## Tomcat

动态站点:

运行程序

客户端

服务端

一处编译,到处运行

java技术的组成部分:

java程序设计语言

java API

java vm

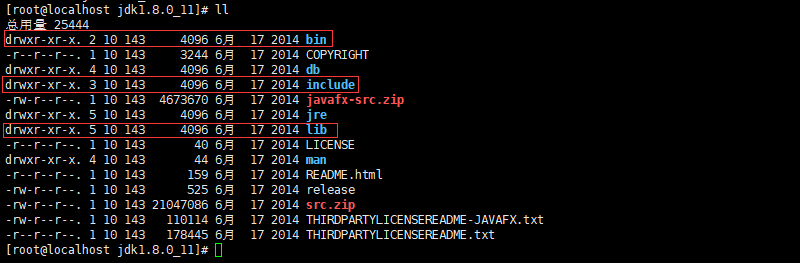
## 下载jdk

wget --no-cookies --no-check-certificate --header "Cookie: gpw\_e24=http%3A%2F%2Fwww.oracle.com%2F;oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.tar.gz"

从官网下载太慢了.

所以我搜了个百度网盘<http://blog.csdn.net/doupeihua/article/details/51831947>

安装解压后我们会看到如下目录:



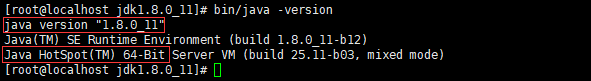
bin:二进制文件

include:头文件

lib:库文件

## 查看java版本:

bin/java -version



最后面的有mixed mode

其中jdk有客户端模式和服务端模式,混合模式表示这两种模式都支持!

## 配置java\_home环境变量

输入命令:

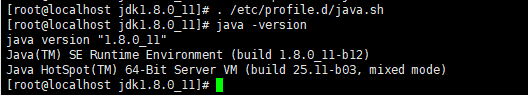
vi /etc/profile.d/java.sh

插入如下

export JAVA\_HOME=/usr/local/jdk1.8.0\_11  
export PATH=$JAVA\_HOME/bin:$PATH

保存后测试:

. /etc/profile.d/java.sh



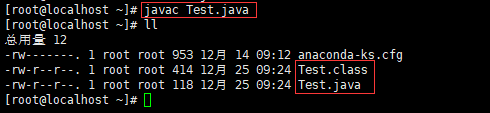
运行无误后,说明配置没问题!

## 编译运行

在/root目录下创建Test.java

**public class Test** {  
 **public static void** main(**String**[] args) {  
 **System**.***out***.println("hello,world!");  
 }  
}

用javac进行编译



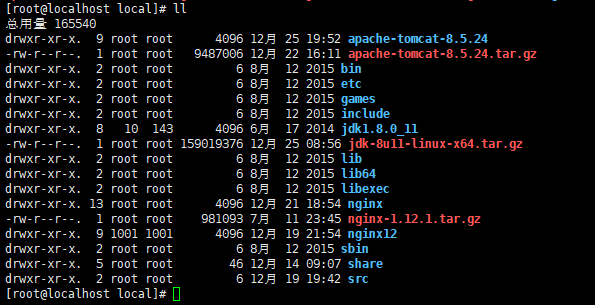
运行



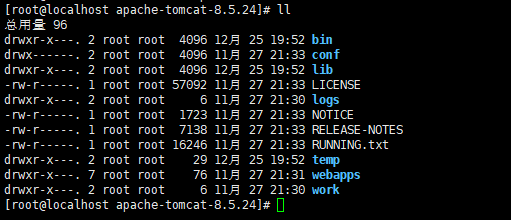
查看java运行程序



## 解压tomcat



目录



bin:shell脚本,bat脚本

conf:xml 格式配置文件(server.xml重要)

