一、Tomcat配置SSL加密

二、Tomcat虚拟主机

1.基于域名的虚拟主机

2.基于端口的虚拟主机

3.基于IP地址的虚拟主机

三、Tomcat多实例配置

在企业实际生产环境中：

四、Nginx基于Tomcat多实例的负载均衡

安装nginx

配置nginx.conf

五、Tomcat管理功能使用

六、WEB站点部署(jpress)

1.部署tomcat内存检测包（meminfo.isp ）

2.部署开源站点（jpress）

安装配置数据库

七、监控Tomcat状态

方法一：开发Java监控页面 ----出力点在开发人员那边

方法二：使用jps命令进行监控 （不建议使用）

方法三：Tomcat远程监控功能--jconsole

特点：

使用普及

后续可以集成到zabbix

八、Tomcat故障排查步骤

九、Tomcat安全优化

1.telnet 管理端口保护（保护）

**前期环境搭建：**

```
[root@localhost ~]# iptables -F
[root@localhost ~]# systemctl stop firewalld
[root@localhost ~]# setenforce 0
setenforce: SELinux is disabled
用安装包安装jdk
[root@localhost ~]# rm -rf /usr/bin/java
[root@localhost ~]# mv jdk1.8.0_191/ /usr/local/java
[root@localhost ~]# vim /etc/profile.d/java.sh
#直接 写入 /etc/profile  ，或者另起脚本写，作用相同       为什么呢？
   当变量特别多，单独占一个文件，就是为了管理方便
JAVA_HOME=/usr/local/java
PATH=$PATH:$JAVA_HOME/bin
[root@localhost ~]# source /etc/profile
```

或者：[root@localhost ~]# source /etc/profile.d/java.sh

[root@localhost ~]# java -version

java version "1.8.0_191"

Java(TM) SE Runtime Environment (build 1.8.0_191-b12)

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

安装Tomcat

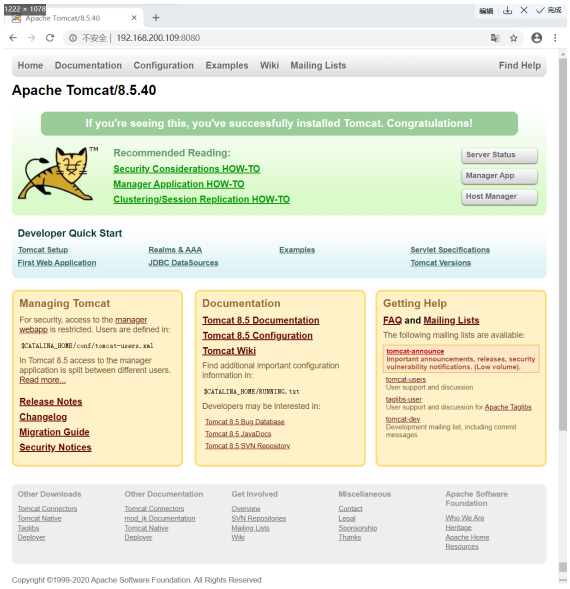
[root@localhost ~]# tar xf apache-tomcat-8.5.40.tar.gz

[root@localhost ~]# mv apache-tomcat-8.5.40 /usr/local/tomcat8

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

[root@localhost ~]# tail -f /usr/local/tomcat8/logs/catalina.out

[root@localhost ~]# netstat -lnpt




# 一、Tomcat配置SSL加密

创建加密用的私钥和证书文件　　---在企业中需要购买且有期限

这种方式不常见；

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

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

- keytool -genkeypair ： 产生密钥
- -alias tomcat ： 起别名叫Tomcat
- RSA： 算法
- -keystore /usr/local/tomcat8/keystore ： 证书存放路径

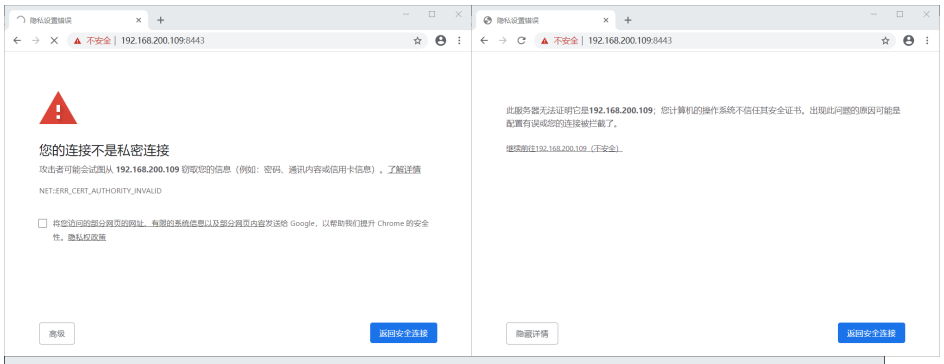
输入密钥库口令:　　# 123456

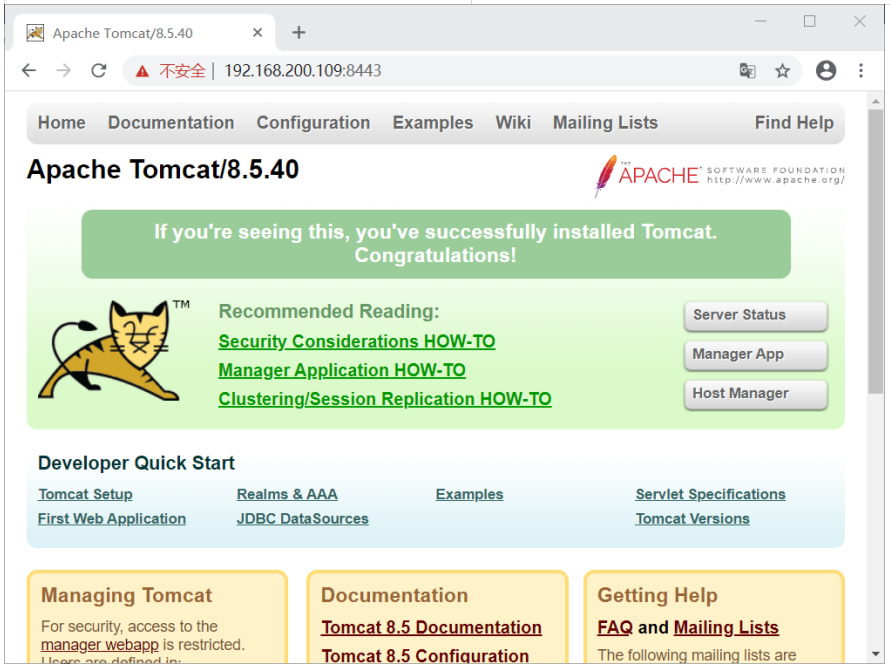
再次输入新口令:　　# 123456

```
您的名字与姓氏是什么?
  [Unknown]:  kitty
您的组织单位名称是什么?
  [Unknown]:  wind
您的组织名称是什么?
  [Unknown]:  wind
您所在的城市或区域名称是什么?
  [Unknown]:  shanghai
您所在的省/市/自治区名称是什么?
  [Unknown]:  huangpuqu
该单位的双字母国家/地区代码是什么?
  [Unknown]:  CN
CN=kitty, OU=wind, O=wind, L=shanghai, ST=huangpuqu, 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 -des
tkeystore /usr/local/tomcat8/keystore -deststoretype pkcs12" 迁移到行业标准格式
PKCS12。
[root@localhost ~]# ls -l /usr/local/tomcat8/keystore
-rw-r--r-- 1 root root 2225 4月   7 21:07 /usr/local/tomcat8/keystore

接下来使Tomcat支持证书
Tomcat运行时的端口：8005关闭端口  8080 服务端口  8009 Apache调度
[root@localhost ~]# cd /usr/local/tomcat8/conf/
[root@localhost conf]# vim server.xml           # 修改主配置文件
88     <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
89                maxThreads="150" SSLEnabled="true" scheme="https" secure="ture"
90                clientAuth="false" sslProtocol="TLS"
keystoreFile="/usr/local/tomcat8/keystore" keystorePass="123456">
91     </Connector>
[root@localhost conf]# /usr/local/tomcat8/bin/shutdown.sh
[root@localhost conf]# /usr/local/tomcat8/bin/startup.sh
```

