**http://www.cnblogs.com/rocomp/p/4802396.html**

**一、常见JavaWeb服务器**

**1、WebLogic：是BEA公司的产品、WebSphereAS：是IBM公司的产品、JBossAS：红帽公司的产品，可以自行了解**

**2、Tomcat服务器：**

**Sun公司在推出的小型Servlet/JSP调试工具的基础上发展起来的一个优秀的Servlet容器，Tomcat本身完全用java语言编写，所以tomcat的运行需要java的支持，所以要先安装JDK，才能运行。目前是Apache开源软件组织的一个软件项目，它的官网**[**http://tomcat.apache.org**](http://tomcat.apache.org/)**。得到了广大开源代码志愿者的大力支持，且可以和目前大部分的主流Web服务器（IIS、Apache服务器）一起工作，它运行稳定、可靠且高效。已成为目前开发企业JavaWeb应用的最佳Servlet容器选择之一。目前最新版本8.0**

**二、Tomcat的目录结构及常用配置**

**1、Tomcat的目录层次结构**

**bin:   存放启动和关闭Tomcat的脚本文件**

**conf： 存放Tomcat服务器的各种配置文件**

**lib:   存放tomcat服务器支撑的jar包**

**logs:  存放Tomcat的日志文件**

**temp:  存放Tomcat运行时产生的临时文件**

**webapps:web应用虽在目录，即供外界访问的web资源的存放目录**

**work:  Tomcat的工作目录**

**2、JavaWeb应用的组成结构**

**mail---------------------------Web应用所在目录**

**|----html、jsp、css、js等文件，根目录下的文件外界可以直接访问**

**|----WEB-INF目录**

**|---------classes目录(java类)**

**|---------lib目录(java类运行所需的jar包)**

**|---------web.xml(web应用的配置文件)**

**WEB-INF 这个目录下的文件外界无法直接访问，由web服务器负责调用**

**3、JavaWeb应用的发布**

**开放式目录方式：**

**打包war方式:   使用   Jar -cvf \*.war .**

**4、Tomcat的组成结构**

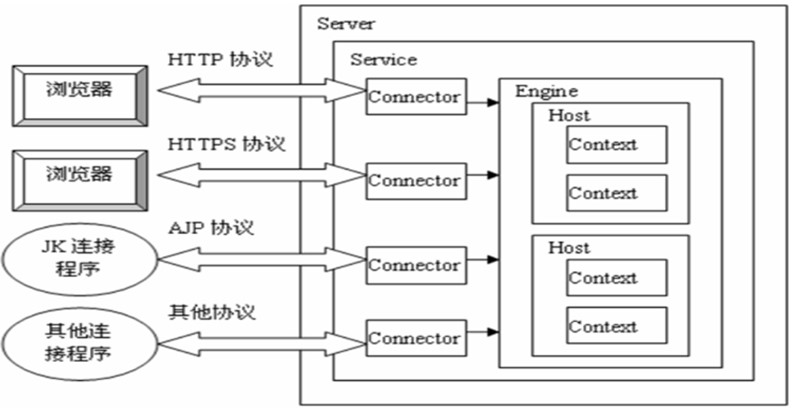
**Tomcat本身由一系列可配置的组件构成，其中核心组件是Servlet容器组件，它是所有其他Tomcat组件的顶层容器。每个组件都可以在Tomcat安装目录/conf/server.xml文件中进行配置，每个Tomcat组件在server.xml文件中对应一种配置元素。下面用XML的形式展示各种Tomcat组件之间的关系**

1. [复制代码](javascript:void(0);)
2. <Server>代表整个Servlet容器组件，是最顶层元素，可以包含一个或多个<Service>元素
3. <Service>包含一个<Engine>元素以及一个或多个<Connector>元素，这些<Connector>共享一个<Engine>
4. <Connector/>代表和客户程序实际交互的组件，负责接收客户请求，以及向客户返回响应
5. <Engine>每个<Service>元素只能包含一个<Engine>元素，它处理在同一个<Service>中所有<Connector>接收到的客户请求
6. <Host>在一个<Engine>中可以包含多个<Host>,它代表一个虚拟主机(即一个服务器程序可以部署在多个有不同IP的服务器主机上)，它可以包含一个或多个应用
7. <Context>使用最频繁的元素，代表了运行在虚拟主机上的单个web应用
8. </Host>
9. </Engine>
10. </Service>

</Server>

[复制代码](javascript:void(0);)

