operation day 05

================================================

**使用tomcat搭建网站服务**

**由于tomcat的运行需要依赖java，要先安装jdk工具包**

**1，配置java环境并拷贝tomcat程序到指定目录(该目录位置可以自定义，本次实验依然在/usr/local目录下)**

[root@web1 ~]# yum -y install java-1.8.0-openjdk

[root@web1 ~]# cd lnmp\_soft/

[root@web1 lnmp\_soft]# tar -xf apache-tomcat-8.0.30.tar.gz

cp -r apache-tomcat-8.0.30 /usr/local/tomcat

cd /usr/local/tomcat

ls

tomcat目录介绍：

conf 存放配置文件

logs 存放日志

bin 存放主程序

webapps 存放网站页面

work 存放了编译的网站页面

temp 临时文件目录

lib 库文件目录

cd /usr/local/tomcat

bin/startup.sh //开启服务

bin/shutdown.sh //关闭服务

<http://192.168.2.100:8080/> 使用火狐打开tomcat网页，tomcat的网页没有

使用默认的80端口而是8080

**2，无法正常开启tomcat时解决方案**

**由于tomcat运行需要random中有海量随机数，有时随机数不足会导致8005端口**

**无法启动**

**解决方案一：**

mv /dev/random /dev/random.bak //备份原有random文件

ln -s /dev/urandom /dev/random //使用urandom创建软连接替代random即可

[root@web1 tomcat]# yum -y install psmisc //安装支持killall命令软件包

[root@web1 tomcat]# killall java //如果tomcat异常可以杀掉

[root@web1 tomcat]# bin/startup.sh //重新开启

[root@web1 tomcat]# ss -ntulp | grep java //再次查询可以看到8005端口

说明tomcat运行正常

**解决方案二：**

yum -y install rng-tools //如果8005端口依然无法开启就安装该软件包

systemctl start rngd //开启服务

http://192.168.2.100:8080/

**3，测试tomcat服务**

编写测试静态页面

[root@web1 tomcat]# echo "tomcat-test~~~~" > webapps/ROOT/test.html

<http://192.168.2.100:8080/test.html>

编写测试动态页面

[root@web1 tomcat]#vim webapps/ROOT/test.jsp

<html>

<body>

<center>

Now time is: <%=new java.util.Date()%>

</center>

</body>

</html>

http://192.168.2.100:8080/test.jsp //使用浏览器访问可以看到当前时间

------------------------------------------------------------------

**利用tomcat搭建虚拟主机**

**回顾**

httpd的创建方式是写多个virtualhost

<VirtualHost \*:80>

servername [www.a.com](http://www.a.com)

documentroot /var/www/html

</virtualhost>

nginx的创建方式是写多个server

http {

server {

listen 80;

server\_name [www.a.com](http://www.a.com);

root html;

index index.html;

}

}

tomcat中创建虚拟主机是写多个Host

<Host name=www.a.com appBase=webapps ….. >

……

……

</Host>

-----------------------------------------------------------------------------

**在tomcat中创建虚拟主机实际操作**

[root@web1 tomcat]# vim conf/server.xml //修改配置文件，添加虚拟主机

<Host name="www.b.com" appBase="web\_b"> //122行添加b网站

</Host> //123行是结尾

<Host name="www.a.com" appBase="webapps" //124行是默认的

虚拟主机，将名字改成a网站

[root@web1 tomcat]# mkdir -p web\_b/ROOT //在/usr/local/tomcat目录下创建b网站的测试目录

[root@web1 tomcat]# echo "tomcat-A~~" > webapps/ROOT/index.html //创建a网站测试页面

[root@web1 tomcat]# echo "tomcat-B~~" > web\_b/ROOT/index.html //创建b网站测试页面

vim /etc/hosts //之后修改hosts文件添加

192.168.2.100 [www.a.com](http://www.a.com) [www.b.com](http://www.b.com) [www.c.com](http://www.c.com)

bin/shutdown.sh //关闭服务

bin/startup.sh //再开启服务

[root@web1 tomcat]# curl [www.a.com:8080](http://www.a.com:8080) //访问测试

tomcat-A~~

[root@web1 tomcat]# curl www.b.com:8080

tomcat-B~~

C:\Windows\System32\drivers\etc\hosts //如果真实主机的火狐要测试

需要修改Windows的hosts文件

192.168.2.100 [www.a.com](http://www.a.com) [www.b.com](http://www.b.com) [www.c.com](http://www.c.com)

