口改成9080端口。



最后是把下图中的8009端口改成9009端口。



修改完成之后，保存退出即可。需要提一下的是，以上三个端口不一定非要改成图中的那三个端口，改成其他的也行，但是必须要保证端口没有被占用。

### 5.4 分别进行访问测试

启动tomcat1的bin目录下的startup.sh以及tomcat2的bin目录下的startup.sh。在启动之前不要忘记在linux的防火墙中开放这里修改的9080端口。另外，为了看出两个tomcat的区别，可以把tomcat2的webapps/ROOT目录下的tomcat.png图片换成其他的图片，但是名字还叫这个，以便观察区别。修改好之后，就可以在windows下的浏览器中分别进行访问了。

由于我linux系统的ip是192.168.31.18，所以访问地址为192.168.31.18:8080和192.168.31.18:9080。

### 5.5 修改windows以及linux中的hosts

由于tomcat以及nginx都安装在了linux中，而我是在windows的浏览器中进行访问的，所以必须要修改windows的hosts文件，在windows的hosts文件中添加下图中所示的内容即可，下图中的192.168.31.18是linux系统的ip地址。



至于为什么要修改linux中的hosts文件，这个我在下面会讲到，其实不修改linux中的hosts文件也是可以的。如果要修改的话，步骤如下。首先通过vim /etc/hosts命令打开hosts文件，然后在hosts文件中加上下图中内容，保存退出即可。



### 5.6 配置nginx的conf文件

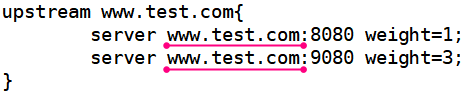
由于在一期的项目中，已经在/usr/local/nginx/conf目录下创建了vhost目录，且已经在/usr/local/nginx/conf目录下的nginx.conf文件中使用include包含了vhost目录中的所有文件，所以这里直接操作vhost目录中的文件即可。首先在vhost目录下新建一个www.test.com.conf文件，然后给该文件添加内容，添加的内容如下。



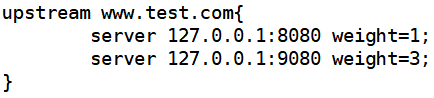
添加完成之后，保存退出。如果之前没有启动nginx的话，启动nginx。如果已经启动了的话，就在/usr/local/nginx/sbin目录下通过./nginx -s reload命令重启nginx。然后还可以通过./nginx -t命令测试是否重启成功。

Nginx启动成功后，由于之前已经启动了tomcat1和tomcat2，所以可以在windows的浏览器中访问www.test.com，然后不停刷新页面，如果之前对tomcat2的tomcat.png文件进行过修改的话，会发现出现的页面是不固定的。因为通过nginx会把请求随机分发给我们配置的对应8080端口的tomcat1以及对应9080端口的tomcat2。而tomcat2显示的页面中的图片又进行了修改，所以一直刷新的话，页面其实是在两个不同的页面之间进行切换。哪个页面出现的概率大，取决于上面conf文件中配置的权重。

这里说一下为什么要修改linux中的hosts文件。www.test.com.conf文件内容的部分截图如下。



该图中的server后面之所以可以使用www.test.com:8080或www.test.com:9080，就是因为在linux中修改了hosts文件。如果没有修改hosts文件的话，这里这样写是访问不到内容的。如果不想修改linux中的hosts文件也行，不过就不能再像上图中那么写了，而是要写成下图中这样。



# 五、XXXXXXXXXXXXXXXXX