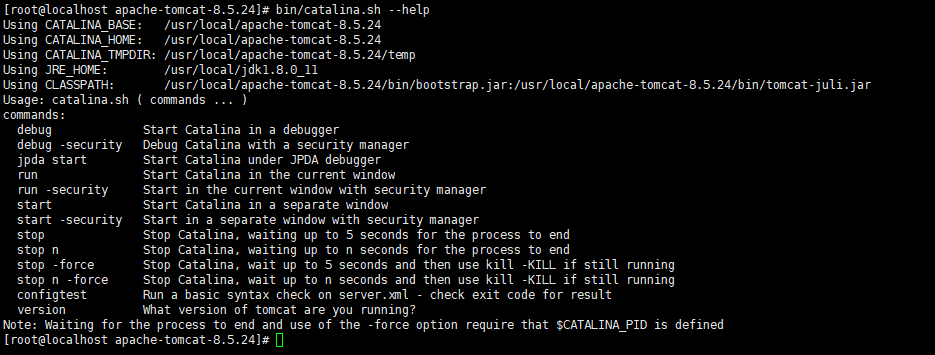
lib:jar文件

logs:日志文件

webapps:应用程序文件

work:tomcat工作目录

## tomcat命令



## 配置tomcat环境变量

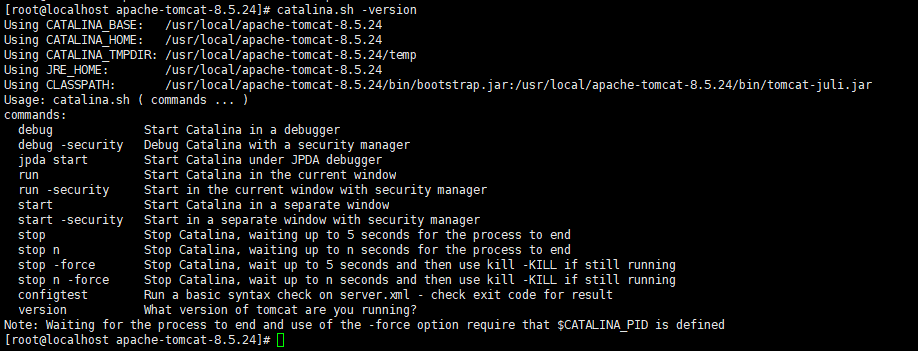
vi /etc/profile.d/tomcat.sh

export CATALINA\_HOME=/usr/local/apache-tomcat-8.5.24  
export PATH=$CATALINA\_HOME/bin:$PATH