**5、Tomcat体系结构**



**6、虚拟目录的映射方式**

**从Tomcat6开始，Tomcat支持自动映射，即tomcat服务器会自动管理webapps目录下的所有web应用，并把它映射成虚拟目录，换句话说，只需把web应用放在webapps目录下，不需要配置Context，外界可以直接访问**

**但是如果你不想讲项目放在webapps下，而是放在其他地方，还是需要配置Context，比如我的tomcat在C盘，我的项目在D盘,则需要在Host下增加如下虚拟目录映射语句**

1. <Contextpath="/myapp"docBase="D:/work/myapp"></Context>

**7、Context元素配置**

**tomcat在加载一个web应用时，会一次按照以下五种方式查找web应用中的<Context>元素，优先级一次降低，直到找到为止**

**1）：到Tomcat安装目录/conf/Context.xml文件中查找<Context>元素。**

**2）：到Tomcat安装目录/conf/[enginename]/[hostname]/context.xml.default文件中查找<Context>元素。**

**[enginename]:表示<Engine>的name属性**

**[hostname]:表示<Host>d的那么属性**

**3）：到Tomcat安装目录/conf/[enginename]/[hostname]/[contextpath].xml文件中查找<Context>元素**

**[contextpath]:表示单个Web应用的URL入口**

**4）：到Web应用的META-INF/context.xml文件中查找<Context>元素**

**5）：到Tomcat安装目录/conf/server.xml文件中查找<Context>元素。只适用于单个Web应用**

**8、将项目部署为Tomcat默认应用**

**即访问**[**http://localhost:8080**](http://localhost:8080/)**时出来的是tomcat自带的欢迎页面，改为登录到自己的项目主页**

**方法一：**

**在Tomcat默认安装后，tomcat的主目录是webapps/root目录，所以如果想改变tomcat的主目录的话可以如下所做：**

**在/conf/server.xml文件的<Host></Host>之间加入代码**

1. <Contextpath=""docBase="C:\tomcat7\webapps\myapp"reloadable="true"debug="0"></Context>
2. <!--
3. docBase改为自己需要的项目路径
4. 如果建立了Apache和tomcat集群，Apache server 的默认端口是80 ，IE访问的方法只需输入：http://localhost,就可以自动定位到xx工程下面去
5. -->

**方法二：**

**将tomcat安装目录下的ROOT下的所有文件删除，换成自己项目的文件，此法有点暴力**

**方法三：**

**Tomcat5.0以下版本在C:/Tomcat/conf/Catalina/localhost目录下会自动生成了一个ROOT.Xml，但是5.0以上版本不再生成此文件，所以可以新建个ROOT.xml,在里面加入如下代码：**

1. <?Xml version='1.0' encoding='utf-8'?>
2. <ContextcrossContext="true"docBase="C:\tomcat7\webapps\myapp"path=""reloadable="true">
3. </Context>
4. <!--但是我自己在7.0版本上测试好像未成功，大家可以自己试试-->

**9、Context元素的属性**

**path：    指定访问该Web应用的URL入口**

**docBase： 指定Web应用的文件路径，可以写绝对路径，也可以写相对于appBase属性的相对路径**

**className：指定实现Context组件的Java类的名字，这个类必须实现org.apache.catalina.Contex接口，该属性默认值为org.apache.catalina.core.StandardContext（不建议大家改动）**

**reloadable：如果为true，Tomcat服务器在运行状态下回监视在WEB-INF/classes和WEB-INF/lib目录下的class文件的改动，以及监视Web应用的WEB-INF/web.xml文件的改动，如果检测到有更新，服务器会自动更新加载web应用。其默认值为false。在开发和调试阶段，将其改为true，一般像Eclipse等开发环境都会默认改为true。在正式发布阶段，应将其该为false，可以降低Tomcat的运行负荷，提高Tomcat的运行性能**

**在一般情况下，<Context>元素都会使用默认的标准Context组件，即className属性采用默认值org.apache.catalina.core.StandardContext，它除了拥有上面介绍到的属性外，还有自身专有的属性：**

