**Tomcat应用服务**

目前来说IBM的WebSphere，Oracle的Weblogic占据了市场上Java语言Web站点的部分份额，该两种软件由于无与伦比的性能及可靠性等优势被广泛应用于大型互联网公司的Web场景中，但是其高昂的价格也使得中小型互联网公司对此望而却步。

Tomcat自5.x版本以来，其性能上已经得到很大幅度的提升，加上其开放性的框架和二次开发等特性，已经完全可以用在访问量不是很大的生产环境下，目前大多数用于JSP技术开发的电子商务网站基本上都应用了Tomcat。

## Tomcat软件简介

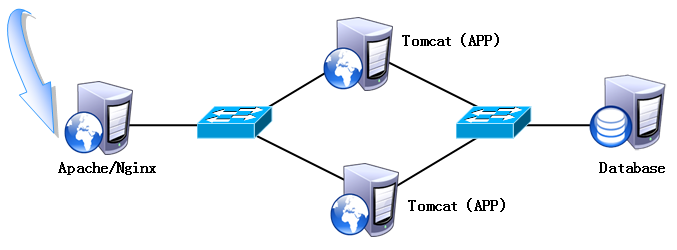
Tomcat是Apache软件基金会（Apache Software Foundation）的Jakarta 项目中的一个核心项目，由Apache、Sun和其他一些公司及个人共同开发而成。Tomcat最初是由Sun的软件构架师詹姆斯·邓肯·戴维森开发的。后来他将其变为开源项目，并由Sun贡献给Apache软件基金会。由于大部分开源项目O'Reilly都会出一本相关的书，并且将其封面设计成某个动物的素描，因此他希望将此项目以一个动物的名字命名。因为他希望这种动物能够自己照顾自己，最终，他将其命名为Tomcat（公猫）。而O'Reilly出版的介绍Tomcat的书籍的封面也被设计成了一个公猫的形象。而Tomcat的Logo兼吉祥物也被设计成了一只公猫。

其实Tomcat最早在开始研发的时候并不叫这个名字，早期的Tomcat的Logo项目的名字叫Catalina，所以当我们安装完Tomcat后会发现安装路径下有很多Catalina有关的目录和文件。这些文件是我们配置或使用Tomcat的重要文件所在。

## Tomcat应用场景

Tomcat服务器是一个免费的开放源代码的web应用服务器，属于轻量级应用服务器，应用在中小型系统和并发访问用户不是很多的场合，是开发和调试JSP页面的首选，Tomcat也可处理静态的HTML页面但是能力不及Apache或Nginx，所以Tomcat通常作为一个servlet(在[服务器](https://baike.so.com/doc/4487696-4696885.html" \t "_blank)上运行的小程序)和JSP容器，单独运行在后端。

Tomcat官网： [http://tomcat.apache.org](http://tomcat.apache.org/)



## JDK软件简介

在安装Tomcat之前必须先安装JDK，JDK全称是Java Development Kit，是SUN公司免费提供的java语言的软件开发工具包，其中包含Java虚拟机（JVM），编写好的java源程序经过编译可生产java字节码，只要安装了JDK，就可以利用JVM解释这些字节码文件，从而保证了Java的跨平台性。

在平台兼容性方面，JDK作为解释字节码文件并根据此调用操作系统的API实现对应功能的java虚拟机，与操作系统位数密切相关，因此存在不同类型的版本，Tomcat也具有上述特征，CentOS7系统中默认已经安装了JDK。