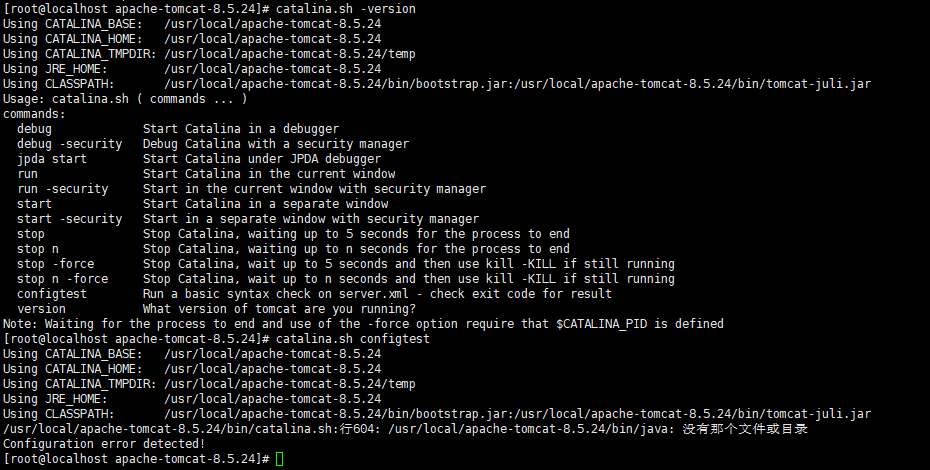
执行source命令

. /etc/profile.d/tomcat.sh

执行

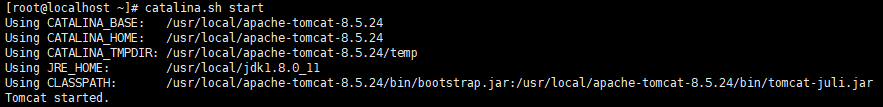


检测配置是否报错

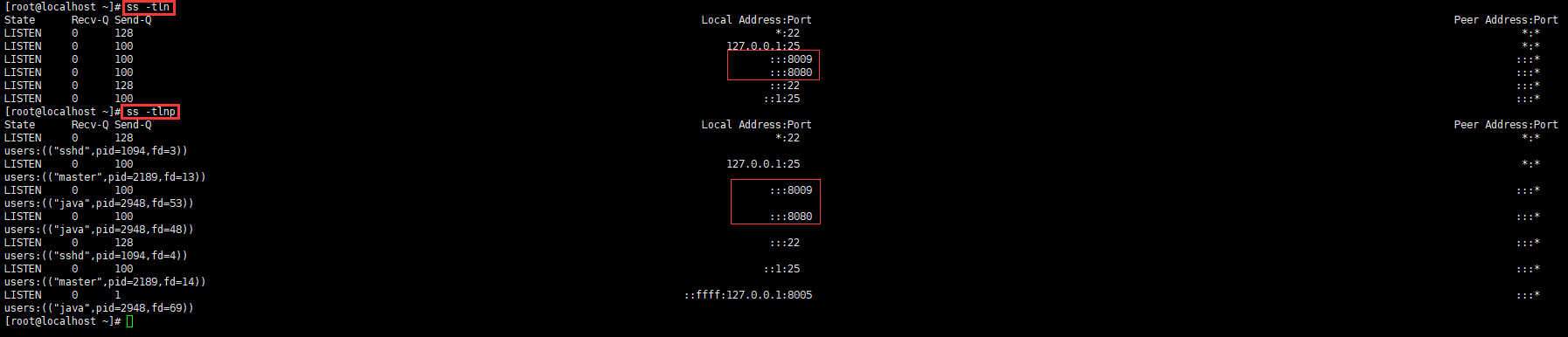


## 启动服务:

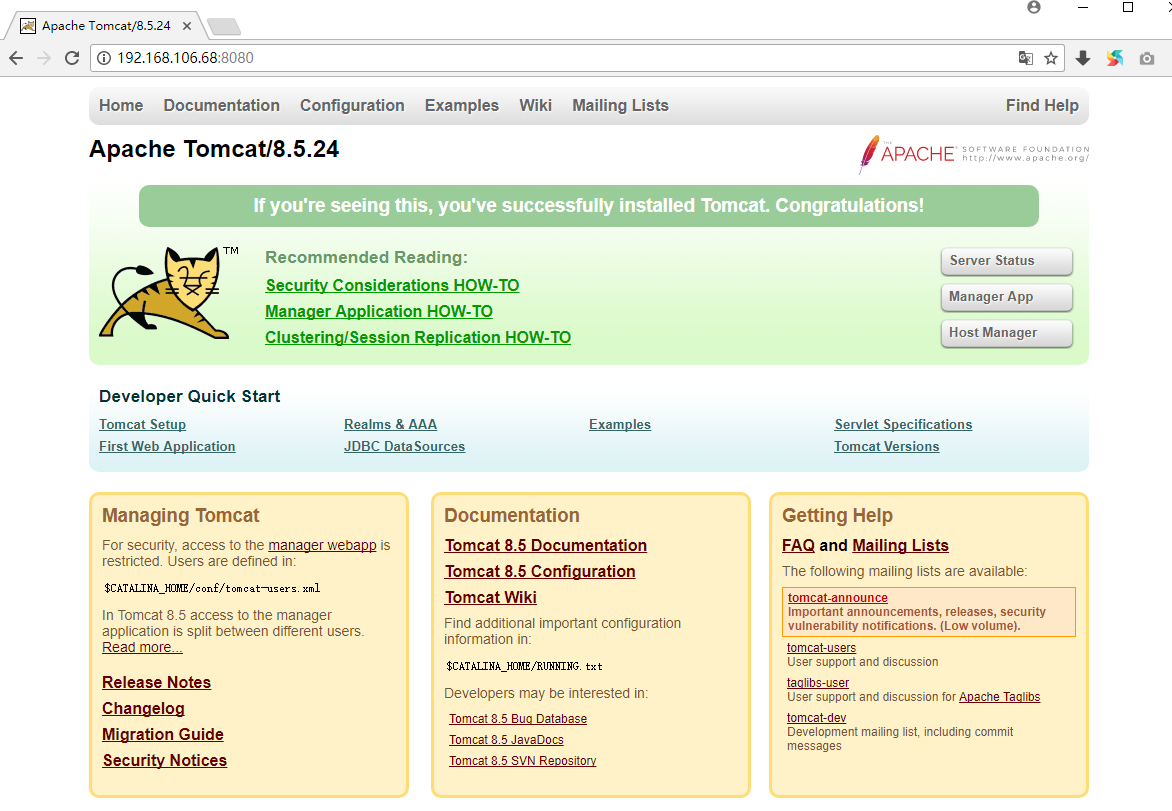
catalina.sh start



查看监听的服务



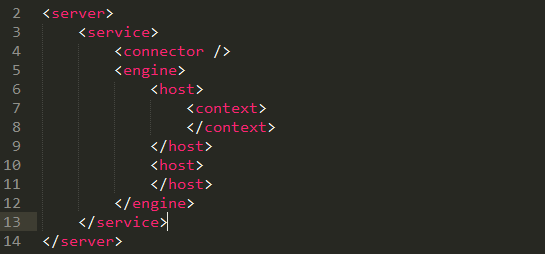
登陆服务:



## 关闭服务:

catalina.sh stop

## server.xml配置文件的层次



web container

Jetty,Resin

WebSphere,weblogic,JBoss,Glassfish

Tomcat:

server.xml

顶级类:server

容器类:engine,host,context

服务器:service

连接器:connector

http,ssl,ajp(apach jserv protocol)

被嵌套类:valve,logger,realm

顶级组件:位于整个配置的最外层;

容器类组件:可以包含其他组件的组件;

连接器组件:连接用户请求至tomcat;

被嵌套类的组件:位于一个容器当中,不能包含其他组件;

容器类:

engine:核心容器,catalina的核心引擎,负责通过connecor接收用户请求;

host:类似于httpd中的虚拟主机;支持基于FQDN的虚拟主机

context:最内层的容器类组件,一个context代表一个web应用程序;配置contxext的主要目的,指定对应的webapp的根目录;还可以为webapp指定额外的属性,如部署方式等;

服务:

service:将连接器关联至engine;

因此一个service内部可以有多个connector,但只能有一个engine;

顶级组件:server,表示一个运行于JVM中的tomcat实例;

嵌套类组件:

valve:拦截请求并在将其转至对应的webapp之前进行某种处理操作

access log valve:

romote address filter valve:基于ip做访问控制

logger:日志记录器,用于记录组件 内部的状态信息;

可以用于除context之外的任何容器中

realm:可以用于任何容器类的组件中,关联一个用户认证库,实现认证和授权;

UserDatabaseRealm: 使用JNDI自定义的用户认证库;

MemoryRealm: tomcat-users.xml中

JDBCRealm: 基于JDBC连接至数据库中查找用户;

部署:使用类加载器,为webapp准备好其依赖所有类;

apache jserv protocol

二进制协议,使用httpd反向代理用户请求至tomcat时,在httpd和tomcat之间使用.

## 书写startup脚本:

#!/bin/sh  
# Tomcat init script for Linux.  
#   
# chkconfig: 2345 96 14  
# description: The Apache Tomcat servlet/JSP containter.  
# JAVA\_OPTS='-Xms64m -Xmx128m'  
JAVA\_HOME=/usr/local/latest  
CATALINA\_HOME=/user/local/Tomcat  
export JAVA\_HOME CATALINA\_HOME  
  
case $1 in  
start)  
 exec $CATALINA\_HOME/bin/catalina.sh start ;;  
stop)  
 exec $CATALINA\_HOME/bin/catalina.sh stop ;;  
restart)  
 $CATALINA\_HOME/bin/catalina.sh stop  
 sleep 2  
 exec $CATALINA\_HOME/bin/catalina.sh start ;;  
\*)  
 echo "Usage:`basename $0` {start|stop|restart}"  
 exit 1  
 ;;  
esac