访问时注意：

- **https://192.168.200.109:8443**
- 使用谷歌浏览器




# 二、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"

绝大多数网站软件都支持虚拟主机

网站软件：apache,nginx,tomcat,sun,IIS(internet信息服务，windows)，Tengine(淘宝nginx)
lighttpd

虚拟主机的类型：域名 IP 端口

## 1.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，根据不同的域名来访问不同的网站

需要全新的环境

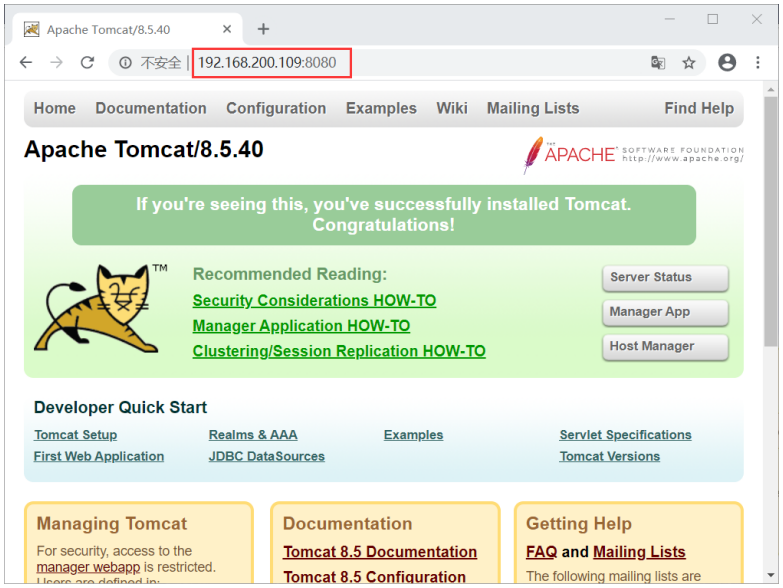
[root@localhost conf]# vim server.xml

将修改过的加注释

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

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

正常访问：




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

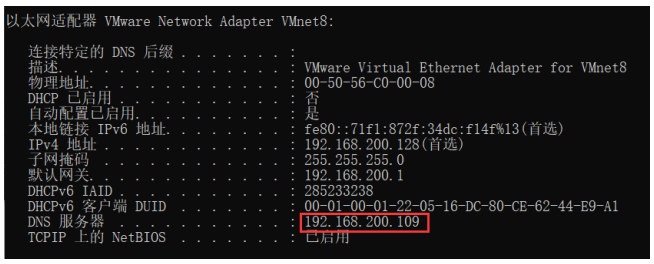
对于大型网络或者外网网络，则需要配置DNS服务器中lp地址与域名的映射关系

```
[root@localhost ~]# rpm -q bind bind-utils bind-libs
未安装软件包 bind
bind-utils-9.9.4-61.el7.x86_64
bind-libs-9.9.4-61.el7.x86_64
[root@localhost ~]# yum -y install bind
[root@localhost ~]# vim /etc/named.conf
    options {
            directory       "/var/named";
    };

    zone "kitty.com" IN {
            type master;
            file "kitty.zheng";
```

```
    };

    zone "pop.com" IN {
            type master;
            file "pop.zheng";
    };
[root@localhost ~]# cd /var/named
[root@localhost named]# vim kitty.zheng
$TTL  86400
@         SOA       kitty.com.    admin.kitty.com.         (
                    202040701
                    3H
                    15M
                    1W
                    1D
)
          IN        NS        ns.kitty.com.
ns        IN        A         192.168.200.109
www       IN        A         192.168.200.109
[root@localhost named]# cp kitty.zheng pop.zheng
[root@localhost named]# vim pop.zheng
$TTL  86400
@         SOA       pop.com.      admin.pop.com.   (
                    202040701
                    3H
                    15M
                    1W
                    1D
)
          IN        NS        ns.pop.com.
ns        IN        A         192.168.200.109
www       IN        A         192.168.200.109
[root@localhost named]# chgrp named kitty.zheng pop.zheng
[root@localhost named]# systemctl restart named
 [root@localhost named]# vim /etc/resolv.conf
 nameserver 192.168.200.109
[root@localhost named]# nslookup www.kitty.com
Server:         192.168.200.109
```

```
Address:        192.168.200.109#53

Name:   www.kitty.com
Address: 192.168.200.109

[root@localhost named]# nslookup www.pop.com
Server:         192.168.200.109
Address:        192.168.200.109#53

Name:   www.pop.com
Address: 192.168.200.109
```

windows测试：




修改server.xml

```
[root@localhost ~]# cd /usr/local/tomcat8/conf
[root@localhost conf]# vim server.xml
 #host1
<Host name="www.kitty.com"  appBase="webapps"
            unpackWARs="true" autoDeploy="true">
            <Context docBase="/web/kitty" path="" reloadable="false" />
#host2
 <Host name="www.pop.com"  appBase="webapps"
            unpackWARs="true" autoDeploy="true">
            <Context docBase="/web/pop" path="" reloadable="false" />
```

创建目录：

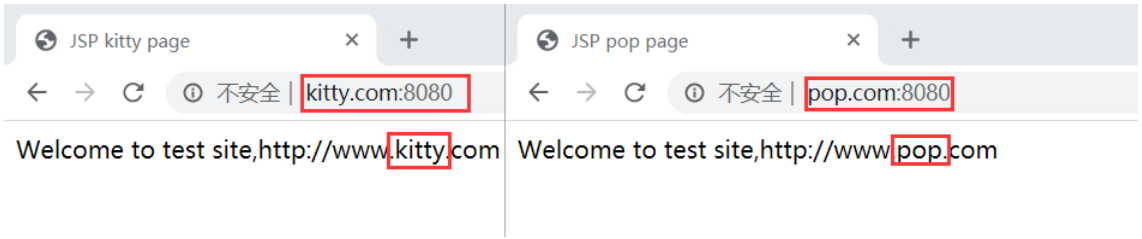
```
[root@localhost conf]# mkdir /web/{kitty,pop} -p
```

写网页：

```
[root@localhost conf]# vim /web/kitty/index.jsp
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<html>
```

```
        <head>
                <title>JSP kitty page</title>
    </head>
    <body>
        <% out.println("Welcome to test site,http://www.kitty.com");%>
        </body>
</html>
[root@localhost conf]# cp /web/kitty/index.jsp /web/pop/index.jsp
[root@localhost conf]# vim /web/pop/index.jsp
 1 <%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
 2 <html>
 3     <head>
 4         <title>JSP pop page</title>
 5     </head>
 6     <body>
 7         <% out.println("Welcome to test site,http://www.pop.com");%>
 8     </body>
 9 </html>
[root@localhost conf]# /usr/local/tomcat8/bin/shutdown.sh
[root@localhost conf]# /usr/local/tomcat8/bin/startup.sh
```

访问测试：




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

第一个services：

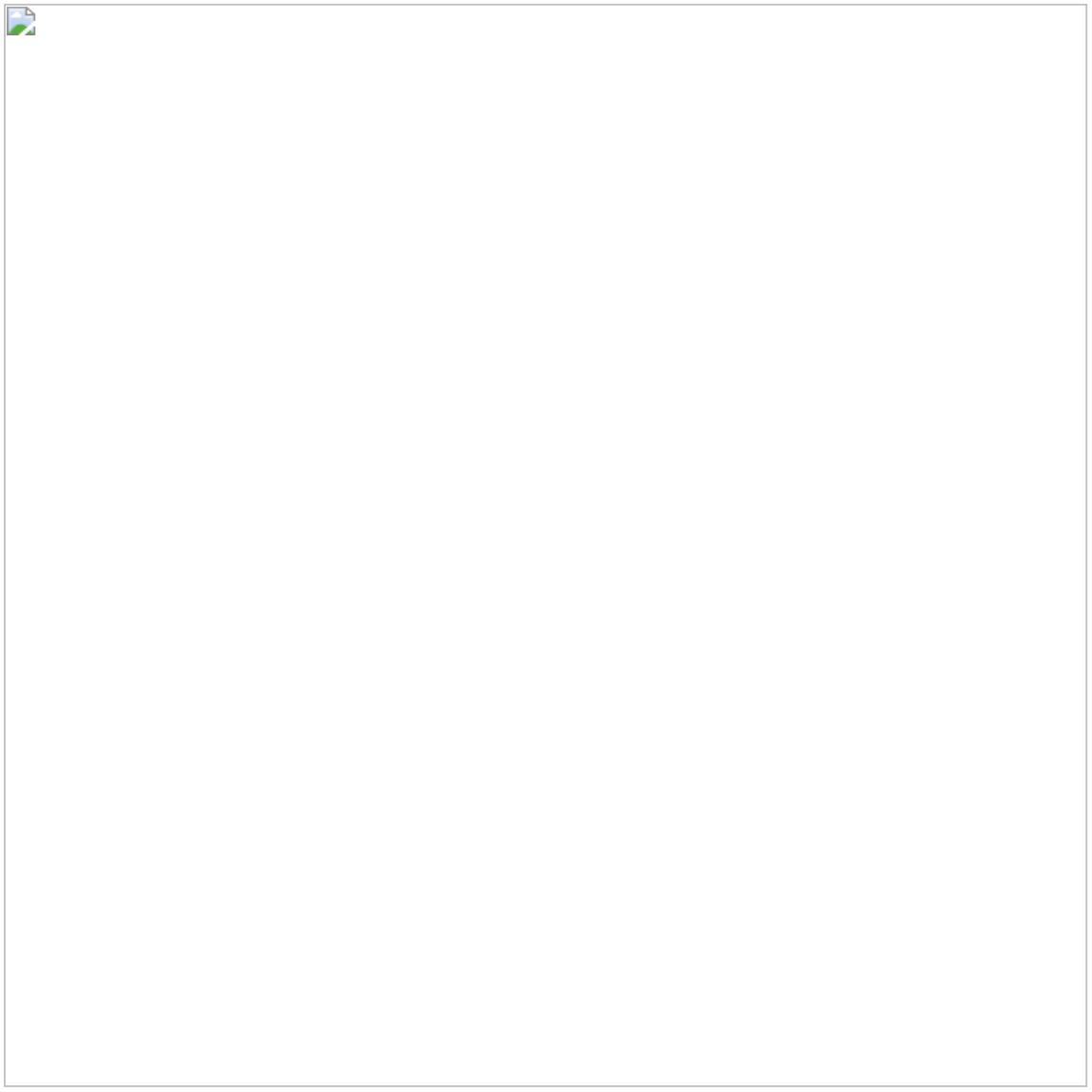
```
<Service name="Catalina1">
    ....................
 </Service>
```

第二个services：

```
<Service name="Catalina2">
```

```
 83      <Connector port="8090" protocol="HTTP/1.1"
 86      <Connector port="8010" protocol="AJP/1.3" redirectPort="8443" />
95              <Context docBase="/web/pop" path="" reloadable="false" />
#为了避免冲突，对第二个需要进行修改，域名可以相同，网页目录需要修改
.............................
</Service>
[root@localhost conf]#  /usr/local/tomcat8/bin/shutdown.sh
[root@localhost conf]#  /usr/local/tomcat8/bin/startup.sh
```

访问测试：



### 3.基于IP地址的虚拟主机

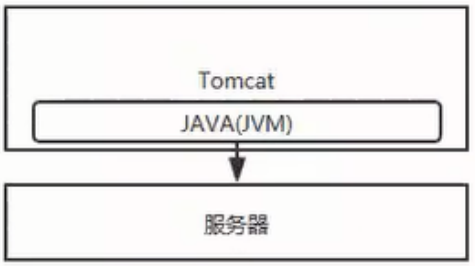
- 服务器使用多网卡配置多个IP地址
- 然后配置WEB服务器
- 把多个网站绑定在不同的IP上
- **该方式浪费Ip资源，Tomcat不支持该方式**