# 安装配置Tomcat（112上安装）

安装时候选择tomcat软件版本要与程序开发使用的版本一致。jdk版本要与tomcat保持一致。

1、所有主机关闭防火墙和selinux：

**[root@localhost ~]# iptables -F**

**[root@localhost ~]# setenforce 0**

**setenforce: SELinux is disabled**

**[root@localhost ~]# systemctl stop firewalld**

2、查看JDK是否安装

**[root@localhost ~]# java -version**

**openjdk version "1.8.0\_161" //这是系统自带的rpm方式安装**

**OpenJDK Runtime Environment (build 1.8.0\_161-b14)**

**OpenJDK 64-Bit Server VM (build 25.161-b14, mixed mode)**

3、卸载rpm方式安装的jdk

方法一：

**[root@localhost ~]# which java**

**/usr/bin/java**

**[root@localhost ~]# rm -rf /usr/bin/java**

方法二：

**[root@localhost ~]# rpm -qa | grep -i openjdk**

**java-1.7.0-openjdk-1.7.0.171-2.6.13.2.el7.x86\_64**

**java-1.7.0-openjdk-headless-1.7.0.171-2.6.13.2.el7.x86\_64**

**java-1.8.0-openjdk-1.8.0.161-2.b14.el7.x86\_64**

**java-1.8.0-openjdk-headless-1.8.0.161-2.b14.el7.x86\_64**

**[root@localhost ~]# rpm -e java-1.7.0-openjdk**

**[root@localhost ~]# rpm -e java-1.7.0-openjdk-headless**

**[root@localhost ~]# rpm -e java-1.8.0-openjdk --nodeps**

**[root@localhost ~]# rpm -e java-1.8.0-openjdk-headless**

**[root@localhost ~]# rpm -qa | grep -i openjdk**

也可以不卸载，只要把新安装的java命令搜索路径置于系统默认java路径的前面就可以

4、JDK安装

**[root@localhost ~]# tar xf jdk-8u91-linux-x64.tar.gz**

**[root@localhost ~]# mv jdk1.8.0\_91/ /usr/local/java**

**[root@localhost ~]# vim /etc/profile**

**export JAVA\_HOME=/usr/local/java**

**export PATH=$JAVA\_HOME/bin:$PATH**

**export TOMCAT\_HOME=/usr/local/tomcat8**

**export PATH=$TOMCAT\_HOME/bin:$PATH**

**#在PATH环境变量中添加java跟目录的bin子目录,置于默认PATH路径之前**

**[root@localhost ~]# source /etc/profile**

**[root@localhost ~]# java -version**

**java version "1.8.0\_91"**

**Java(TM) SE Runtime Environment (build 1.8.0\_91-b12)**

**Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)**

5、解压apache-tomcat-8.5.16.tar.gz 包

**[root@localhost ~]# tar xf apache-tomcat-8.5.16.tar.gz**

6、解压后生成apache-tomcat-8.5.16文件夹，将该文件夹移动到/usr/local下，并改名为tomcat8

**[root@localhost ~]# mv apache-tomcat-8.5.16 /usr/local/tomcat8**

7、启动Tomcat

**[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh**

**Using CATALINA\_BASE: /usr/local/tomcat8**

**Using CATALINA\_HOME: /usr/local/tomcat8**

**Using CATALINA\_TMPDIR: /usr/local/tomcat8/temp**

**Using JRE\_HOME: /usr**

**Using CLASSPATH: /usr/local/tomcat8/bin/bootstrap.jar:/usr/local/tomcat8/bin/tomcat-juli.jar**

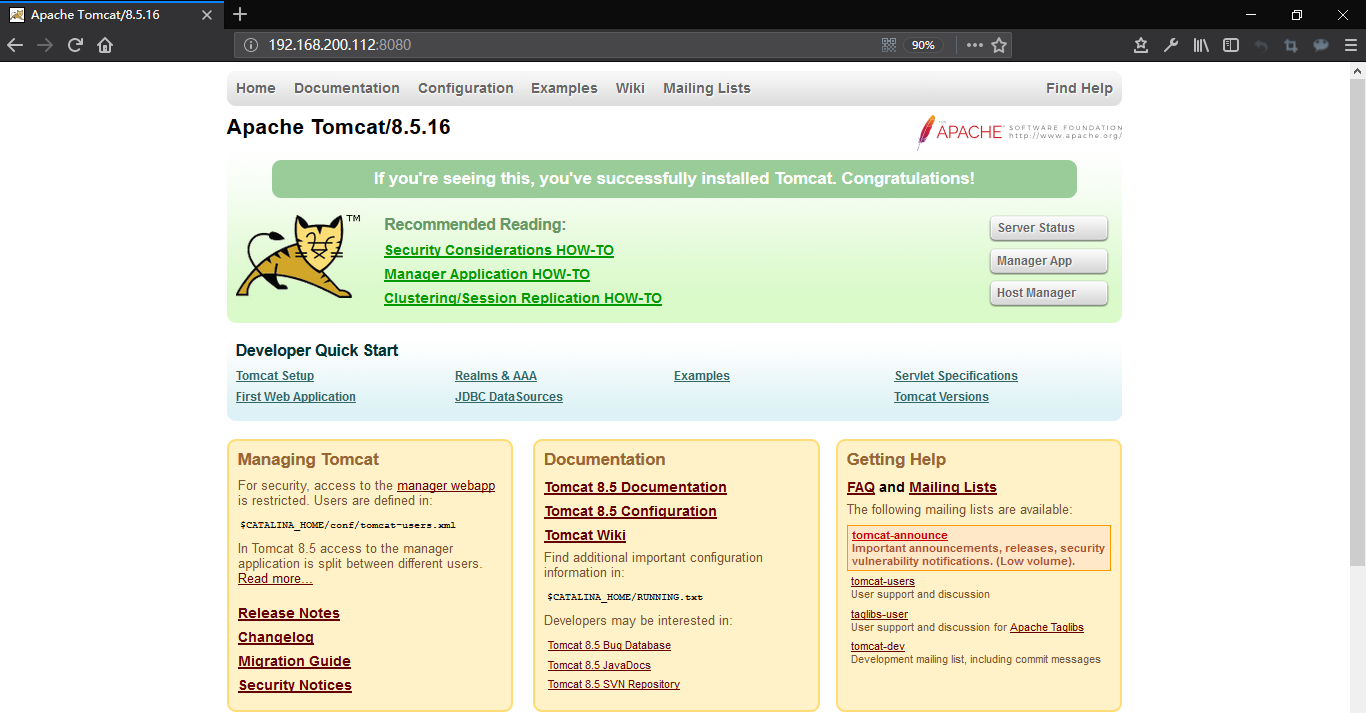
**Tomcat started.**

8、Tomcat 默认运行在8080端口

**[root@localhost ~]# netstat -lnpt | grep :8080**

**tcp6 0 0 :::8080 :::\* LISTEN 1533/java**

9、浏览器访问测试 http://192.168.200.112:8080



10、关闭Tomcat

**[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh**

# Tomcat 配置目录及文件说明

**[root@localhost ~]# ll /usr/local/tomcat8/**

**总用量 92**

**drwxr-x--- 2 root root 4096 1月 21 22:53 bin**

**drwx------ 3 root root 254 1月 21 22:54 conf**

**drwxr-x--- 2 root root 4096 1月 21 22:53 lib**

**-rw-r----- 1 root root 57092 6月 22 2017 LICENSE**

**drwxr-x--- 2 root root 197 1月 21 22:54 logs**

**-rw-r----- 1 root root 1723 6月 22 2017 NOTICE**

**-rw-r----- 1 root root 7064 6月 22 2017 RELEASE-NOTES**

**-rw-r----- 1 root root 15946 6月 22 2017 RUNNING.txt**

**drwxr-x--- 2 root root 30 1月 21 22:53 temp**

**drwxr-x--- 7 root root 81 6月 22 2017 webapps**

**drwxr-x--- 3 root root 22 1月 21 22:54 work**

## tomcat主目录介绍

bin //存放启动或关闭Tomcat的脚本文件

conf //存放Tomcat的各种全局配置文件，其中最主要的是server.xml和web.xml

lib //存放Tomcat运行需要的库文件(JARS)

logs //存放Tomcat执行时的LOG文件

webapps //Tomcat的主要Web发布目录、类似于nginx的html（包括应用程实例）

work //存放jsp编译后产生的.class文件

temp //存放临时文件

## webapps目录介绍

docs //tomcat帮助文档

examples //web应用实例

host-manager //主机管理

manager //管理

ROOT //默认站点根目录

**[root@localhost ~]# ll /usr/local/tomcat8/conf/**

**总用量 224**

**drwxr-x--- 3 root root 23 1月 21 22:54 Catalina**

**-rw------- 1 root root 13816 6月 22 2017 catalina.policy**

**-rw------- 1 root root 7376 6月 22 2017 catalina.properties**

**-rw------- 1 root root 1338 6月 22 2017 context.xml**

**-rw------- 1 root root 1149 6月 22 2017 jaspic-providers.xml**

**-rw------- 1 root root 2358 6月 22 2017 jaspic-providers.xsd**

**-rw------- 1 root root 3622 6月 22 2017 logging.properties**

**-rw------- 1 root root 7511 6月 22 2017 server.xml**

**-rw------- 1 root root 2164 6月 22 2017 tomcat-users.xml**

**-rw------- 1 root root 2633 6月 22 2017 tomcat-users.xsd**

**-rw------- 1 root root 168251 6月 22 2017 web.xml**

catalina.policy //权限控制配置文件

catalina.properties //Tomcat属性配置文件

context.xml //context用于指定额外的web目录

logging.properties //日志log相关配置文件

server.xml //主配置文件

tomcat-users.xml //manager-gui管理用户配置文件（Tomcat安装后生成的管理界面，该文件可开启访问）

web.xml //Tomcat的servlet，servlet-mapping，filter，MIME等相关配置

## Tomcat主配置文件说明

server.xml 主要配置文件，可修改启动端口，设置网站根目录，虚拟主机，多实例、开启https加密等功能。

server.xml的结构构成：

<Server>

<Service>

<Connector />

<Engine>

<Host>

<Context> </Context>

</Host>

</Engine>

</Service>

</Server>

<!-- --> 内的内容是注视信息

//tomcat关闭端口，默认只对本机地址开放，可以在本机通过telnet 127.0.0.1 8005访问，对Tomcat进行关闭操作。

**<Server port="8005" shutdown="SHUTDOWN">**

Tomcat在server.[xml](https://so.csdn.net/so/search?q=xml&spm=1001.2101.3001.7020)中配置了两种连接器。  
HTTP Connector  
拥有这个连接器，Tomcat才能成为一个web服务器。  
AJP Connector  
AJP连接器可以通过AJP协议和另一个web容器进行交互

//tomcat启动的默认端口号8080，可以根据需要进行修改

**<Connector port="8080" protocol="HTTP/1.1"**

**connectionTimeout="20000"**

**redirectPort="8443" />**

//tomcat启动AJP1.3连接器时默认的端口号，可以根据需要进行修改

**<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />**

//以下是tomcat定义虚拟主机时的配置及日志配置

**<Host name="localhost" appBase="webapps"**

**unpackWARs="true" autoDeploy="true">**

**<Context docBase="/web/webapp" path="" reloadable="false" >**

**</Context>**

**<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"**

**prefix="localhost\_access\_log" suffix=".txt"**

**pattern="%h %l %u %t &quot;%r&quot; %s %b" />**

**</Host>**

## Server.xml主要参数说明

|  |  |
| --- | --- |
| 组件名称 | 功能介绍 |
| server | 表示一个运行于JVM中的tomcat实例。 |
| service（服务） | 将connector关联至engine，因此一个service内部可以有多个connector，但只能有一个引擎engine，负责处理所有Connector所获得的客户请求。service内部有两个connector，一个engine。因此，一般情况下一个server内部只有一个service，一个service内部只有一个engine，但一个service内部可以有多个connector。 |
| connector | 接收用户请求，类似于httpd的listen配置监听端口的。  一个Connector在某个指定端口上侦听客户请求，并将获得的请求交给Engine来处理，从Engine处获得回应并返回客户。  Tomcat Engine有两个典型的Connector，一个直接侦听来自browser的http请求，一个侦听来自其他webserver的请求  Connector在端口8080处侦听来自客户Browser的http请求。  Connector在端口8009处侦听来自其他webserver（Apache）的servlet/jsp代理请求。 |
| engine | 核心容器组件，catalina引擎，负责通过connector接收用户请求，并处理请求，将请求转至对应的虚拟主机host。  Engine有一个默认的虚拟主机，当请求无法匹配到任何一个Host上的时候，将交给该默认Host来处理。 |
| host | 代表一个Virtual Host，虚拟主机，每个虚拟主机和某个网络域名Domain Name相匹配  每个虚拟主机下都可以部署（deploy）一个或者多个Web app，每个web app 对应一个Context，有一个Context path。  当Host获得一个请求时，将把该请求匹配到某个Context上，然后把该请求交给该Context来处理，匹配的方法是最长匹配，所以一个path==“”的Context将成为该Host的默认Context匹配。  所有无法与其他Context的路径名匹配的请求都将最终和该默认Context匹配。 |
| context | 定义一个应用程序，是一个最内层的容器类组件（不能再嵌套）。配置context的主要目的指定对应对的webapp的根目录，类似于httpd的alias，其还能为webapp指定额外的属性，如部署方式等。 |
| realm | 可以用于任意容器类的组件中，关联一个用户认证库，实现认证和授权。可以关联的认证库有两种：UserDatabaseRealm、MemoryRealm和JDBCRealm。 |
| UserDatabaseRealm | 使用JNDI自定义的用户认证库。 |
| MemoryRealm | 认证信息定义在tomcat-users.xml中。 |
|  |  |

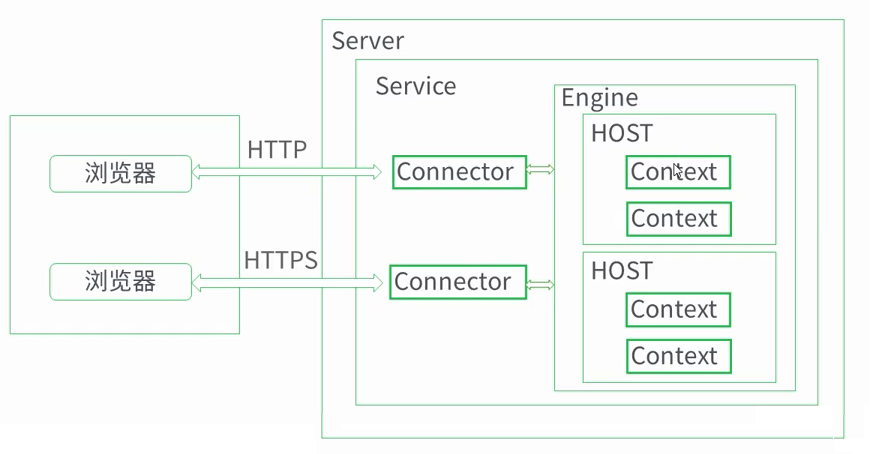
## host参数详解

|  |  |
| --- | --- |
| 参数 | 参数说明 |
| host | 表示一个虚拟主机 |
| name | 指定主机名（域名） |
| appBase | 应用程序基本目录，即存放应用程序的目录.一般为appBase="webapps" ，相对于CATALINA\_HOME而言的，也可以写绝对路径。 |
| unpackWARs | 如果为true，则tomcat会自动将WAR文件解压，否则不解压，直接从WAR文件中运行应用程序 |
| autoDeploy | 在tomcat启动时，是否自动部署。在webapps目录中增加新的目录、war文件、修改WEB-INF/web.xml，autoDeploy="true"会新建或重新部署应用，该选项方便部署 |