**cachingAllowed：是否允许启用静态资源(HTML、图片、声音等)的缓存。默认值为true。**

**cacheMaxSize：设置静态资源缓存的最大值，单位为K。**

**workDir：指定Web应用的工作目录。**

**uppackWAR：如果为true，会把war文件展开为开放目录后再运行。为false，直接运行war文件。默认值为true。**

**10、web.xml文件**

**该文件必须放在/WEB-INF目录下，但是从7.0版本开始，Tomcat可以不使用web.xml文件，而是使用注解方式。**

**三、Tomcat的优化**

**1、内存使用配置**

**Tomcat是依赖于JVM的，所以Tomcat的使用内存配置实质上是JVM的内存配置**

**在/bin目录下的catalina.bat可以直接通过Tomcat设置JVM内存参数,windows下**

**打开catalina.bat文件，在大概中间的位置，找到**

**set "JAVA\_OPTS=%JAVA\_OPT% -server -Xms2048m -Xmx2048m -XX:PermSize=256m -XX:MaxPermSize=512m -Djava.awt.headless=true"**

**通过内存设置充分利用服务器内存**

**-server：   模式启动应用慢，但是极大程度提高运行性能**

**-Xms2048m -Xmx2048m：   设置的是堆内存**

**java8开始，PermSize被MetaspaceSize代替，MetaspaceSize共享heap，不会再有java.lang.OutOfMemoryError：PermGen space，可以不设置**

**Headless=true：   适用于Linux系统，与图形操作有关，如生成验证码含义是当前的是无显示器的服务器，应用中如果获取系统显示有关的参数会抛出异常，windows系统可不用设置**

**可通过jmap -heap process\_id查看设置是否成功**

**2、最大连接数配置（并发能力）**

**在/conf/server.xml里进行配置**

**在<Connector>属性中**

**将protocol属性改为 "org.apache.coyote.http11.Http11NioProtocol",NIO模型的效率更高**

**加入**

**maxThreads="500" //当前可以同时处理的最大用户访问数**

**minSpareThreads="100" //最小空闲线程连接数，用于优化线程池**

**maxSpareThreads="200"**

**acceptCount="200" //当所有的线程以分配，仍然允许连接进来，但是出于等待状态的用户数。等待线程数+工作线程数=总的可最大连接数,如果超过此数，新的连接将不会被接受，就会产生一个http错误**

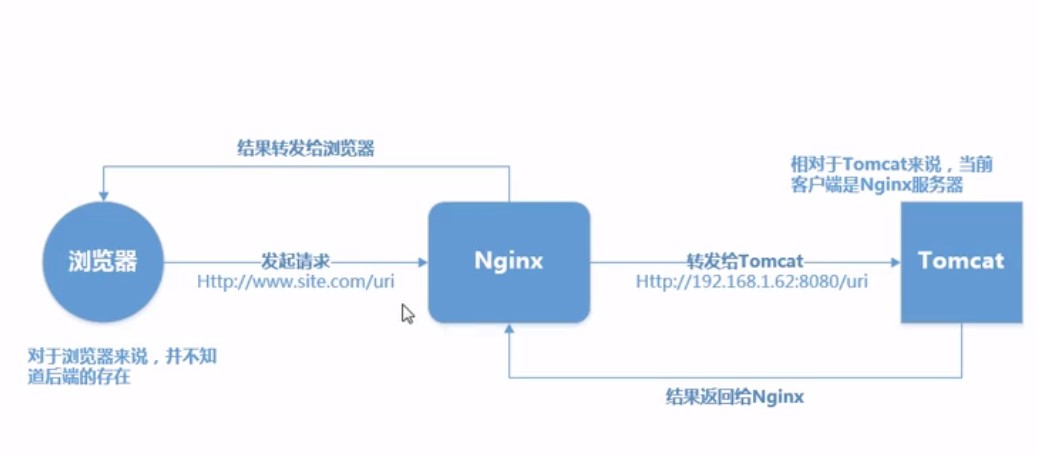
**enableLookups="false"//是否允许DNS反查，如果为true，request.getRemoteHost会执行DNS查找，反向解析ip对应域名或主机名，当没有这样需要的时候，可以将这个功能关闭，在一定程度上提高了Tomcat服务器的性能**