# 三、Tomcat多实例配置

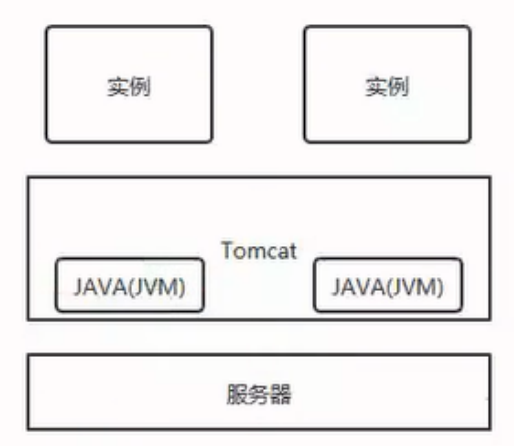
WEB应用程序需要部署在服务器上时

Tomcat 软件的部署方式可以分为以下几种

- 单实例单应用(webapps/a) -----访问会员商城，只有一个目录




- 单实例多应用(webapps/{a,b}) 既可以访问会员商城，还能访问论坛，有两个目录
- **多实例单应用 充分利用服务器底层的资源**




- **多实例多应用**

实例：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

缺点：

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

ps:虚拟主机也存在相同的问题，底层只有一个java进程，修改重启时会影响另外一个虚拟主机

➢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

**配置多实例实验：**

**还原环境：**

```
[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh
[root@localhost ~]# rm -rf /usr/local/tomcat8/
```

**解压并部署Tomcat程序：**

```
[root@localhost ~]# tar xf apache-tomcat-8.5.40.tar.gz
[root@localhost ~]# mv apache-tomcat-8.5.40 /usr/local/
[root@localhost ~]# ls /usr/local/
apache-tomcat-8.5.40  bin  etc  games  include  java  lib  lib64  libexec  sbin  share
src
```

**部署实例：**

**创建实例工作目录：**

```
[root@localhost ~]# mkdir -p /usr/local/tomcat_instance/tomcat1
[root@localhost ~]# cp -R /usr/local/apache-tomcat-8.5.40/conf/
/usr/local/tomcat_instance/tomcat1
[root@localhost ~]# ls /usr/local/tomcat_instance/tomcat1
conf
[root@localhost ~]# ls /usr/local/tomcat_instance/tomcat1/conf/
catalina.policy      context.xml           jaspic-providers.xsd  server.xml
tomcat-users.xsd
catalina.properties  jaspic-providers.xml  logging.properties    tomcat-users.xml
web.xml
```

**在Tomcat实例目录下创建/bin目录，并在/bin目录下创建实例启停脚本**

```
[root@localhost ~]# mkdir /usr/local/tomcat_instance/tomcat1/bin
[root@localhost ~]# touch /usr/local/tomcat_instance/tomcat1/bin/startup.sh
[root@localhost ~]# chmod +x /usr/local/tomcat_instance/tomcat1/bin/startup.sh
[root@localhost ~]# vim /usr/local/tomcat_instance/tomcat1/bin/startup.sh
    #!/bin/bash
    export CATALINA_HOME="/usr/local/apache-tomcat-8.5.40"   # 定义Tomcat部署路径
    export CATALINA_BASE="/usr/local/tomcat_instance/tomcat1"   # 实例的部署路径
    export CATALINA_TMPDIR="$CATALINA_BASE/temp"   #临时目录，在实例目录下创建
    export CATALINA_PID="$CATALINA_BASE/bin/tomcat.pid"   # 存放pid文件的位置
```

```
    export JAVA_OPTS="-server -Xms1024m -Xmx1024m -Djava.awt.headless=true -
    Dtomcat.name=tomcat1"   # 分配的内存相关信息
    #创建logs目录
    if [ ! -d "$CATALINA_BASE/logs" ]; then
      mkdir $CATALINA_BASE/logs
    fi
    #创建temp目录
    if [ ! -d "$CATALINA_BASE/temp" ]; then
      mkdir $CATALINA_BASE/temp
    fi
    #调用Tomcat启动脚本（在安装路径下调用）
    bash $CATALINA_HOME/bin/startup.sh "$@"
[root@localhost ~]# touch /usr/local/tomcat_instance/tomcat1/bin/startup.sh
[root@localhost ~]# chmod +x /usr/local/tomcat_instance/tomcat1/bin/shutdown.sh
[root@localhost ~]# vim /usr/local/tomcat_instance/tomcat1/bin/shutdown.sh
    #  在编辑文件的同时就创建了文件，可以两部合成一步，在加一下权限即可
    #!/bin/bash
    export CATALINA_HOME="/usr/local/apache-tomcat-8.5.40"
    export CATALINA_BASE="/usr/local/tomcat_instance/tomcat1"
    export CATALINA_TMPDIR="$CATALINA_BASE/temp"
    export CATALINA_PID="$CATALINA_BASE/bin/tomcat.pid"

    bash $CATALINA_HOME/bin/startup.sh "$@"
```

修改每个Tomcat实例中的server/xml的端口（8005 8080 8009）

注意：多实例的情况下，每个实例之间的端口不能冲突

```
#在实例1下面创建一个存放目录
[root@localhost ~]# mkdir -p /usr/local/tomcat_instance/tomcat1/webapps/ROOT
#在其目录下写一个网页文件
[root@localhost ~]# vim /usr/local/tomcat_instance/tomcat1/webapps/ROOT/index.jsp
    <html>
    <title>Tomcat-1</title>
    <body>
        Hello This is Tomcat-1.
    </body>
    </html>
```

启动实例1的脚本：

```
[root@localhost ~]# /usr/local/tomcat_instance/tomcat1/bin/startup.sh
[root@localhost ~]# netstat -lnpt | grep java
```

```
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN
65359/java
tcp6       0      0 :::8009                 :::*                    LISTEN
65359/java
tcp6       0      0 :::8080                 :::*                    LISTEN
65359/java
```


将/usr/local/tomcat_instance/下的Tomcat1拷贝为Tomcat2

```
[root@localhost ~]# cd /usr/local/tomcat_instance/
[root@localhost tomcat_instance]# ls
tomcat1
[root@localhost tomcat_instance]# cp -r tomcat1/ tomcat2
[root@localhost tomcat_instance]# cd tomcat2
[root@localhost tomcat2]# ls
bin  conf  logs  temp  webapps  work
[root@localhost tomcat2]# ls bin/
shutdown.sh  startup.sh  tomcat.pid
[root@localhost tomcat2]# rm -rf bin/tomcat.pid
      # 如果不删除，当我们启动脚本的时候，会报错，有pid文件证明已经启动了，就会报错
```

修改启停文件：

```
[root@localhost tomcat2]# vim bin/shutdown.sh
  3 export CATALINA_BASE="/usr/local/tomcat_instance/tomcat2"
[root@localhost tomcat2]# vim bin/startup.sh
  3 export CATALINA_BASE="/usr/local/tomcat_instance/tomcat2"
  6 export JAVA_OPTS="-server -Xms1024m -Xmx1024m -Djava.awt.headless=true -
Dtomcat.name=tomcat2"
```

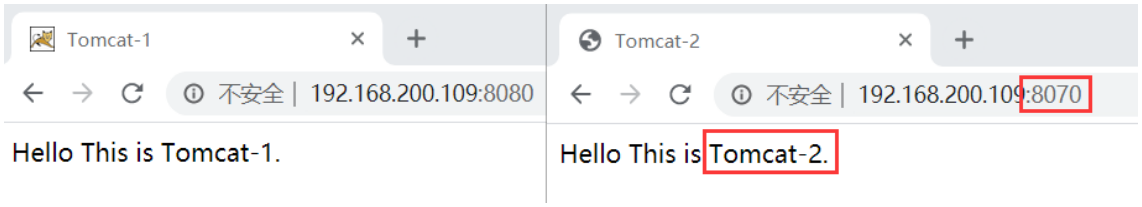
修改主配置文件：

```
[root@localhost tomcat2]# vim conf/server.xml
 22 <Server port="8004" shutdown="SHUTDOWN">
 70   <Connector port="8070" protocol="HTTP/1.1"
117   <Connector port="8008" protocol="AJP/1.3" redirectPort="8443" />
```

修改网页文件：

```
[root@localhost tomcat2]# vim webapps/ROOT/index.jsp
    <html>
```

```
    <title>Tomcat-2</title>
    <body>
        Hello This is Tomcat-2.
    </body>
    </html>
[root@localhost tomcat2]# /usr/local/tomcat_instance/tomcat2/bin/startup.sh
[root@localhost tomcat2]# netstat -lnpt | grep java
tcp6       0      0 127.0.0.1:8004          :::*                    LISTEN
66255/java
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN
65359/java
tcp6       0      0 :::8070                 :::*                    LISTEN
66255/java
tcp6       0      0 :::8008                 :::*                    LISTEN
66255/java
tcp6       0      0 :::8009                 :::*                    LISTEN
65359/java
tcp6       0      0 :::8080                 :::*                    LISTEN
65359/java
[root@localhost tomcat2]# tail -f /usr/local/tomcat_instance/tomcat2/logs/catalina.out
```




再来一次，做实例3：

```
[root@localhost ~]# cd /usr/local/tomcat_instance/
[root@localhost tomcat_instance]# cp -r tomcat1 tomcat3
```