## Context参数说明

|  |  |
| --- | --- |
| 参数 | 参数说明 |
| Context | 表示一个web应用程序加载位置，通常为WAR文件位置 |
| docBase | 应用程序的路径或者是WAR文件存放的路径,也可以使用相对路径，起始路径为此Context所属Host中appBase定义的路径。 |
| path | 表示此web应用程序的url的前缀，这样请求的url为http://localhost:8080/path/\*\*\*\* |
| reloadable | 这个属性非常重要，如果为true，则tomcat会自动检测应用程序的/WEB-INF/lib 和/WEB-INF/classes目录的变化，自动装载新的应用程序，可以在不重启tomcat的情况下改变应用程序 |

## 请求处理流程



处理流程：用户发送请求到WEB服务器，该请求会被正在监听的Connector连接器接收，并把该请求交给Service下的Engine来处理，并等待Engine处理的结果。Engine获得请求后会根据请求的主机信息来匹配相应的Host主机，Host主机会根据请求的路径匹配对应的Context，Context web应用匹配上之后就构建request、response请求对象，调用指定的Servlet来处理请求。请求处理完成后会将response对象返回给Host主机，Host主机将response对象返回给Engine引擎，Engine再将response对象返回给Connector链接器，最后Connector连接器将response返回给浏览器。

# 自定义默认网站目录

首先在根目录下建立一个web目录，并在里面建立一个webapp目录，用于存放网站文件

**[root@localhost ~]# mkdir -pv /web/webapp**

**mkdir: 已创建目录 "/web"**

**mkdir: 已创建目录 "/web/webapp"**

在webapp目录下建立一个index.jsp的测试页面

**[root@localhost ~]# echo 192.168.200.112 > /web/webapp/index.jsp**

修改Tomcat的server.xml文件

定义一个虚拟主机，并将网站文件路径指向已经建立的/web/webapp，在host段增加context段

**[root@localhost ~]# cp /usr/local/tomcat8/conf/server.xml{,.-$(date +%F)}**

**[root@localhost ~]# vim /usr/local/tomcat8/conf/server.xml**

**<Host name="localhost" appBase="webapps"**

**unpackWARs="true" autoDeploy="true">**

**<Context docBase="/web/webapp" path="" reloadable="flase" >**

**</Context>**

重新启动tomcat服务

**[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh**

**Using CATALINA\_BASE: /usr/local/tomcat8**

**Using CATALINA\_HOME: /usr/local/tomcat8**

**Using CATALINA\_TMPDIR: /usr/local/tomcat8/temp**

**Using JRE\_HOME: /usr**

**Using CLASSPATH: /usr/local/tomcat8/bin/bootstrap.jar:/usr/local/tomcat8/bin/tomcat-juli.jar**

**[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh**

**Using CATALINA\_BASE: /usr/local/tomcat8**

**Using CATALINA\_HOME: /usr/local/tomcat8**

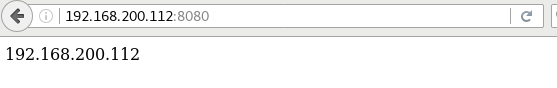
**Using CATALINA\_TMPDIR: /usr/local/tomcat8/temp**

**Using JRE\_HOME: /usr**

**Using CLASSPATH: /usr/local/tomcat8/bin/bootstrap.jar:/usr/local/tomcat8/bin/tomcat-juli.jar**

**Tomcat started.**

浏览器访问测试 <http://192.168.200.112:8080>



# Nginx+Tomcat 负载均衡集群（群集 Cluster）

## Nginx代理介绍

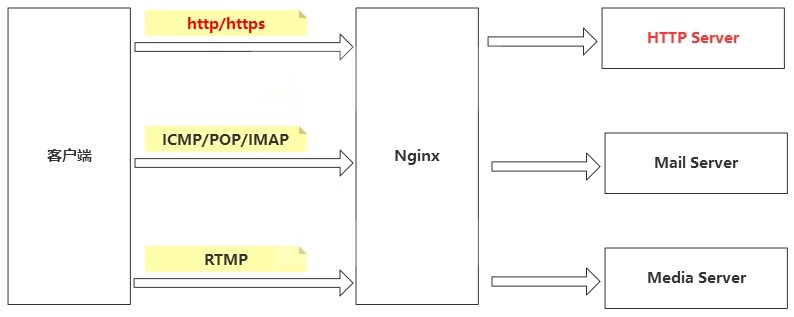
Nginx 服务器的反向代理服务是其最常用的重要功能，由反向代理服务也可以衍生出很多与此相关的 Nginx 服务器重要功能，比如后面会介绍的负载均衡。当然在了解反向代理之前，我们需要先知道什么是代理以及什么是正反向代理。

### 1、代理概述

举一个现实生活中的例子：比如我们要买一间二手房，虽然我们可以自己去找房源，但是这太花费时间精力了，而且房屋质量检测以及房屋过户等一系列手续也都得我们去办，再说现在这个社会，等我们找到房源，说不定房子都已经涨价了，那么怎么办呢？最简单快捷的方法就是找二手房中介公司（为什么？别人那里房源多啊），于是我们就委托中介公司来给我找合适的房子，以及后续的质量检测过户等操作，我们只需要选好自己想要的房子，然后交钱就行了。

　　代理简单来说，就是如果我们想做什么，但又不想直接去做，那么这时候就找另外一个人帮我们去做。那么这个例子里面的中介公司就是给我们做代理服务的，我们委托中介公司帮我们找房子。

　　Nginx 主要能够代理如下几种协议，其中用到的最多的就是做Http代理服务器。



### 2、正向代理

　　弄清楚什么是代理了，那么什么又是正向代理呢？

　　这里我再举一个例子：大家都知道，现在国内是访问不了 Google的，那么怎么才能访问 Google呢？我们又想，美国人不是能访问 Google吗（这不废话，Google就是美国的），如果我们电脑的对外公网 IP 地址能变成美国的 IP 地址，那不就可以访问 Google了。你很聪明，VPN 就是这样产生的。我们在访问 Google 时，先连上 VPN 服务器将我们的 IP 地址变成美国的 IP 地址，然后就可以顺利的访问了。

这里的 VPN 就是做正向代理的。正向代理服务器位于客户端和服务器之间，为了向服务器获取数据，客户端要向代理服务器发送一个请求，并指定目标服务器，代理服务器将目标服务器返回的数据转交给客户端。这里客户端是要进行一些正向代理的设置的（需要知道代理服务器是谁）。

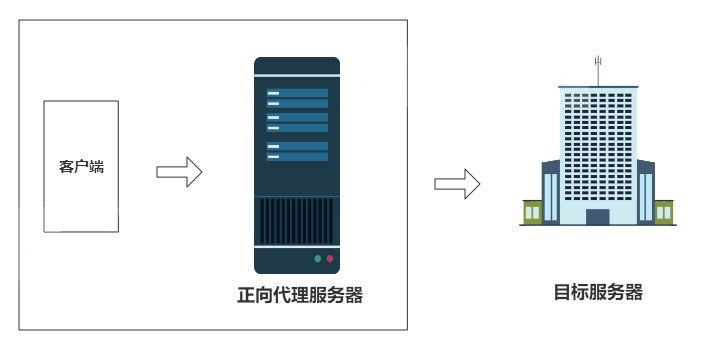
正向代理客户端一般是公司内网用户访问外网，客户端要设置代理服务器地址（比如公司网关），访问时访问的是外网真正的服务器地址（比如百度）

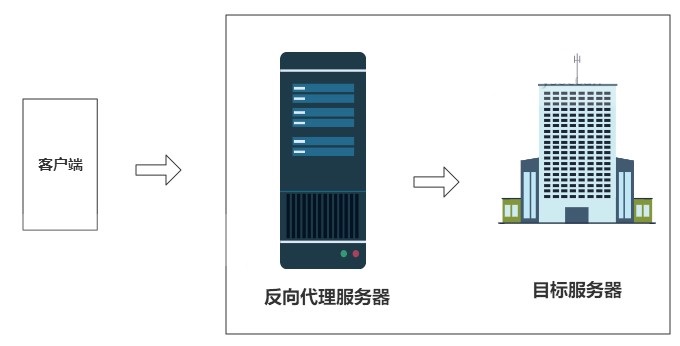
### 3、反向代理

反向代理，其实客户端对代理是无感知的，因为客户端不需要任何配置就可以访问，我们只需要将请求发送到反向代理服务器，由反向代理服务器去选择目标服务器获取数据后，再返回给客户端，此时反向代理服务器和目标服务器对外就是一个服务器，暴露的是代理服务器地址，隐藏了真实服务器IP地址。

反向代理客户端一般是外网用户，要访问公司（内网）的服务器，访问时直接访问的其实是代理服务器，但客户端并不知道有代理的存在，不直接访问公司的真正服务器。群集中的调度器一般是反向代理。

　　下面我们通过两张图来对比正向代理和方向代理：





　　理解这两种代理的关键在于代理服务器所代理的对象是什么，正向代理代理的是客户端，我们需要在客户端进行一些代理的设置。而反向代理代理的是服务器，作为客户端的我们是无法感知到服务器的真实存在的。

反向代理和正向代理的区别就是：正向代理代理客户端，反向代理代理服务器。

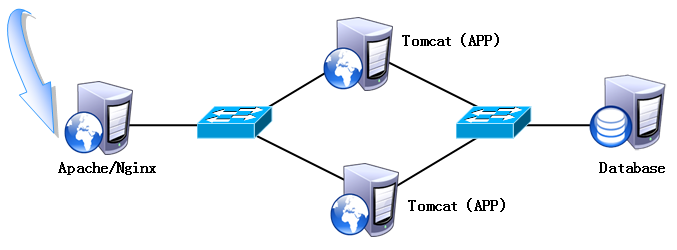
一台Tomcat站点由于可能出现单点故障及无法应付过多客户复杂多样的请求等问题，不能单独应用于生产环境下，所以需要一套可靠的解决方案来完善web站点架构。