**四、负载均衡配置**

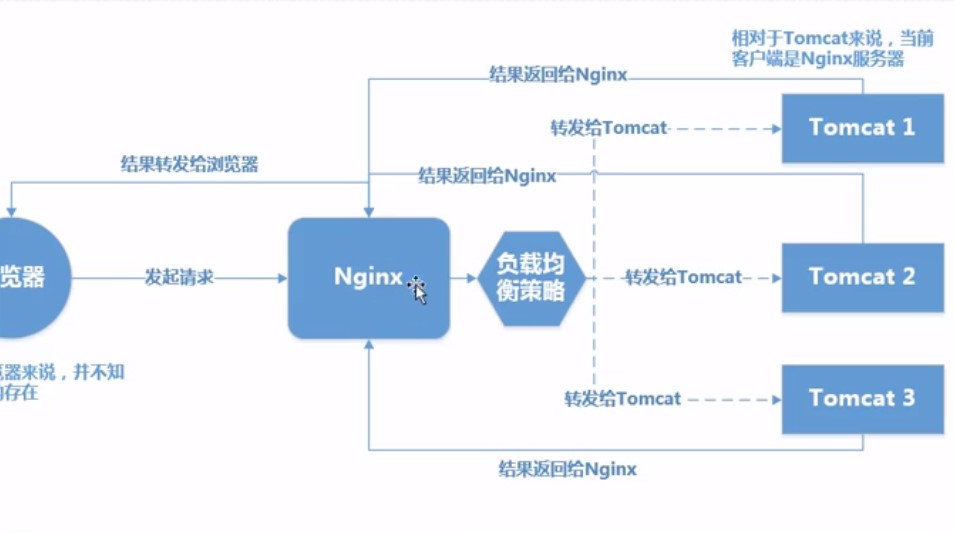
**负载均衡是提高应用负载量和容错的有效手段，可以结合Nginxs与Tomcat实现，Nginx是一款性能优异的反向代理服务器**

**1、负载均衡配置实现**

**反向代理原理示意图：**



**负载均衡示意图：**



**负载均衡策略：**

**Nginx 提供轮询（round robin）、用户IP哈希（client IP）和指定权重 3 种方式**

**2、负载均衡时Session的处理策略**

**Session是Tomcat服务器上的内存空间，如果一个用户发出多个请求，却发到了多个tomcat服务器中，那么就会出现Session不同步的问题**

**解决方案1：将一个用户的请求锁定到某一台服务器上，简单，但是缺乏容错性，一旦某个服务器发生故障，Session可能丢失，（**但是服务器发生故障是一个低概率事件，如果一个服务器经常挂掉，要么是硬件有问题，要么是应用有问题**）**

**可以使用用户IP哈希实现**

**解决方案2：Session复制策略，基于网络的广播策略，一个节点Session变化，其他节点同步复制，具有容错性，但节点多或复制量大时对网络负荷大，使网络效率低下，甚至阻塞**

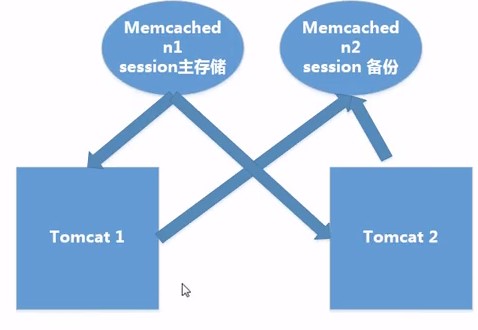
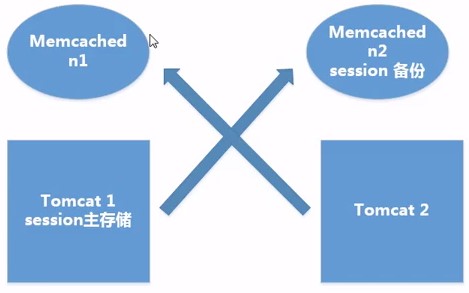
**① 在/conf/server.xml文件中开启Session复制的选项，将<Cluster>注释去掉，7.0默认值以配置好，需要接收器<Receiver>绑定内网(一般服务器都有两块网卡)网卡地址,修改端口>。**

**② 在应用中指定应用是在分布式部署之下，在web.xml中添加<distributable/>选项**

**解决方案3：额外创建一个共享的空间用来存放Session，所有服务器共享一个Session**

**memcached缓存共享方案基本原理**

**粘性Session与非粘性Session方式：**



**需要一些工具jar包，官网有目录和下载连接**

**在/conf/context.xml中进行配置，全局有效**

**3、集群环境中应用代码应注意的问题**

**① 传递Session需要实体类序列化支持，实现可序列化接口，设置版本号**

**② 获取用户IP地址方法的变化，获取真实客户端ip而不是Nginx代理地址**

**③ 动静分离结构的预规划，**

**一般高并发的网站上，资源类的文件，如js,css,图片通常是由静态服务器处理，Nginx处理静态文件效率就非常高，而tomcat处理静态文件是它很大的一个弱势，tomcat只负责动态请求的处理。那么编码的时候就要考虑静态资源最后可能要被拿出去，地址会有变化，所以在开发的时候应该规划访问地址，保持最大灵活性。**