**修改cd /usr/local/tomcat_instance/tomcat3下的各个文件：**

**bin：**

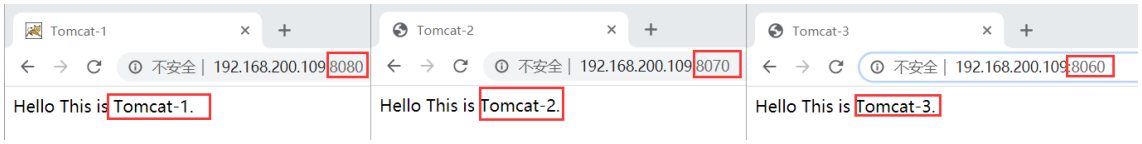
```
[root@localhost tomcat3]# rm -rf bin/tomcat.pid
  3 export CATALINA_BASE="/usr/local/tomcat_instance/tomcat3"
[root@localhost tomcat3]# vim bin/startup.sh
  3 export CATALINA_BASE="/usr/local/tomcat_instance/tomcat3"
  6 export JAVA_OPTS="-server -Xms1024m -Xmx1024m -Djava.awt.headless=true -
Dtomcat.name=tomcat3"
```

**conf：**

```
[root@localhost tomcat3]# vim conf/server.xml
 22 <Server port="8003" shutdown="SHUTDOWN">
```

```
 69     <Connector port="8060" protocol="HTTP/1.1"
116     <Connector port="8007" protocol="AJP/1.3" redirectPort="8443" />
[root@localhost tomcat3]# vim webapps/ROOT/index.jsp
<html>
<title>Tomcat-3</title>
<body>
    Hello This is Tomcat-3.
</body>
</html>
[root@localhost tomcat3]# /usr/local/tomcat_instance/tomcat3/bin/startup.sh
[root@localhost tomcat3]# !net
netstat -lnpt | grep java
tcp6       0      0 :::8060                 :::*                    LISTEN
66616/java
tcp6       0      0 127.0.0.1:8003          :::*                    LISTEN
66616/java
tcp6       0      0 127.0.0.1:8004          :::*                    LISTEN
66255/java
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN
65359/java
tcp6       0      0 :::8070                 :::*                    LISTEN
66255/java
tcp6       0      0 :::8007                 :::*                    LISTEN
66616/java
tcp6       0      0 :::8008                 :::*                    LISTEN
66255/java
tcp6       0      0 :::8009                 :::*                    LISTEN
65359/java
tcp6       0      0 :::8080                 :::*                    LISTEN
65359/java
[root@localhost tomcat3]# tail -f /usr/local/tomcat_instance/tomcat3/logs/catalina.out
[root@localhost tomcat3]# ps aux | grep java
root       65359  0.2  9.7 3609812 198096 pts/0  Sl   12:22   0:13
/usr/local/java/bin/java -
Djava.util.logging.config.file=/usr/local/tomcat_instance/tomcat1/conf/logging.properties
 -Djava.util.logging.manager
=org.apache.juli.ClassLoaderLogManager -server -Xms1024m -Xmx1024m -
Djava.awt.headless=true -Dtomcat.name=tomcat1 -Djdk.tls.ephemeralDHKeySize=2048 -
```

```
.....
root       66255  1.0  9.0 3609812 184128 pts/0  Sl   13:48   0:15
/usr/local/java/bin/java -
Djava.util.logging.config.file=/usr/local/tomcat_instance/tomcat2/conf/logging.properties
 -Djava.util.logging.manager
=org.apache.juli.ClassLoaderLogManager -server -Xms1024m -Xmx1024m -
Djava.awt.headless=true -Dtomcat.name=tomcat2 -Djdk.tls.ephemeralDHKeySize=2048 -
.....
root       66616  3.3  5.4 3601620 111588 pts/0  Sl   14:09   0:06
/usr/local/java/bin/java -
Djava.util.logging.config.file=/usr/local/tomcat_instance/tomcat3/conf/logging.properties
 -Djava.util.logging.manager
=org.apache.juli.ClassLoaderLogManager -server -Xms1024m -Xmx1024m -
Djava.awt.headless=true -Dtomcat.name=tomcat3 -Djdk.tls.ephemeralDHKeySize=2048 -
.....
root       66712  0.0  0.0 112720   980 pts/0    S+   14:13   0:00 grep --color=auto
java
```




**在企业实际生产环境中：**

- 一个实例占用内存：4G~8G　　普遍400并发;一般是200并发左右
- 一个64G的机器：通常6~8个实例
- 虽然它是通过端口来访问的，我们可以通过nginx来做调度

```
[root@localhost local]# ls
apache-tomcat-8.5.40  etc    include  lib    libexec  share  tomcat_instance
bin                   games  java     lib64  sbin     src
```

- 服务器启动之后，会分别在相应的实例目录下生成 logs、 temp、work等目录
- 另外也可以在实例目录下创建lib目录，用于存放app的 jar，所以各个实例之间是独立的，
- 现在来看实例的安装目录，就和tomcat的安装包解压后的目录结构一样了，
- 但是所有实例共享同一套tomcat安装程序的 bin和 lib
- 后面如果需要升级tomcat或修改tomcat脚本的相关配置，只需要更新这一套程序就行，也
- 方便了日后的维护。多实例部署最大作用就是最大化利用服务器资源

# 四、Nginx基于Tomcat多实例的负载均衡

## 安装nginx

```
[root@localhost  ~]#  systemctl  stop  firewalld
[root@localhost  ~]#  iptables  -F
[root@localhost  ~]#  setenforce  0
setenforce:  SELinux  is  disabled
[root@localhost  ~]#  useradd  -M  -s  /sbin/nologin  nginx
[root@localhost  ~]#  yum  -y  install  pcre-devel  zlib-devel  openssl-devel
[root@localhost  ~]#  tar  xf  nginx-1.15.9.tar.gz  -C  /usr/src/
[root@localhost  ~]#  cd  /usr/src/nginx-1.15.9/
[root@localhost nginx-1.15.9]#./configure --prefix=/usr/local/nginx --user=nginx --
group=nginx && make && make install
[root@localhost nginx-1.15.9]# ln -s /usr/local/nginx/sbin/nginx /usr/local/bin/
[root@localhost nginx-1.15.9]# nginx
```

## 配置nginx.conf

```
[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf
      2 user  nginx nginx;
      3 worker_processes  2;
      4 worker_cpu_affinity 01 10;
      5 worker_rlimit_nofile 102400;     # ulimit 102400
      6 error_log  logs/error.log;
     11 pid  logs/nginx.pid;
     14 events {
     15     worker_connections  4069;
     16     use epoll;
     17 }
     20 http {
     21     include       mime.types;
     22     default_type  application/octet-stream;
     23
     24     log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
     25                       '$status $body_bytes_sent "$http_referer" '
     26                       '"$http_user_agent" "$http_x_forwarded_for"';
     27
     28     access_log  logs/access.log  main;
     29
     30     sendfile        on;
     34     keepalive_timeout  65;
     52         location / {
```

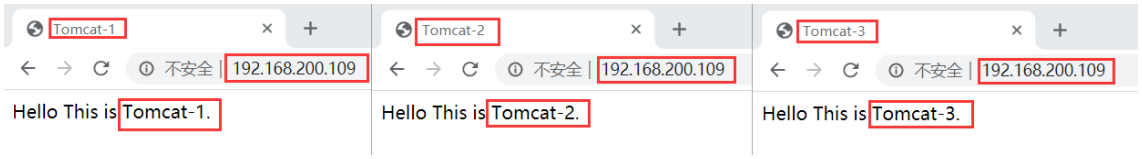
```
53              proxy_pass http://tomcat;
54          }
55          location ~ \.html$ {
56              root   html;
57              index  index.html index.htm;
58          }
```

```
36      upstream tomcat {
37      server 192.168.200.109:8080 weight=1;
38      server 192.168.200.109:8070 weight=1;
39      server 192.168.200.109:8060 weight=1;
40      }
```

```
44      server {
45          listen       80;
46          server_name  localhost;
47
48          charset utf-8;
```

```
[root@localhost ~]# killall -HUP nginx
```

**多实例负载均衡测试访问：反复刷新**




# 五、Tomcat管理功能使用

**注意:测试功能，生产环境不要用　----不安全**

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

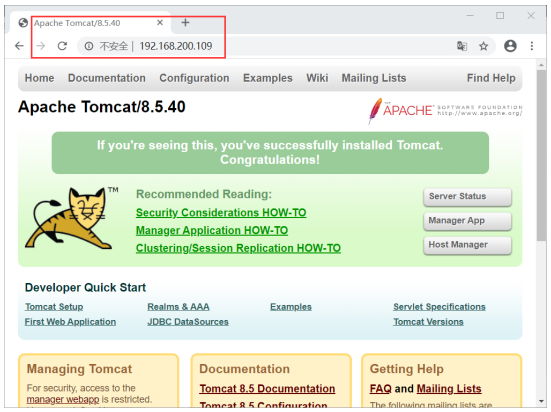
在默认情况下是处于禁用状态的

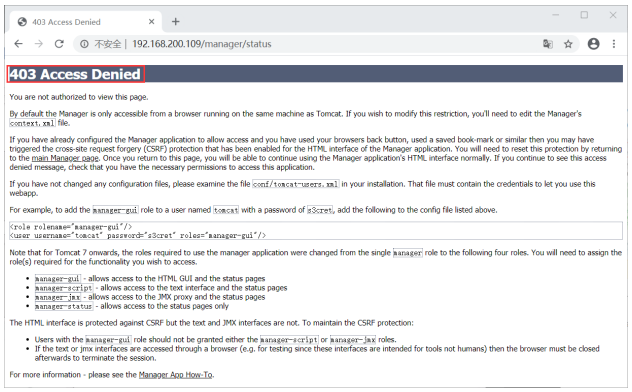
如果需要开启这个功能，需要配置管理用户，即配置 tomcat-users.xml 文件

**还原环境：**

```
[root@localhost ~]# cd /usr/local/
[root@localhost local]# rm -rf tomcat_instance*
[root@localhost local]# ls
apache-tomcat-8.5.40  bin  etc  games  include  java  lib  lib64  libexec  nginx  sbin
share  src
[root@localhost local]# mv apache-tomcat-8.5.40/ tomcat8
[root@localhost local]# killall -9 java
[root@localhost local]# ps aux | grep java
root      71791  0.0  0.0 112720   980 pts/0    S+   17:07   0:00 grep --color=auto
java
```

```
[root@localhost local]# cd
[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh
[root@localhost ~]# netstat -lnpt | grep java
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN
71825/java
tcp6       0      0 :::8009                 :::*                    LISTEN
71825/java
tcp6       0      0 :::8080                 :::*                    LISTEN
71825/java
```