Nginx 是一款非常优秀的http服务器软件，它能够支持高达50000个并发连接数的响应，拥有强大的静态资源处理能力，运行稳定，并且内存，CPU等系统资源消耗非常低，目前很多大型网站都用Nginx服务器做后端网站程序的反向代理及负载均衡器，来提升整个站点的负载并发能力。

Nginx 服务器 CentOS 7.5 x86\_64 192.168.200.111 nginx

Tomcat服务器1 CentOS 7.5 x86\_64 192.168.200.112 jdk tomcat

Tomcat服务器2 CentOS 7.5 x86\_64 192.168.200.113 jdk tomcat



Tomcat 2 配置方法基本与Tomcat 1 相同（可以将tomcat1的文件scp到tomcat2上再修改）

1）关闭防火墙，selinux

2）安装JDK，配置Java环境，版本保持一致

3）安装Tomcat，版本与Tomcat 1 保持一致

4）建立一个web目录，并在里面建立一个webapp目录，用于存放网站文件

5）在webapp目录下建立一个index.jsp的测试页面

**[root@localhost ~]# tar xf apache-tomcat-8.5.16.tar.gz**

**[root@localhost ~]# mv apache-tomcat-8.5.16 /usr/local/tomcat8**

**[root@localhost ~]# cp /usr/local/tomcat8/conf/server.xml{,.bak}**

**[root@localhost ~]# vim /usr/local/tomcat8/conf/server.xml**

**<Host name="localhost" appBase="webapps"**

**unpackWARs="true" autoDeploy="true">**

**<Context docBase="/web/webapp" path="" reloadable="flase" >**

**</Context>**

**[root@localhost local]# mkdir -p /web/webapp**

**[root@localhost ~]# echo 192.168.200.113 > /web/webapp/index.jsp**

**[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh**

**Using CATALINA\_BASE: /usr/local/tomcat8**

**Using CATALINA\_HOME: /usr/local/tomcat8**

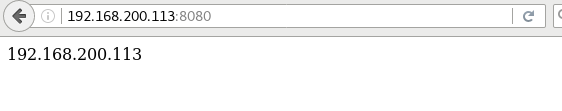
**Using CATALINA\_TMPDIR: /usr/local/tomcat8/temp**

**Using JRE\_HOME: /usr**

**Using CLASSPATH: /usr/local/tomcat8/bin/bootstrap.jar:/usr/local/tomcat8/bin/tomcat-juli.jar**

**Tomcat started.**

浏览器访问测试 http://192.168.200.113:8080



Nginx 服务器配置

**[root@nginx ~]# yum -y install pcre-devel zlib-devel openssl-devel lrzsz**

**[root@nginx ~]# useradd -M -s /sbin/nologin nginx**

**[root@nginx ~]# tar xf nginx-1.14.2.tar.gz -C /usr/src/**

**[root@nginx ~]# cd /usr/src/nginx-1.14.2/**

**[root@nginx nginx-1.14.2]# ./configure --prefix=/usr/local/nginx --user=nginx --group=nginx && make && make install**

**--prefix=/usr/local/nginx //指定安装目录**

**--user=nginx --group=nginx //指定运行的用户和组**

配置nginx.conf

**[root@nginx nginx-1.14.2]# cp /usr/local/nginx/conf/nginx.conf{,.bak}**

**[root@nginx nginx-1.14.2]# vim /usr/local/nginx/conf/nginx.conf**

**user nginx nginx;**

**worker\_processes 2;**

**error\_log logs/error.log;**

**pid logs/nginx.pid;**

**events {**

**use epoll;**

**worker\_connections 10240;**

**}**

**http {**

**include mime.types;**

**default\_type application/octet-stream;**

**log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '**

**'$status $body\_bytes\_sent "$http\_referer" '**

**'"$http\_user\_agent" "$http\_x\_forwarded\_for"';**

**access\_log logs/access.log main;**

**sendfile on;**

**keepalive\_timeout 65;**

**upstream tomcat {**

**server 192.168.200.112:8080 weight=1;**

**server 192.168.200.113:8080 weight=1;**

**}**

**server {**

**listen 80;**

**server\_name localhost;**

**location / {**

**root html;**

**index index.html index.htm;**

**}**

**location ~ \.jsp$ {**

**proxy\_pass http://tomcat;**

**}**

**error\_page 500 502 503 504 /50x.html;**

**location = /50x.html {**

**root html;**

**}**

**}**

**}**

**[root@nginx nginx-1.14.2]# /usr/local/nginx/sbin/nginx -t**

**nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok**

**nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful**

**[root@nginx nginx-1.14.2]# /usr/local/nginx/sbin/nginx**

**[root@nginx nginx-1.14.2]# netstat -anpt |grep :80**

**tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 4632/nginx: master**

**[root@nginx nginx-1.14.2]# ps aux | grep nginx | grep -v grep**

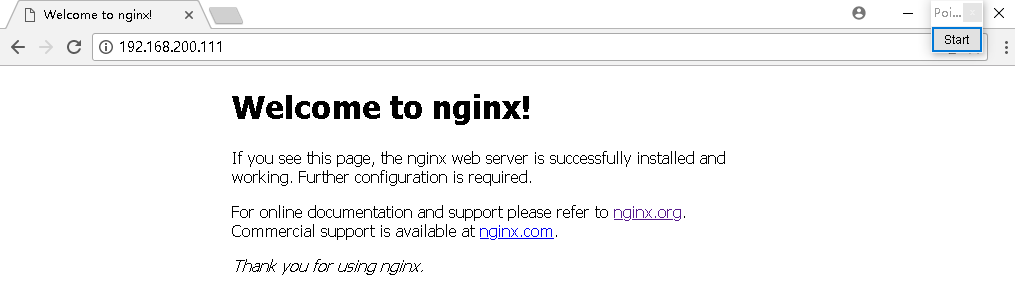
**root 4632 0.0 0.1 45936 1148 ? Ss 23:41 0:00 nginx: master process /usr/local/nginx/sbin/nginx -c /u**

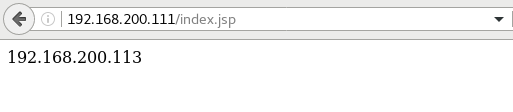
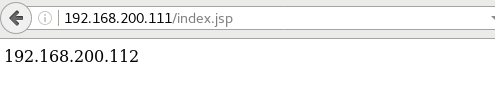
**sr/local/nginx/conf/nginx.confnginx 4633 0.0 0.5 52276 5708 ? S 23:41 0:00 nginx: worker process**

**nginx 4634 0.0 0.5 52276 5708 ? S 23:41 0:00 nginx: worker process**

客户端测试：

打开浏览器访问: <http://192.168.200.111>/index.jsp不断刷新可看到由于权重相同，页面会反复切换





## Nginx负载均衡算法

1、轮询（默认）

        每个请求按时间顺序逐一分配到不同的后端服务，如果后端某台服务器死机，自动剔除故障系统，使用户访问不受影响。

**upstream tomcat\_server {**

**server 192.168.200.112:8080 weight=1;**

**server 192.168.200.113:8080 weight=1;**

weight（轮询权值）

        weight的值越大分配到的访问概率越高，主要用于后端每台服务器性能不均衡的情况下。或者仅仅为在主从的情况下设置不同的权值，达到合理有效的地利用主机资源。

**upstream tomcat\_server {**

**server 192.168.200.112:8080 weight=1;**

**server 192.168.200.113:8080 weight=2;**

2、least\_conn

least-connected方式可以更公平的将负载分配到多个机器上面。使用least-connected，nginx不会将请求分发到繁忙的机器上面，而是将新的请求分发到较清闲的机器上面。

**upstream tomcat\_server {**

**least\_conn;**

**server 192.168.200.112:8080 weight=1;**

**server 192.168.200.113:8080 weight=1;**

3、ip\_hash

        每个请求按访问IP的哈希结果分配，使来自同一个IP的访客固定访问一台后端服务器，并且可以有效解决动态网页存在的session共享问题。

**upstream tomcat\_server {**

**ip\_hash;**

**server 192.168.200.112:8080 weight=1;**

**server 192.168.200.113:8080 weight=1;**

4、fair

        比 weight、ip\_hash更加智能的负载均衡算法，fair算法可以根据页面大小和加载时间长短智能地进行负载均衡，也就是根据后端服务器的响应时间来分配请求，响应时间短的优先分配。Nginx本身不支持fair，如果需要这种调度算法，则必须安装upstream\_fair模块。

**upstream tomcat\_server {**

**fair;**

**server 192.168.200.112:8080 weight=1;**

**server 192.168.200.113:8080 weight=1;**

5、url\_hash

        按访问的URL的哈希结果来分配请求，使每个URL定向到一台后端服务器，可以进一步提高后端缓存服务器的效率。Nginx本身不支持url\_hash，如果需要这种调度算法，则必须安装Nginx的hash软件包。

**upstream tomcat\_server {**

**hash $request\_uri;**

**hash\_method crc32;**

**server 192.168.200.112:8080;**

**server 192.168.200.113:8080;**

## Nginx负载均衡调度状态

  在Nginx upstream模块中，可以设定每台后端服务器在负载均衡调度中的状态，常用的状态有：

down：表示当前的server暂时不参与负载均衡。

backup：预留的备份机器。当其他所有的非backup机器出现故障的时候，才会请求backup机器，因此这台机器的访问压力最低。

max\_fails：允许请求失败的次数，默认为1，当超过最大次数时，返回proxy\_next\_upstream模块定义的错误。

fail\_timeout：请求失败超时时间，在经历了max\_fails次失败后，暂停服务的时间。max\_fails和fail\_timeout可以一起使用。