**另外，虚拟主机配置文件中的autoDeploy和unpackWARs功能解释如下：**

**autoDeploy //自动更新网站功能，有助于开发工程师**

**unpackWARs //自动解war包，通常开发工程师习惯将网站文件打成war包(类似tar包)上传到服务器，但是还需要解包很麻烦，但有了这个功能就可以自动解开**

**测试自动解war包功能：**

[root@web1 /]# yum -y install java-1.8.0-openjdk-devel //安装创建war包的工具包

cd /

[root@web1 /]# jar -cf xyz.war /var/log 创建war包，名字是xyz.war，内

容是/var/log

cp xyz.war /usr/local/tomcat/web\_b //把war包扔到虚拟主机的网页目录下

ls /usr/local/tomcat/web\_b //查看b网站目录里面会自动解开xyz.war包，并产

生xyz目录，前提是tomcat服务正常开启中

-----------------------------------------------------------------------

**tomcat页面路径测试**

**默认情况下，tomcat网站页面的文件存放在webapps目录的ROOT下，如果需要自定义路径可以按以下几种方式进行配置，以下测试在b网站进行，b网站目录是web\_b**

**测试一:打开配置文件，找到b网站的配置**

<Host name="www.b.com" appBase="web\_b"

unpackWARs="true" autoDeploy="true">

<Context path="" docBase="" /> //增加本行内容

</Host>

[root@web1 tomcat]# echo "web\_b/index.html" > web\_b/index.html

重启tomcat服务

<http://www.b.com:8080> //测试，看到的是web\_b/index.html中的页面，而不是web\_b/ROOT/index.html的

**测试二：**

<Host name="www.b.com" appBase="web\_b"

unpackWARs="true" autoDeploy="true">

<Context path="" docBase="abc" /> //这里改成docBase="abc"

</Host>

[root@web1 tomcat]# mkdir web\_b/abc //创建测试目录

[root@web1 tomcat]# echo "web\_b/abc/index.html" > web\_b/abc/index.html

重启tomcat服务

ss -ntulp | grep java

[root@web1 tomcat]# curl [www.b.com:8080](http://www.b.com:8080)

访问测试，看到的是web\_b/abc/index.html的内容

**测试三：**

<Host name="www.b.com" appBase="web\_b"

unpackWARs="true" autoDeploy="true">

<Context path="" docBase="/abc" /> //将abc修改为/abc

</Host>

[root@web1 tomcat]# mkdir /abc //创建测试目录

[root@web1 tomcat]# echo "/abc/index.html" > /abc/index.html

重启tomcat服务

[root@web1 tomcat]# curl www.b.com:8080

访问测试，看到的是/abc/index.html中的内容

**测试四：**

<Host name="www.b.com" appBase="web\_b"

unpackWARs="true" autoDeploy="true">

<Context path="/test" docBase="/abc" />

</Host>

重启tomcat服务

访问[www.b.com:8080/test/](http://www.b.com:8080/test/) 显示的是/abc/index.html的内容

访问[www.b.com:8080/](http://www.b.com:8080/test/) 显示的是web\_b/ROOT/index.html的内容

测试五：

<Host name="www.b.com" appBase="web\_b"

unpackWARs="true" autoDeploy="true">

<Context path="/test" docBase="abc" />

</Host>

重启tomcat服务

访问[www.b.com:8080/test/](http://www.b.com:8080/test/) 网页内容是web\_b/abc/index.html

访问[www.b.com:8080/](http://www.b.com:8080/test/) 网页内容是web\_b/ROOT/index.html

**练习：下列需求如何实现？**

访问[www.b.com:8080](http://www.b.com:8080) 看到的是/usr/local/tomcat/test01/a/index.html中的内容

访问[www.b.com:8080/abc/](http://www.b.com:8080/abc/) 看到的是/var/www/html/index.html中的内容

答案:

<Host name="www.b.com" appBase="test01"

unpackWARs="true" autoDeploy="true">

<Context path="" docBase="a" />

<Context path="/abc" docBase="/var/www/html" />

</Host>

--------------------------------------------------------

**使用tomcat搭建https网站**

修改配置文件，打开找到85行，将前后的注释删除 <!-- -->

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

maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

clientAuth="false" sslProtocol="TLS"

keystoreFile="/usr/local/tomcat/keystore" keystorePass="123456" /> //这里

添加keystoreFile(指定公钥私钥文件)和keystorePass(公钥私钥文件密码)

[root@web1 tomcat]# keytool -genkeypair -alias tomcat -keyalg

RSA -keystore /usr/local/tomcat/keystore //创建密钥对文件，-genkeypair是

创建密钥对，-alias是定义别名，-keyalg是指定算法，-keystore是指定

密钥对文件存放位置命令输入完毕后，先输入两次123456的密

码，然后回答问题，最后y确认，然后密码不用输入直接回车。

重启tomcat服务

curl -k <https://www.b.com:8443> //测试只要看到页面内容则成功