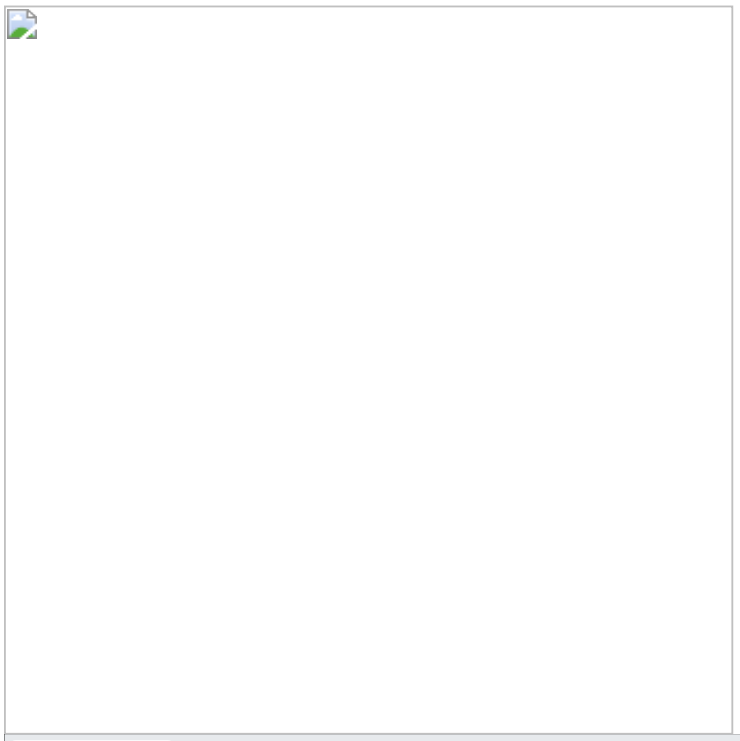

配用户，解地址

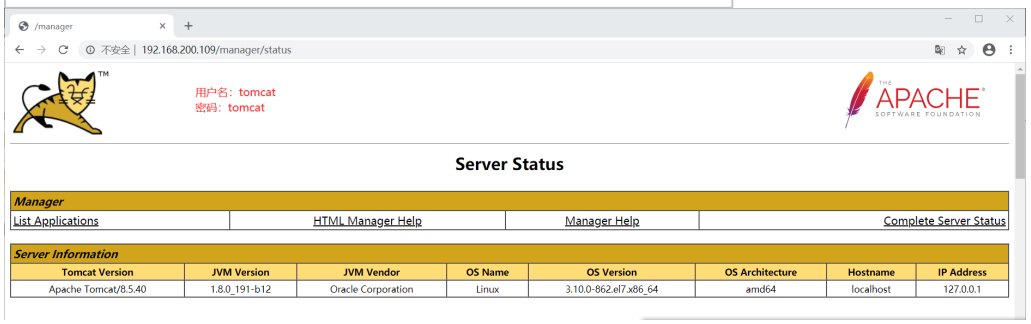
```
[root@localhost ~]# vim /usr/local/tomcat8/conf/tomcat-users.xml
 44 <role rolename="manager-gui"/>        #角色    角色名
 45 <role rolename="admin-gui"/>          # 管理员角色
 46 <user username="tomcat" password="tomcat" roles="manager-gui,admin-gui"/>
         # 用户名   密码     角色为 gui 图形
```

**默认访问webapp,修改里面的文件：**

```
[root@localhost ~]# vim /usr/local/tomcat8/webapps/manager/META-INF/context.xml
 19   <!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"
 20           allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
       #将这行注释，因为授权地址，默认是127……，即只允许127这个访问
[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh
[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh
```

访问测试：






# 六、WEB站点部署(jpress)

类似部署论坛

上线的代码有两种方式

第一种方式：直接将程序目录放在webapps目录下面

第二种方式：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

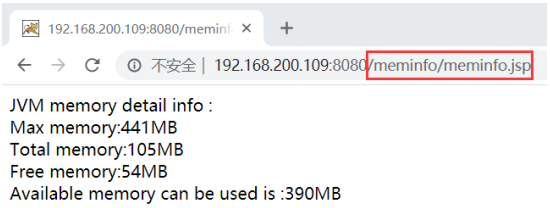
上传meminfo.war包到/usr/local/tomcat8/webapps目录中

## 1.部署tomcat内存检测包（meminfo.isp ）

```
[root@localhost ~]# vim /usr/local/tomcat8/conf/server.xml
149              unpackWARs="true" autoDeploy="true">
[root@localhost ~]# cd /usr/local/tomcat8/webapps/
[root@localhost webapps]# ls
docs  examples  host-manager  manager  ROOT
```

```
[root@localhost webapps]# rz      # 一个查看内存的war包
[root@localhost webapps]# ls
docs  examples  host-manager  manager  meminfo.war  ROOT
[root@localhost webapps]# ls
docs  examples  host-manager  manager  meminfo  meminfo.war  ROOT
   # meminfo.war 这个包最好不要删，如果删了，有些应用程序是访问不了的
   # 隔几秒会自动解压
[root@localhost webapps]# ls meminfo
meminfo.jsp  META-INF
[root@localhost webapps]# cat meminfo/meminfo.jsp
<%
Runtime rtm = Runtime.getRuntime();
long mm = rtm.maxMemory()/1024/1024;
long tm = rtm.totalMemory()/1024/1024;
long fm = rtm.freeMemory()/1024/1024;

out.println("JVM memory detail info :<br>");
out.println("Max memory:"+mm+"MB"+"<br>");
out.println("Total memory:"+tm+"MB"+"<br>");
out.println("Free memory:"+fm+"MB"+"<br>");
out.println("Available memory can be used is :"+(mm+fm-tm)+"MB"+"<br>");
%>
# 都是一些变量来获取内存相关信息，做一个输出
```

**访问测试：当小的监控来用**




## 2.部署开源站点（jpress）

### 安装配置数据库

```
[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: 5.5.64-MariaDB MariaDB Server

    Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
```

```
    Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
    # 创建数据库，声明字符集
    MariaDB [(none)]> create database jpress DEFAULT CHARACTER SET utf8;
    Query OK, 1 row affected (0.00 sec)
    # 授权 所有权限，用户名localhost，密码123456
    MariaDB [(none)]> grant all on jpress.* to 'jpress'@'localhost' identified by
    '123456';
    Query OK, 0 rows affected (0.00 sec)
    #  刷新授权表，目的使上一条语句生效
    MariaDB [(none)]> flush privileges;
    Query OK, 0 rows affected (0.00 sec)

    MariaDB [(none)]> exit
    Bye
[root@localhost ~]# rz
[root@localhost ~]# tar xf 0.4.0.tar.gz
[root@localhost ~]# ls
0.4.0.tar.gz     apache-tomcat-8.5.40.tar.gz  jdk-8u191-linux-x64.tar.gz  nginx-
1.15.9.tar.gz
anaconda-ks.cfg  initial-setup-ks.cfg         jpress-0.4.0
[root@localhost ~]# cd jpress-0.4.0/
[root@localhost jpress-0.4.0]# ls
DOC.md  jpress  LICENSE  README.md  wars
[root@localhost jpress-0.4.0]# cd wars/
[root@localhost wars]# ls
jpress-web-newest.war
[root@localhost wars]# mv jpress-web-newest.war /usr/local/tomcat8/webapps/
[root@localhost wars]# ls /usr/local/tomcat8/webapps/
docs      host-manager       jpress-web-newest.war  meminfo      ROOT
examples  jpress-web-newest  manager                meminfo.war
  # war包会自动解压
```

**直接来访问：**





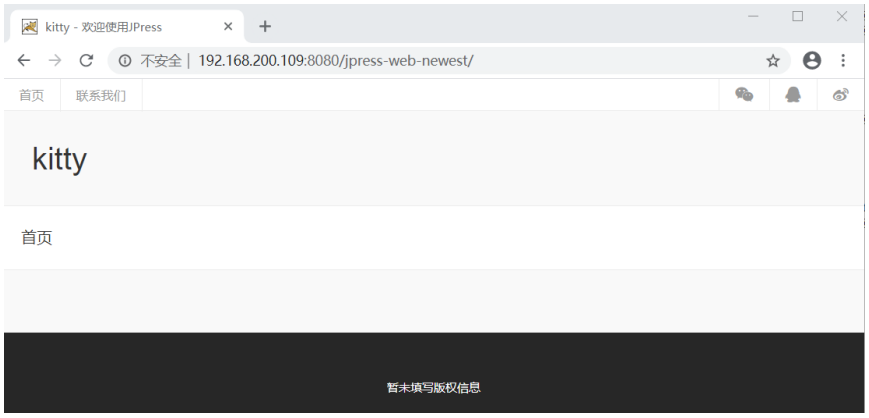




web容器就是Tomcat

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

来~重新访问，不就行了吗




ps：

手动代码上线

自动代码上线(上面做的就是)

开发人员，给一个war包，会给你说几点几分上线

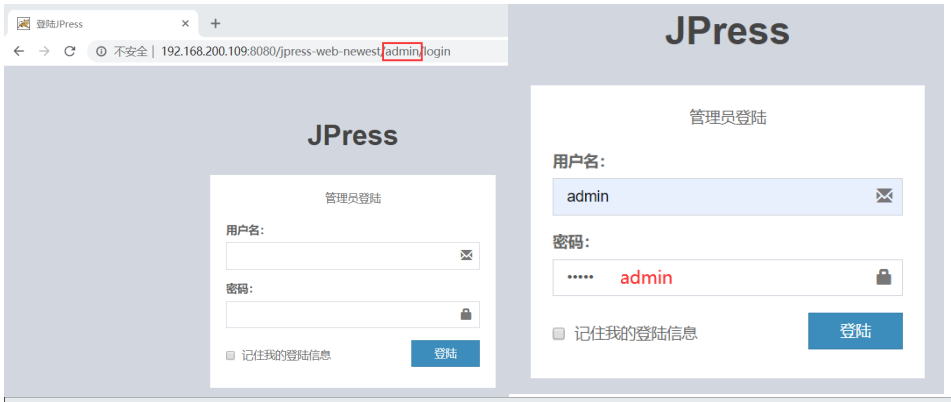
上线新版本代码时，先将旧版本代码（比如jpress-web-newest）进行压缩打包，然后将新版本war包，扔进来自动解压即可