为了模拟添加一台服务器，可以在112上设置两个tomcat进程，只需要把service内容复制一份，然后更改service的名称、两个connector的端口号、host的根目录和主页内容即可（具体步骤见后面基于端口的虚拟主机）

[root@localhost ~]# netstat -anpt|grep 80

tcp6 0 0 127.0.0.1:8005 :::\* LISTEN 77592/java

tcp6 0 0 :::8009 :::\* LISTEN 77592/java

tcp6 0 0 :::8010 :::\* LISTEN 77592/java

tcp6 0 0 :::8080 :::\* LISTEN 77592/java

tcp6 0 0 :::8090 :::\* LISTEN 77592/java

**[root@localhost nginx-1.14.2]# vim /usr/local/nginx/conf/nginx.conf 修改为以下配置**

**upstream tomcat\_server {**

**server 192.168.200.112:8080 weight=1;**

**server 192.168.200.112:8090 weight=1;**

**server 192.168.200.113:8080 weight=1 backup;**

**}**

**[root@localhost nginx-1.14.2]# killall -HUP nginx**

**[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh**

客户端测试：

打开浏览器访问: <http://192.168.200.111/index.jsp不断刷新可看到由于112>的两个tomcat已经关闭，页面会切换到113的实例页面。



在nginx中HTTP, HTTPS, FastCGI, memcached的负载均衡都是通过反向代理实现的。

要配置https的负载均衡只需要将http协议改https就可以了，其他配置不变。

# Tomcat连接数据库

**[root@localhost ~]# tar xf SLSaleSystem.tar.gz -C /web/webapp/**

**[root@localhost ~]# ls /web/webapp/SLSaleSystem/**

**logs META-INF statics WEB-INF**

**[root@localhost ~]# vim /usr/local/tomcat8/conf/server.xml**

**<Context docBase="/web/webapp/SLSaleSystem" path="" reloadable="flase" >**

**</Context>**

**[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh**

**[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh**

浏览器访问：http://192.168.200.112:8080



192.168.200.114安装数据库

**[root@localhost ~]# yum -y install mariadb-server mariadb**

**[root@localhost ~]# systemctl start mariadb**

**[root@localhost ~]# mysql**

**Welcome to the MariaDB monitor. Commands end with ; or \g.**

**Your MariaDB connection id is 2**

**Server version: 5.5.56-MariaDB MariaDB Server**

**Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.**

**Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.**

**MariaDB [(none)]> create database slsaledb;**

**Query OK, 1 row affected (0.00 sec)**

**MariaDB [(none)]> grant all on slsaledb.\* to admin@'%' identified by '123456';**

**MariaDB [(none)]> grant all on slsaledb.\* to admin@'server02' identified by '123456';**

**Query OK, 0 rows affected (0.00 sec)**

**MariaDB [(none)]> flush privileges;**

**Query OK, 0 rows affected (0.00 sec)**

**[root@localhost ~]# mysql -uroot < slsaledb-2014-4-10.sql**

192.168.200.112上配置连接数据库

**[root@localhost ~]# vim /web/webapp/SLSaleSystem/WEB-INF/classes/jdbc.properties**

**driverClassName=com.mysql.jdbc.Driver**

**url=jdbc\:mysql\://192.168.200.114\:3306/slsaledb?useUnicode\=true&characterEncoding\=UTF-8**

**uname=admin**

**password=123456**

**minIdle=10**

**maxIdle=50**

**initialSize=5**

**maxActive=100**

**maxWait=100**

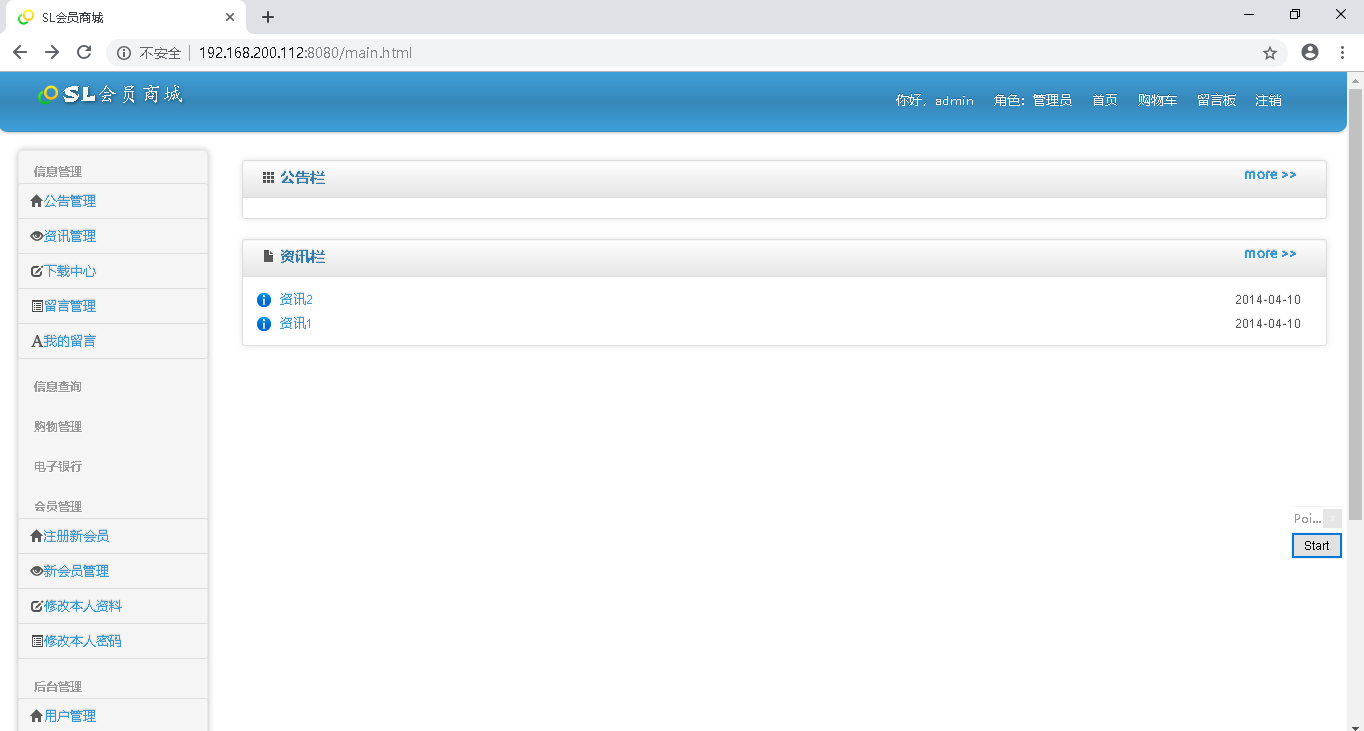
**removeAbandonedTimeout=180**

**removeAbandoned=true**

**[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh**

**[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh**

用户密码admin、123456，能成功登陆到会员商城



# Tomcat配置SSL加密

说明：当通过Nginx调度tomcat进行工作时，如果Nginx配置了ssl，Tomcat是无需配置的。

扩展实验：Nginx+OpenSSL实现https协议

创建加密用的私钥和证书文件

**[root@localhost ~]# keytool -genkeypair -alias tomcat -keyalg RSA -keystore /usr/local/tomcat8/keystore**

**输入密钥库口令:**

**再次输入新口令:**

**您的名字与姓氏是什么?**

**[Unknown]: benet**

**您的组织单位名称是什么?**

**[Unknown]: accp**

**您的组织名称是什么?**

**[Unknown]: accp**

**您所在的城市或区域名称是什么?**

**[Unknown]: beijing**

**您所在的省/市/自治区名称是什么?**

**[Unknown]: haidian**

**该单位的双字母国家/地区代码是什么?**

**[Unknown]: CN**

**CN=benet, OU=accp, O=accp, L=beijing, ST=haidian, C=CN是否正确?**

**[否]: y**

**输入 <tomcat> 的密钥口令**

**(如果和密钥库口令相同, 按回车):**

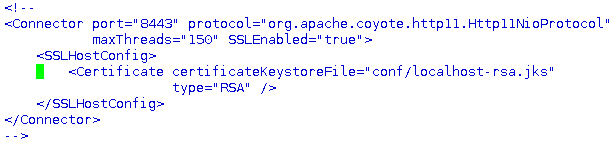
**再次输入新口令:**

**Warning:**

**JKS 密钥库使用专用格式。建议使用 "keytool -importkeystore -srckeystore /usr/local/tomcat8/keystore -destkeystore /usr/local/tomcat8/keystore -deststoretype pkcs12" 迁移到行业标准格式 PKCS12。**

修改server.xml配置文件，创建支持加密连接的Connector

**[root@localhost ~]# vim /usr/local/tomcat8/conf/server.xml**



将上述配置注释去掉并修改为如下配置

**<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"**

**maxThreads="150" SSLEnabled="true" scheme="https" secure="true" clientAuth="false" sslProtocol="TLS" keystoreFile="/usr/local/tomcat8/keystore" keystorePass="123456">**