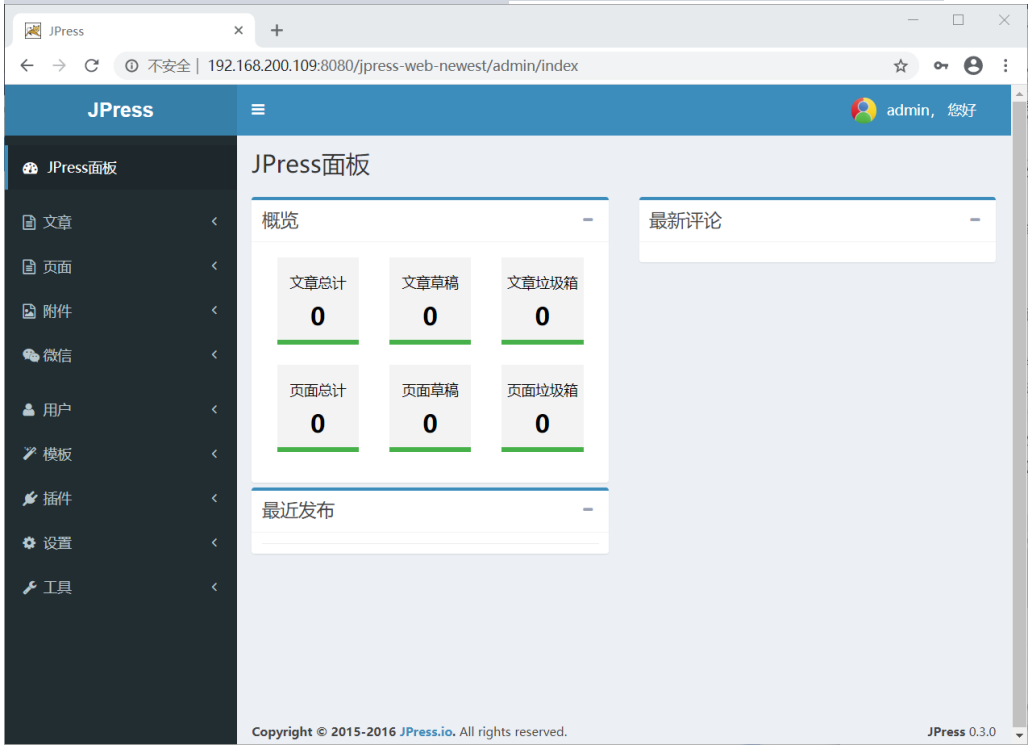
打包压缩旧版本的目的，实现快速回滚，一旦新上线的版本出现问题，就将新的移除，再将旧的版本拿回来，保证业务第一时间对外提供访问

**jpress是用来写博客的，bbs,wordpress,jpress**

开源博客系统

进入后台：




制作war包：

将.jsp文件打包压缩为zip包，将后缀改为.war，就制作成啦

开发人员主要用的jar来压缩的

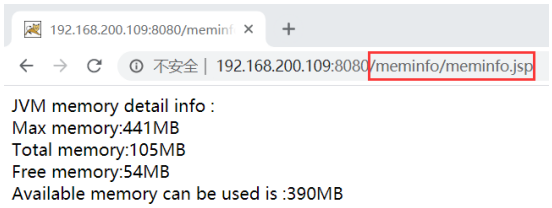
开发--->代码仓库(SVN/Git) <------Jenkins(MAVEN) -----> War ---> Tomcat

全自动，自动化

# 七、监控Tomcat状态

## 方法一：开发Java监控页面 ----出力点在开发人员那边

```
[root@localhost ~]# mkdir /usr/local/tomcat8/webapps/memtest/
[root@localhost ~1# vim /usr/local/tomcat8/webapps/memtest/meminfo.jsp
<%
Runtime rtm = Runtime.getRuntime();
long mm = rtm.maxMemory()/1024/1024;
long tm = rtm.totalMemory()/1024/1024;
long fm = rtm.freeMemory(/1024/1024;

out.println("JVM memory detail info :<br>");
out.println("Max memory:"+mm+"MB"+"<br>");
out.println("Total memory:"+tm+"MB"+"<br>");
out.println("Free memory:"+fm+"MB"+"<br>");
out.println("Available memory can be used is :"+(mm+fm-tm)+"MB"+"<br>");
%>
```




## 方法二：使用jps命令进行监控　（不建议使用）

```
[root@localhost ~]# jps
75889 Jps
75316 Bootstrap
# 查看运行的Java类进程
[root@localhost ~]# jps -lvm
75316 org.apache.catalina.startup.Bootstrap start -
Djava.util.logging.config.file=/usr/local/tomcat8/co
nf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
-Djdk.tls.ephemeralDHKeySize=2048 -
Djava.protocol.handler.pkgs=org.apache.catalina.webresources -
Dorg.apache.catalina.security.SecurityListener.UMASK=0027 -Dignore.endorsed.dirs= -
Dcatalina.base=/usr/local/tomcat8 -Dcatalina.home=/usr/local/tomcat8 -
Djava.io.tmpdir=/usr/local/tomcat8/temp75910 sun.tools.jps.Jps -lvm -
Dapplication.home=/usr/local/java -Xms8m
```

## 方法三：Tomcat远程监控功能--jconsole

**特点:**

**使用普及**

**后续可以集成到zabbix**

修改配置文件，在Tomcat开启远程监控

```
[root@localhost ~]# vim /usr/local/tomcat8/bin/catalina.sh
  3 CATALINA_OPTS="$CATALINA_OPTS
  4 -Dcom.sun.management.jmxremote     # 开启远程监控功能
  5 -Dcom.sun.management.jmxremote.port=12345  # jmx远程端口,监听端口是12345,ZABBIX添加时必须一致
  6 -Dcom.sun.management.jmxremote.authenticate=false    # 不开启用户密码验证
  7 -Dcom.sun.management.jmxremote.ssl=false
  8 -Djava.rmi.server.hostname=192.168.200.107"          # 运行Tomcat服务（注意IP不要填错）
[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh
[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh
tcp6       0      0 :::8080                 :::*                    LISTEN
76247/java
tcp6       0      0 :::12345                :::*                    LISTEN
76247/java
```

在windows上监控Tomcat

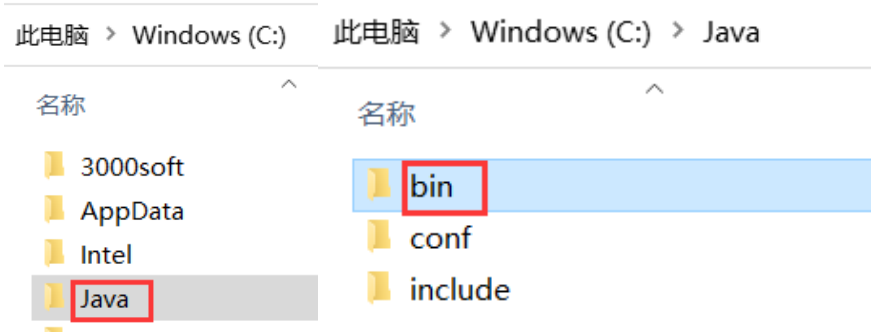
注意：windows需要安装jdk环境

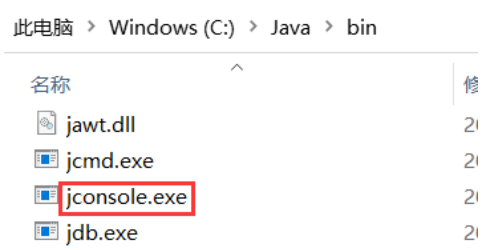
下载地址：http://www.oracle.com/technetwork/java/javase/downloads/index.html

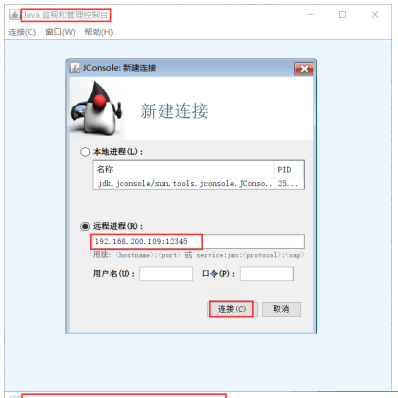
将jdk直接解压到C:\java\中，双击运行C:\java\jdk-12\bin\jconsole.exe

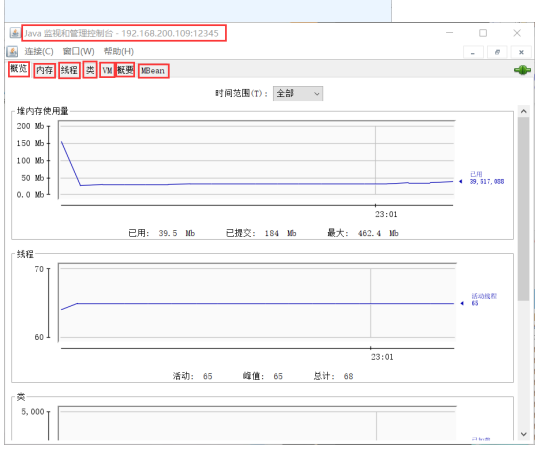
当没修改任何变量的时候，可以修改一下根目录下的名称为Java

后期放在一个环境里进行调整预值，进行邮箱报警






# 八、Tomcat故障排查步骤

a. 查看catalina.out（日志）

b. 使用sh show-busy-java-threads.sh脚本进行检测

脚本下载地址

https://files.cnblogs.com/files/clsn/show-busy-java-threads.sh.

注意：脚本执行故障排查效果不会太好，最好别使用

```
[root@localhost ~]# cd /usr/local/tomcat8/logs/
[root@localhost logs]# ls
catalina.2020-04-08.log  host-manager.2020-04-08.log  localhost_access_log.2020-04-08.txt
catalina.out             localhost.2020-04-08.log     manager.2020-04-08.log
```

catalina.out ，catalina.2020-04-08.log 需要排错时主要处理这两个文件就可以，两个都一样，一般用catalina.out

localhost_access_log.2020-04-08.txt 用户访问日志

一般用这三个文件即可

# 九、Tomcat安全优化

8080 http 8005 telnet（能不开就不开，将端口伪装一下）8009 ajp

## 1.telnet 管理端口保护（保护）

| 类别 | 配置内容及说明 | 标准配置 | 备注 |
| --- | --- | --- | --- |
| telnet管理端口保护 | 1.修改默认的8005管理 | <Server port="8527" | 1.以上配置项的配置内容 |

| | | | |
|---|---|---|---|
| | 端口为不易猜测的端口(大于1024)<br>2.修改SHUTDOWN<br>指令为其他字符串 | shutdown="crushlinux"> | 只是建议配置，可以按照服务实际情况进行合<br><br>理配置，但要求端口配置在80008999之间 |

端口修改，伪装

```
[root@localhost ~]# vim /usr/local/tomcat8/conf/server.xml
 22 <Server port="8005" shutdown="guanji">
# 系统22端口企业也会修改
```

## 2. ajp连接端口保护(推荐)

| 类别 | 配置内容及说明 | 标准配置 | 备注 |
|---|---|---|---|
| Ajp 连接端口保护 | 1.修改默认的ajp8009端口为不易冲突的大于1024端口;<br>2.通iptables（防火墙）规则限制ajp 端口访问的权限仅为线上机器<br>#或者注释掉 | <Connector port="8528" protocol="AJP/1.3" /> | 以上配置项的配置内容仅为建议配置，请按照服务实际情况进行合理配置,但要求端口配置在8000~8999之间;保护此端口的目的在于防止线下的测试流量被mod. jk转发至线上tomcat服务器; |

```
116     <!--<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" /> -->
```

## 3.禁用管理端(强制)

| 类别 | 配置内容及说明 | 标准配置 | 备注 |
|---|---|---|---|
| 禁用管理端 | 1.删除默认的{Tomcat安装目录}/conf/tomcat-users.xml文件，重启会Tomcat后将会自动生成新的文件<br>2.删除{Tomcat安装目录}/webapps下默认的所有目录和文件<br>3.将tomcat应用根目录配置为tomcat安装目录以外的目录 | <Context path=""docBase="/web/webapps" debug="0" reloadable="false" crossContext="true*/ > | 对于前端web 模块，Tomcat管理端属于tomcat的高危安全陷患,一旦被攻破，黑客通过上传web shell的方式将会直接取得服务器的控制权,后果极其严重 |

不要让任何人看到后台管理页面

## 4.降权启动(强制)

| 类别 | 配置内容及说明 | 标准配置 | 备注 |
|---|---|---|---|
| 降权启动 | 1.tomcat启动用户权限必须为非root权限，尽量降低tomcat启动用户的目录访问权限 | | 避免一旦tomcat服务被入侵，黑客直接获取高级用户权限危害整个server的安全 |

| | | | |
|---|---|---|---|
| | 2.如需直接对外使用80端口,可通过普通账号启动后，配置iptables规则进行转发 | | |

[root@localhost ~]# pe aux | grep java

从这条命令可以看出，Tomcat是以root管理员的身份运行的，权限大，风险也大

让Tomcat以用户的身份运行，降低权限，降低风险

useradd  tomcat

cp -a /usr/local/tomcat8 /home/tomcat/tomcat8_1/

chown -R tomcat.tomcat /home/tomcat/tomcat8_1/

su -c '/home/tomcat/tomcat8 1/bin/startup.sh' tomcat

ps -ef | grep tomcat

## 5.文件列表访问控制(强制)

| 类别 | 配置内容及说明 | 标准配置 | 备注 |
|---|---|---|---|
| 文件列表访问控制 | 1.conf/web.xml文件中default部分listings的配置必须为false | <init-param> <paramname>listings</param-name> <param-value>false</ param-value> </init-param> | false为不列出目录文件，true为允许列出，默认为false |

禁止列出目录：不允许它以列表的形式让用户来查看

```
[root@localhost ~]# vim /usr/local/tomcat8/conf/web.xml
 110         <init-param>
 111             <param-name>listings</param-name>
 112             <param-value>false</param-value>
 113         </init-param>
```

## 6.版本信息隐藏(强制)

| 类别 | 配置内容及说明 | 标准配置 | 备注 |
|---|---|---|---|
| 版本信息隐藏 | 1.修改conf/web.xml，重定向403.404以及500等错误到指定的错误页面; 2.也可以通过修改应用程序目录下的WEB-INF/web.xml下的配置进行错误页面的重定向 | <error-page> <error-code>403</error-code> <location>/forbidden.jsp</location> </error-page> <error-page> <error-code>404</error-code> <location>/notfound.jsp</location> </error-page> <error-page> <error-code>500</error-code> | 在配置中对一些常见错误进行重定向,避免当出现错误时tomcat默认显示的错误页面暴露服务器和版本信息;必须确保程序根目录下的错误页面已经存在 |

| | | | |
|---|---|---|---|
| | | <location>/systembusy.jsp</location><br></error-page> | |

## 7. Server header重写(推荐)

| 类别 | 配置内容及说明 | 标准配置 | 备注 |
|---|---|---|---|
| Server header重写 | 在HTTP Connector配置中加入server的配置 | server "webserver" | 当tomcatTTP端口直接提供web服务时此配置生效，加入此配置，将会替换ht响应Server header部分的默认配置，默认是Apache Coyote/1.1 |

## 8.访问限制(可选)

| 类别 | 配置内容及说明 | 标准配置 | 备注 |
|---|---|---|---|
| 访问限制 | 通过配置，限定访问的ip来源 | <Context path="" docBase="/web/webapps" debug="0" reloadable="false" crossContext="true"><br><ValveclassName="org.apache.catalina.valves.RemoteAddrValve" allow="61.148.18.138,61.135.165.* deny="*.*.*.*"/><br></Context> | 通过配置信任ip的白名单，拒绝非白名单ip的访问，此配置主要是针对高保密级别的系统，一般产品线不需要 |

## 9.起停脚本权限回收(推荐)

| 类别 | 配置内容及说明 | 标准配置 | 备注 |
|---|---|---|---|
| 起停脚本权限回收 | 去除其他用户对Tomcat的bin目录下shutdown.sh、startup.sh. catalina.sh的可执行权限 | chmod -R 744 tomcat/bin/* | 防止其他用户有起停线上Tomcat的权限 |

查看 [root@localhost ~]# ls -l /usr/local/tomcat8/bin/

发现有些文件的所属组也有权限，将它的权限降低

修改：

```
[root@localhost ~]# cd /usr/local/tomcat8/bin/
[root@localhost bin]# chmod -R 744 *
[root@localhost bin]# ls -l /usr/local/tomcat8/bin/
```

## 10.访问日志格式规范(推荐)

| 类别 | 配置内容及说明 | 标准配置 | 备注 |
|---|---|---|---|
| 访问日志格式规范 | 开启Tomcat默认访问日 | <Valve | 开启Referer 和User |

| | | | |
|---|---|---|---|
| 访问日志格式规范 | 开启Tomcat默认访问日志中的Referer和User-Agent 记录 | <Valve className="org.apache.catalina.valves.AccessLogValue" directory="logs" preix"localhost_access_log" suffix=".txt" pattern="%h %l %u %t %r %s %b %{Referer}i %{UserAgent)i %D" resolveHosts="false"/> | 开启Referer 和User-Agent是为了一旦出现<br><br>安全问题能够更好的根据日志进行问题排查 |

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

查看服务状态：

```
[root@localhost ~]# netstat -lnpt | grep java
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN
78911/java
tcp6       0      0 :::8009                 :::*                    LISTEN
78911/java
tcp6       0      0 :::33580                :::*                    LISTEN
78911/java
tcp6       0      0 :::8080                 :::*                    LISTEN
78911/java
tcp6       0      0 :::12345                :::*                    LISTEN
78911/java
tcp6       0      0 :::38202                :::*                    LISTEN
78911/java
[root@localhost ~]# tail -f /usr/local/tomcat8/logs/localhost_access_log.2020-04-09.txt
192.168.200.128 - - [09/Apr/2020:18:54:13 +0800] "GET /bg-upper.png HTTP/1.1" 200 3103
192.168.200.128 - tomcat [09/Apr/2020:19:36:37 +0800] GET /manager/status HTTP/1.1 200
6891 http://192.168.200.109:8080/ ???{???UserAgent)i 3
```

## 11、页面超时

```
581  <session-config>
582     <session-timeout>30</session-timeout>
583  </session-config>
```

## 12、默认页面

```
[root@localhost ~]# vim /usr/local/tomcat8/conf/web.xml
4679 <welcome-file-list>
4680    <welcome-file>index.htmk</welcome-file>
```

```
4681     <welcome-file>index.htm</welcome-file>
4682     <welcome-file>index.jsp</welcome-file>
4683 </welcome-file-list>
```

### 13、配置网页传输压缩

注:如若有apache、nginx 等做代理，tomcat 则不必配置传输压缩

```
[root@localhost ~]# vim /usr/local/tomcat8/conf/server.xml
69 <Connector port="8080" protocol="HTTP/1.1"
70 connectionTimeout="20000".
71 redirectPort="8443"
72 compression="on"        #开启压缩
73 compressionMinSize="50"     #指定最小的压缩文件，单位是字节
74 noCompressionUserAgents="oilla, Traviata"        #此浏览器类型不进行压缩
75 compressableMimeType="text/html,text/xml,text/javscript,text/css,text/plain"/>
                        #文件的式
[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh
[root@localhost ~]# /us/local/tomcat8/bin/startup.sh
```