**</Connector>**

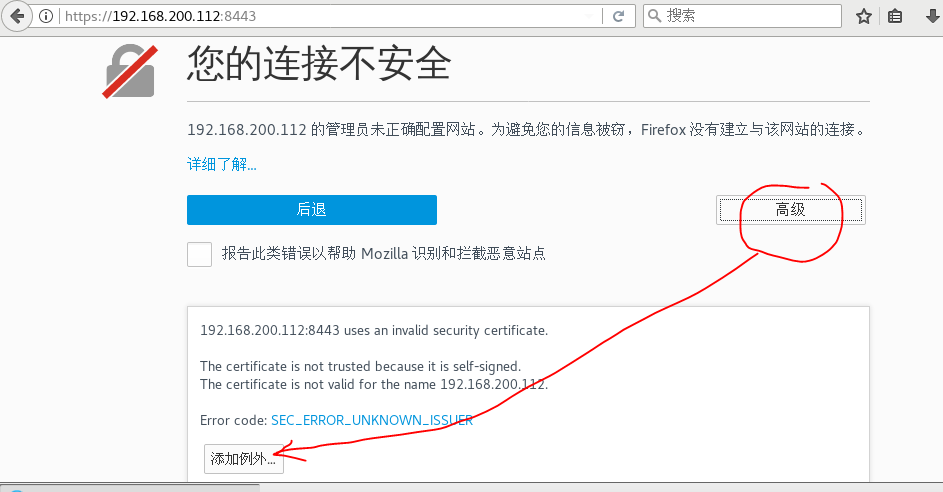
**[root@localhost conf]# /usr/local/tomcat8/bin/shutdown.sh**

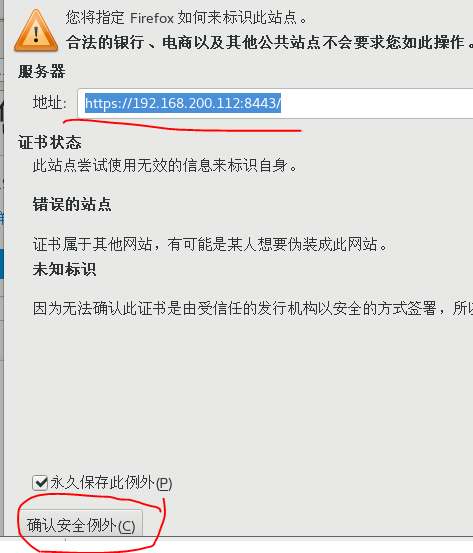
**[root@localhost conf]# /usr/local/tomcat8/bin/startup.sh**

密码和证书的位置根据个人的具体环境而设置，属性参数如下所述：

|  |  |
| --- | --- |
| 属性 | 描述 |
| clientAuth | 如果设为true，表示Tomcat要求所有的SSL客户出示安全证书，对SSL客户进行身份验证 |
| keystoreFile | 指定keystore文件的存放位置，可以指定绝对路径，也可以指定相对于<CATALINA\_HOME>（Tomcat安装目录）环境变量 的相对路径。如果此项没有设定，默认情况下，Tomcat将从当前操作系统用户的用户目录下读取名为“.keystore”的文件。 |
| keystorePass | 指定keystore的密码，如果此项没有设定，在默认情况下，Tomcat将使用“changeit”作为默认密码。 |
| sslProtocol | 指定套接字（Socket）使用的加密/解密协议，默认值为TLS，用户不应该修改这个默认值。 |
| ciphers | 指定套接字可用的用于加密的密码清单，多个密码间以逗号（,）分隔。如果此项没有设定，在默认情况下，套接字可以使用任意一个可用的密码。 |

浏览器访问：https://192.168.200.112:8443





即可成功访问

# Tomcat虚拟主机

虚拟主机用于在一台物理机上搭建多个web站点，每个web站点独立运行，互不干扰，这些站点就是"虚拟主机"。

## 基于域名的虚拟主机

多个域名解析到同一个IP地址，在WEB服务器里添加多个站点，每个站点绑定一个域名。HTTP协议请求里包含了域名信息，当WEB服务器收到访问请求时，就可以根据不同的域名来访问不同的网站。

**[root@localhost ~]# tar xf apache-tomcat-8.5.16.tar.gz**

**[root@localhost ~]# mv apache-tomcat-8.5.16 /usr/local/tomcat8**

配置域名与IP的映射管理

对于本地局域网我们使用在host文件中添加。对于大型网络或者外网网络则需要配置DNS服务器中Ip地址与域名的映射关系。

**[root@localhost ~]# tail -2 /etc/hosts**

**192.168.200.112 www.benet.com**

**192.168.200.112 www.accp.com**

修改server.xml

**[root@localhost conf]# cp server.xml server.xml\_$(date +%F)**

**[root@localhost conf]# vim server.xml**

**准备jsp网页文件**

**<Host name="www.benet.com" appBase="webapps"**

**unpackWARs="true" autoDeploy="true">**

**<Context docBase="/web/benet" path="" reloadable="flase" />**

**<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"**

**prefix="localhost\_access\_log" suffix=".txt"**

**pattern="%h %l %u %t &quot;%r&quot; %s %b" />**

**</Host>**

**<Host name="www.accp.com" appBase="webapps"**

**unpackWARs="true" autoDeploy="true">**

**<Context docBase="/web/accp" path="" reloadable="flase" />**

**<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"**

**prefix="localhost\_access\_log" suffix=".txt"**

**pattern="%h %l %u %t &quot;%r&quot; %s %b" />**

**</Host>**

**[root@localhost ~]# mkdir -pv /web/{benet,accp}**

**mkdir: 已创建目录 "/web/benet"**

**mkdir: 已创建目录 "/web/accp"**

**[root@localhost ~]# echo www.benet.com >/web/benet/index.jsp**

**[root@localhost ~]# echo www.accp.com >/web/accp/index.jsp**

**[root@localhost conf]# /usr/local/tomcat8/bin/startup.sh**

**Using CATALINA\_BASE: /usr/local/tomcat8**

**Using CATALINA\_HOME: /usr/local/tomcat8**

**Using CATALINA\_TMPDIR: /usr/local/tomcat8/temp**

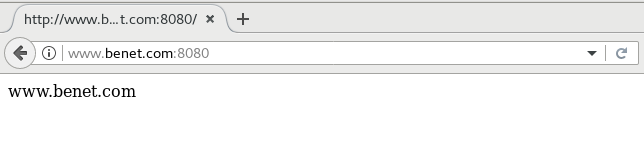
**Using JRE\_HOME: /usr**

**Using CLASSPATH: /usr/local/tomcat8/bin/bootstrap.jar:/usr/local/tomcat8/bin/tomcat-**

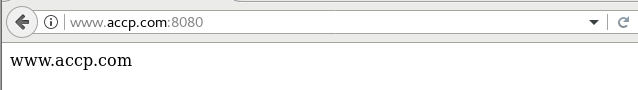
**juli.jarTomcat started.**

访问测试

[http://www.benet.com:8080](http://www.crushlinux.com:8080)



[http://www.accp.com:8080](http://www.cloud.com:8080)



## 基于端口的虚拟主机

主机只拥有一个IP地址，通过不同的端口实现不同WEB站点的访问。

在server.xml中设置两个service组件

**[root@localhost conf]# vim server.xml**

**<Service name="Catalina1">**

**<Connector port="8080" protocol="HTTP/1.1"**

**connectionTimeout="20000"**

**redirectPort="8443" />**

**<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />**

**<Engine name="Catalina" defaultHost="localhost">**

**<Realm className="org.apache.catalina.realm.LockOutRealm">**

**<Realm className="org.apache.catalina.realm.UserDatabaseRealm"**

**resourceName="UserDatabase"/>**

**</Realm>**

**<Host name="www.benet.com" appBase="webapps"**

**unpackWARs="true" autoDeploy="true">**

**<Context docBase="/web/benet" path="" reloadable="flase" />**

**<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"**

**prefix="localhost\_access\_log" suffix=".txt"**

**pattern="%h %l %u %t &quot;%r&quot; %s %b" />**

**</Host>**

**</Engine>**

**</Service>**

**<Service name="Catalina2">**

**<Connector port="8090" protocol="HTTP/1.1"**

**connectionTimeout="20000"**

**redirectPort="8443" />**

**<Connector port="8010" protocol="AJP/1.3" redirectPort="8443" />**

**<Engine name="Catalina" defaultHost="localhost">**

**<Realm className="org.apache.catalina.realm.LockOutRealm">**

**<Realm className="org.apache.catalina.realm.UserDatabaseRealm"**

**resourceName="UserDatabase"/>**

**</Realm>**

**<Host name="www.benet.com" appBase="webapps"**

**unpackWARs="true" autoDeploy="true">**

**<Context docBase="/web/accp" path="" reloadable="flase" />**

**<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"**

**prefix="localhost\_access\_log" suffix=".txt"**

**pattern="%h %l %u %t &quot;%r&quot; %s %b" />**

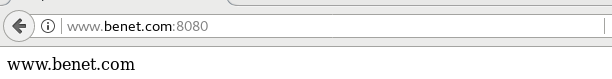
**</Host>**

**</Engine>**

**</Service>**

访问测试

[http://www.benet.com:8080](http://www.crushlinux.com:8080)



[http://www.benet.com:8090](http://www.crushlinux.com:8090)



这里由于我把8090端口的根目录指向了上个实验中域名[accp](http://www.accp.com)的目录，因此访问的内容就是www.accp.com

## 基于IP地址的虚拟主机

服务器使用多网卡配置多个IP地址，然后配置WEB服务器，把多个网站绑定在不同的IP上。（该方式浪费Ip资源，Tomcat不支持该方式）。

# Tomcat多实例配置

WEB应用程序需要部署在服务器上时 Tomcat 软件的部署方式可以分为以下几种：

* 单实例单应用（webapps/a）
* 单实例多应用（/web/{accp,benet}）
* 多实例单应用
* 多实例多应用

实例的概念可以先理解为一个 Tomcat 目录中的一个工作子目录

* 单实例单应用：比较常用的一种方式，只需要把做好的 war 包丢在 webapps目录下，执行启动 Tomcat 的脚本就行了。
* 单实例多应用：有两个不同的 Web 项目的 war 包，还是只需要丢在webapps目录下，执行启动 Tomcat 的脚本，访问不同项目会加载不同的WEB虚拟目录。这种方式在生产环境中要慎用，因为重启或挂掉 Tomcat 程序后会影响另外一个应用的访问。一个配置文件（实例），两个service（各有一个host），使用不同的connector端口对应不同的应用
* 多实例单应用：多个 Tomcat 实例部署同一个项目，端口号不同，可以利用 Nginx 做负载均衡。 多个配置文件，不同的端口号，各有一个host，但是应用相同
* 多实例多应用：多个 Tomcat 实例部署多个不同的项目。这种模式在服务器资源有限，或者对服务器资源要求并不是很高的情况下，可以实现多个不同项目部署在同一台服务器上的需求，来实现资源使用的最大化。

a、解压并部署tomcat程序

**[root@localhost~]# tar xf apache-tomcat-8.5.16.tar.gz**

**[root@localhost~]# cp -r apache-tomcat-8.5.16 /usr/local/tomcat1**

**[root@localhost~]# cp -r apache-tomcat-8.5.16 /usr/local/tomcat2**

b、修改每个tomcat实例中server.xml中的端口（分别修改以上三个端口（Server port、Connector port、AJP），不要和其它实例的端口或系统已经占用的端口发生冲突）。

**[root@localhost~]# cat /usr/local/tomcat2/conf/server.xml**

**<?xml version="1.0" encoding="UTF-8"?>**

**<Server port="8006" shutdown="SHUTDOWN">**

**<Listener className="org.apache.catalina.startup.VersionLoggerListener" />**

**<Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />**

**<Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />**

**<Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />**

**<Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />**

**<GlobalNamingResources>**

**<Resource name="UserDatabase" auth="Container"**

**type="org.apache.catalina.UserDatabase"**

**description="User database that can be updated and saved"**

**factory="org.apache.catalina.users.MemoryUserDatabaseFactory"**

**pathname="conf/tomcat-users.xml" />**

**</GlobalNamingResources>**

**<Service name="Catalina">**

**<Connector port="8090" protocol="HTTP/1.1"**

**connectionTimeout="20000"**

**redirectPort="8443" />**

**<Connector port="8010" protocol="AJP/1.3" redirectPort="8443" />**

**<Engine name="Catalina" defaultHost="localhost">**

**<Realm className="org.apache.catalina.realm.LockOutRealm">**

**<Realm className="org.apache.catalina.realm.UserDatabaseRealm"**

**resourceName="UserDatabase"/>**

**</Realm>**

**<Host name="localhost" appBase="webapps"**

**unpackWARs="true" autoDeploy="true">**

**<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"**

**prefix="localhost\_access\_log" suffix=".txt"**

**pattern="%h %l %u %t &quot;%r&quot; %s %b" />**

**</Host>**

**</Engine>**

**</Service>**

**</Server>**

注意server.xml中Host节点的appBase属性的值指向。

c、创建测试页

**[root@localhost~]# vim /usr/local/tomcat1/webapps/ROOT/index.jsp**

**hello tomcat1:8080**

**[root@localhost~]# vim /usr/local/tomcat2/webapps/ROOT/index.jsp**

**hello tomcat2:8090**

d、启动tomcat实例

**[root@localhost~]# /usr/local/tomcat1/bin/startup.sh**

**[root@localhost~]# /usr/local/tomcat2/bin/startup.sh**

查看监听端口

**[root@localhost~]# netstat -lnpt | grep java**

**tcp6 0 0 127.0.0.1:8005 :::\* LISTEN 65208/java**

**tcp6 0 0 127.0.0.1:8006 :::\* LISTEN 65246/java**

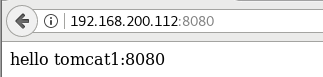
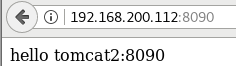
**tcp6 0 0 :::8009 :::\* LISTEN 65208/java**

**tcp6 0 0 :::8010 :::\* LISTEN 65246/java**

**tcp6 0 0 :::8080 :::\* LISTEN 65208/java**

**tcp6 0 0 :::8090 :::\* LISTEN 65246/java**

访问测试实例内容

# Nginx基于Tomcat多实例的负载均衡

配置nginx.conf

**[root@nginx nginx-1.14.2]# cp /usr/local/nginx/conf/nginx.conf{,.bak}**

**[root@nginx nginx-1.14.2]# vim /usr/local/nginx/conf/nginx.conf**

**user nginx nginx;**

**worker\_processes 2;**

**error\_log logs/error.log;**

**pid logs/nginx.pid;**

**events {**

**use epoll;**

**worker\_connections 10240;**

**}**

**http {**

**include mime.types;**

**default\_type application/octet-stream;**

**log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '**

**'$status $body\_bytes\_sent "$http\_referer" '**

**'"$http\_user\_agent" "$http\_x\_forwarded\_for"';**

**access\_log logs/access.log main;**

**sendfile on;**

**keepalive\_timeout 65;**

**upstream tomcat\_server {**

**server 192.168.200.112:8080 weight=1;**

**server 192.168.200.112:8090 weight=1;**

**}**

**server {**

**listen 80;**

**server\_name localhost;**

**location / {**

**root html;**

**index index.html index.htm;**

**}**

**location ~ \.jsp$ {**

**proxy\_pass** [**http://tomcat\_server**](http://tomcat_server)**;**

**}**

**error\_page 500 502 503 504 /50x.html;**

**location = /50x.html {**

**root html;**

**}**

**}**

**}**

**[root@nginx nginx-1.14.2]# /usr/local/nginx/sbin/nginx -t**

**nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok**

**nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful**

**[root@nginx nginx-1.14.2]# /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf**

**[root@nginx nginx-1.14.2]# netstat -anpt |grep :80**

**tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 4632/nginx: master**

**[root@nginx nginx-1.14.2]# ps aux | grep nginx | grep -v grep**

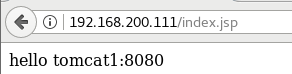
**root 4632 0.0 0.1 45936 1148 ? Ss 23:41 0:00 nginx: master process /usr/local/nginx/sbin/nginx -c /u**

**sr/local/nginx/conf/nginx.confnginx 4633 0.0 0.5 52276 5708 ? S 23:41 0:00 nginx: worker process**

**nginx 4634 0.0 0.5 52276 5708 ? S 23:41 0:00 nginx: worker process**

客户端测试：

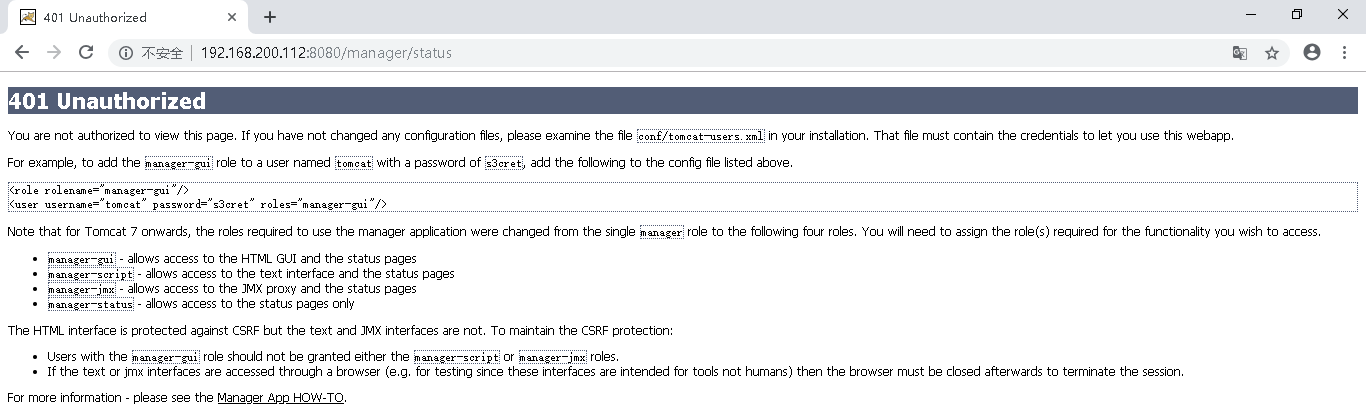
打开浏览器访问: <http://192.168.200.111>/index.jsp、不断刷新可看到由于权重相同，页面会反复切换



# Tomcat管理功能使用

tomcat的管理界面host-manager可以用于管理虚拟主机（新增、删除、单独重启一台虚拟主机）  
 manger可以用于Java项目部署  
 server status可以查看服务器状态

Tomcat管理功能用于对Tomcat自身以及部署在Tomcat上的应用进行管理的web应用。在默认情况下是处于禁用状态的。如果需要开启这个功能，就需要配置管理用户，即配置tomcat-users.xml 文件。在112搭建tomcat并配置，并且使用默认站点内容（不要删除默认主页）



**[root@localhost ~]# vim /usr/local/tomcat8/conf/tomcat-users.xml**

**<role rolename="manager-gui"/>**

**<role rolename="admin-gui"/>**

**<user username="tomcat" password="tomcat" roles="manager-gui,admin-gui"/>**

**</tomcat-users> # 在此行前加入上面三行添加用户角色设置密码**

**[root@localhost ~]# vim /usr/local/tomcat8/webapps/manager/META-INF/context.xml**

**<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"**

**allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->**

**[root@localhost ~]# vim /usr/local/tomcat/webapps/host-manager/META-INF/context.xml**

**<Context antiResourceLocking="false" privileged="true" >**

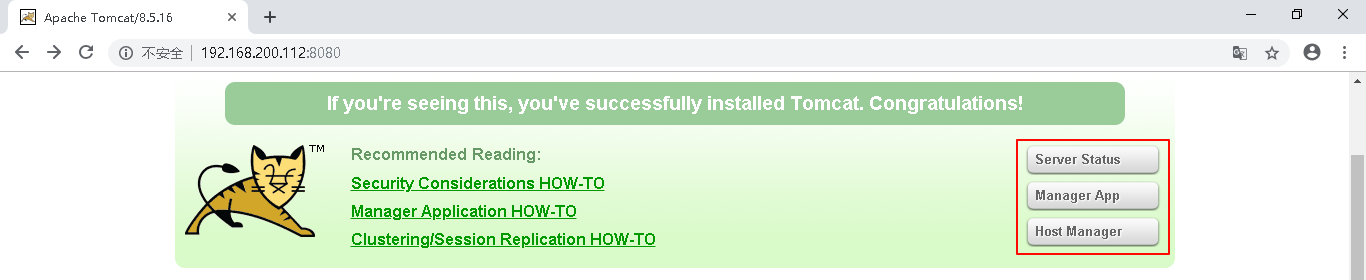
**<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"**

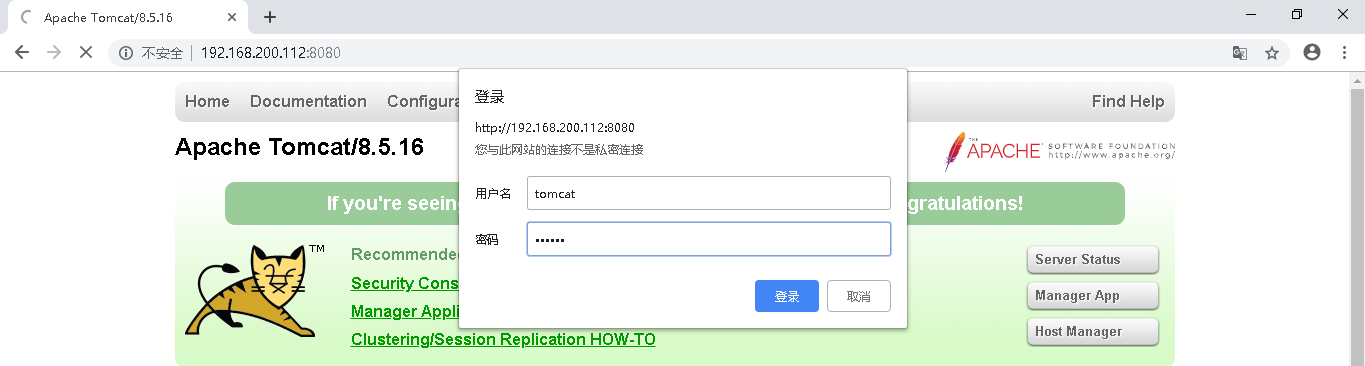
**allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->**

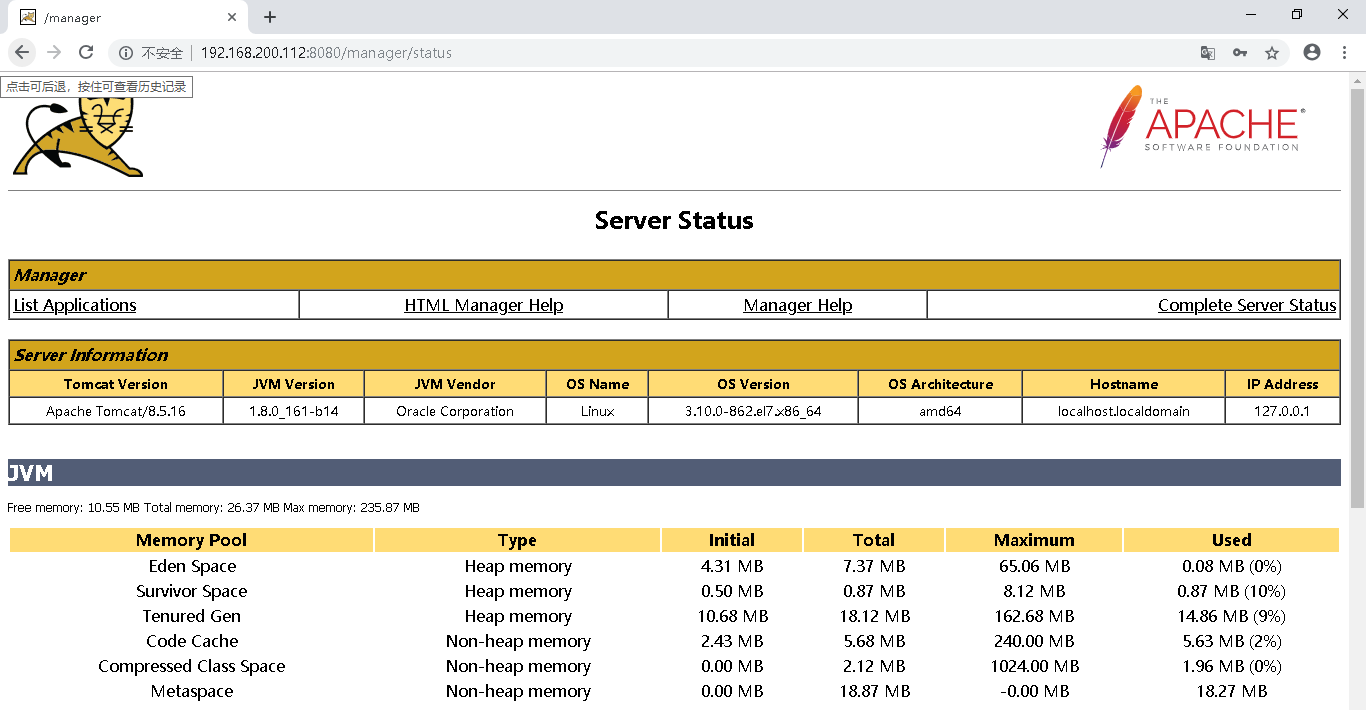
**将只允许本地访问manager和host-manager的配置注释掉**

**[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh**

**[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh**









# WEB站点部署

上线的代码有两种方式：

第一种方式是直接将程序目录放在webapps目录下面，这种方式大家已经明白了，就不多说了。

第二种方式是使用开发工具将程序打包成war包，然后上传到webapps目录下面。

使用war包部署web站点

部署tomcat内存检测包

上传meminfo.war包到/usr/local/tomcat8/webapps目录中

**[root@localhost webapps]# ls**

**docs examples host-manager manager meminfo.war ROOT**

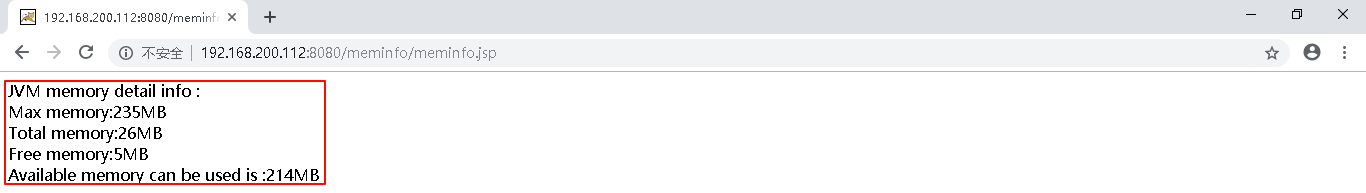
站点主动解压部署

**[root@localhost webapps]# ls**

**docs examples host-manager manager meminfo meminfo.war ROOT**

浏览器访问：

http://192.168.200.112:8080/meminfo/meminfo.jsp



**案例：部署开源站点（jpress）**

jpress官网：http://jpress.io

下载地址：<https://github.com/JpressProjects/jpress>（已下载好，无需自己下载）

Java使用默认版本

[root@server01 ~]# java -version

openjdk version "1.8.0\_161"

OpenJDK Runtime Environment (build 1.8.0\_161-b14)

OpenJDK 64-Bit Server VM (build 25.161-b14, mixed mode)

安装配置数据库

**[root@localhost ~]# yum -y install mariadb-server mariadb**

**[root@localhost ~]# systemctl start mariadb**

配置数据库

**[root@localhost ~]# mysql**

**Welcome to the MariaDB monitor. Commands end with ; or \g.**

**Your MariaDB connection id is 2**

**Server version: 5.5.56-MariaDB MariaDB Server**

**Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.**

**Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.**

**MariaDB [(none)]> create database jpress DEFAULT CHARACTER SET utf8;**

**Query OK, 1 row affected (0.00 sec)**

**MariaDB [(none)]> grant all on jpress.\* to jpress@'localhost' identified by '123456';**

**Query OK, 0 rows affected (0.00 sec)**

**MariaDB [(none)]> flush privileges;**

**Query OK, 0 rows affected (0.00 sec)**

**MariaDB [(none)]> exit**

**Bye**

jpress站点上线：将jpress-v3.06.war包上传到tomcat服务器

**[root@localhost ~]# mv jpress-v3.0.6.war /usr/local/tomcat8/webapps/**

**[root@localhost ~]# ls /usr/local/tomcat8/webapps/**

**docs examples host-manager jpress-v3.0.6 jpress-v3.0.6.war manager ROOT**

浏览器访问：http://192.168.200.112:8080/jpress-v3.0.6/install