## 十、Tomcat运行模式

运行模式的选择对工作效率影响很大

### 1、同步概念

➢ 同步 ：

自己亲自出马持银行卡到银行取钱（使用同步IO时，Java自己处理IO读写）

同步：使用同步IO时，Java自己处理IO读写（什么都自己亲力亲为）

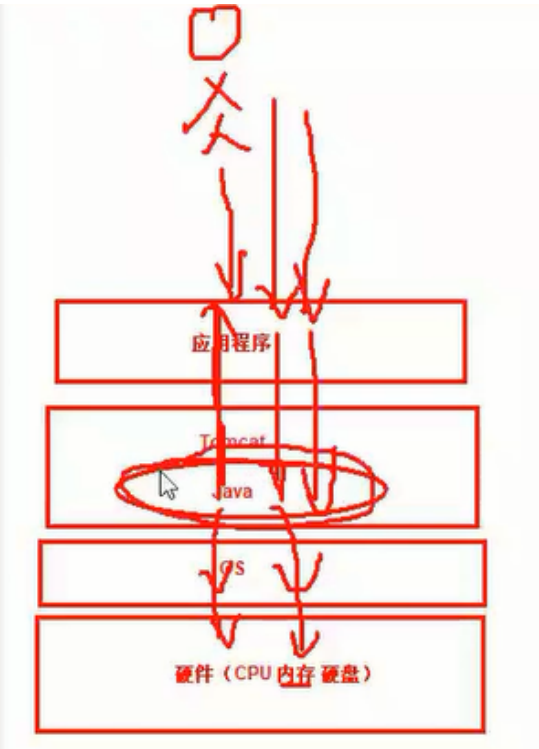
优点：更可靠

➢ 异步 ：

委托一小弟拿银行卡到银行取钱，然后给你（使用异步IO时，Java将IO读写委托给OS处理，需要将数据缓冲区地址和大小传给OS(银行卡和密码)，OS需要支持异步IO操作API）

异步：异步操作就是Java将IO读写交给OS系统处理

优点：效率更高




➢阻塞 ：

ATM排队取款，你只能等待（使用阻塞IO时，Java调用会一直阻塞到读写完成才返回）

阻塞：死脑筋；使用这个时候，Java调用会一直等到读写完成才返回，慢的时候回很慢。

➢ 非阻塞 ：（更灵活）

柜台取款，取个号，然后坐在椅子上做其它事，等号广播会通知你办理，没到号你就不能去，你可以不断问大堂经理排到了没有，大堂经理如果说还没到你就不能去

使用非阻塞IO时，如果不能读写Java调用会马上返回，当IO事件分发器会通知可读写时再继续进行读写，不断循环直到读写完成

## 2、Java对BIO NIO AIO的支持

➢JavaBIO

同步并阻塞

服务器实现模式为一个连接一个线程，即客户端有连接请求时服务器端就需要启动一个线程进行处理，

如果这个连接不做任何事情会造成不必要的线程开销，当然可以通过线程池机制改善

➢Java NIO （默认设置）

同步非阻塞

服务器实现模式为一个请求一个线程，即客户端发送的连接请求都会注册到多路复用器上，多路复用器轮询到连接有I/O请求时才启动一个线程进行处理

➢JavaAIO (NIO.2)

异步非阻塞

服务器实现模式为一个有效请求一个线程，客户端的I/O请求都是由OS先完成了再通知服务器应用去启动线程进行处理的I/O请求都是由OS先完成了再通知服务器应用去启动线程进行处理

## 3、BIO NIO AIO适用场景分析

➢BIO

BIO方式适用于连接数目比较小且固定的架构，

这种方式对服务器资源要求比较高，并发局限于应用中，

JDK1.4以前的唯一选择， 但程序直观简单易理解，

性能非常低下，没有经过任何优化处理和支持

➢NIO

Nio(new I/O),是java SE1.4及后续版本提供的一种新的I/O操作方式(即java.nio包及其子包)

Javanio是一个基于缓冲区，

并能提供非阻塞I/O操作的javaAPI,因此nio也被看成是non-blockingI/O的缩写

它拥有比传统I/O操作(bio)更好的并发运行性能

适用于连接数目多且连接比较短(轻操作)的架构，比如聊天服务器，并发局限于应用中

➢APR

安装起来最困难，

但是从操作系统级别来解决异步的IO问题，大幅度的提高了性能

tomcat bio、nio、apr 软件测试版本（10分钟压测结果）

tomcat默认首页

| tomcat 模式 | 并发数 | 样本数 | 平均响应时间(ms) | 吞吐量(s) | 偏移 | 错误率(%) | KB/sec |
|---|---|---|---|---|---|---|---|
| bio | 200 | 620140 | 114 | 1033 | 194 | 0 | 11479 |
| bio | 300 | 596220 | 174 | 990 | 505 | 0.01 | 10997 |
| bio | 500 | 460704 | 645 | 764 | 1994 | 0.65 | 8443 |
| bio | 700 | 440252 | 947 | 730 | 2914 | 1.65 | 8002 |
| bio | 1000 | 255481 | 2350 | 422 | 5020 | 5.87 | 4471 |
| bio | 1300 | 185352 | 4227 | 305 | 6767 | 12.56 | 3049 |
| bio | 1500 | 253813 | 3583 | 413 | 6565 | 11.23 | 4173 |
| | | | | | | | |
| nio | 200 | 840157 | 121 | 1396 | 68 | 0 | 15512 |
| nio | 300 | 805874 | 60 | 1340 | 145 | 0 | 14889 |
| nio | 500 | 817107 | 235 | 1354 | 205 | 0.01 | 15041 |
| nio | 700 | 817243 | 265 | 1356 | 283 | 0 | 15062 |
| nio | 1000 | 736392 | 262 | 1218 | 505 | 0 | 13526 |
| nio | 1300 | 738334 | 250 | 1217 | 567 | 0 | 13519 |
| nio | 1500 | 726248 | 252 | 1200 | 663 | 0 | 13330 |
| | | | | | | | |
| apr | 200 | 882166 | 94 | 1463 | 73 | 0 | 16247 |
| apr | 300 | 711858 | 139 | 1175 | 268 | 0 | 13056 |
| apr | 500 | 701720 | 155 | 1158 | 410 | 0.03 | 12862 |
| apr | 700 | 896486 | 127 | 1482 | 297 | 0 | 16465 |
| apr | 1000 | 709752 | 207 | 1169 | 689 | 0 | 12981 |
| apr | 1300 | 629729 | 385 | 1037 | 1145 | 0 | 11518 |
| apr | 1500 | 627767 | 390 | 1028 | 1253 | 0.01 | 11419 |

bio已淘汰

nio:（默认）

并发：

200~400

选择nio

并发：

500~1000

选择apr

Tomcat8默认运行模式为NIO

```
[root@localhost ~]# tail -1 /usr/local/tomcat8/logs/catalina.out
09-Apr-2020 19:28:51.753 信息 [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["ajp-nio-8009"]
[root@localhost ~]# vim /usr/local/tomcat8/conf/server.xml
```

```
    <Connector port="8080" protocol="org.apache.coyote.http11.Http11NioProtocol"
    connectionTimeout="20000"
    redirectPort="8443"
    compression="on"
    compressionMinSize="50"
    noCompressionUserAgents=" goilla, traviata"
    compressableMimeType="text/html,text/xml,text/javascript,texss,text/plain" />“
[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh
[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh
[root@localhost ~]# tail -f /usr/local/tomcat8/logs/catalina.out
08-Apr-2019 11:56:02.441信息[main] org.apache.coyote.AbstractProtocol.start Starting
ProtocolHandler ["http-nio-8080"]
```

## 模式三　Apr:

```
[root@localhost ~]# rpm -e apr --nodeps
[root@localhost ~]# yum y install apr apr-devel
[root@localhost ~]# cp /usr/local/tomcat8/bin/tomcat-native.tar.gz /root/
[root@localhost ~]# tar xf tomcat-native.tar.gz
[root@localhost ~]# cd tomcat-native-1.2.21-src/native/
[root@localhost native]#./configure     -with-apr=/usr/bin/apr-1-config --with-java-
home=/usr/local/java && make && make install
```

安装好的界面如下

```
----------------------------------------------------------------------
Libraries have been installed in:
   /usr/local/apr/lib

If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the `-LLIBDIR'
flag during linking and do at least one of the following:
   - add LIBDIR to the `LD_LIBRARY_PATH' environment variable
     during execution
   - add LIBDIR to the `LD_RUN_PATH' environment variable
     during linking
   - use the `-Wl,-rpath -Wl,LIBDIR' linker flag
   - have your system administrator add LIBDIR to `/etc/ld.so.conf'

See any operating system documentation about shared libraries for
```

```
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------
```

修改路径：

```
[root@localhost tomcat-native-1.2.17-src]# cd
[root@localhost native]# vim /usr/local/tomcat8/bin/catalina.sh
   #末尾(开头也可以)添加， 设置环境变量
    CATALINA_OPTS="-Djava.library.path=/usr/local/apr/lib"     # 异步，非阻塞
[root@localhost native]#vim     /usr/local/tomcat8/conf/server.xml
    69  <Connector port="8080" protocol="org.apache.coyote.http11.Http11AprProtocol"
[root@localhost ~]# vim /etc/profile     ##在最后一行添加
     81 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/apr/lib
[root@localhost native]# /usr/local/tomcat8/bin/shutdown.sh
[root@localhost native]# /usr/local/tomcat8/bin/startup.sh
[root@localhost ~]# netstat -lnpt | grep java
tcp6       0      0 :::45597                :::*                    LISTEN
3338/java
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN
3338/java
tcp6       0      0 :::8009                 :::*                    LISTEN
3338/java
tcp6       0      0 :::8080                 :::*                    LISTEN
3338/java
tcp6       0      0 :::33624                :::*                    LISTEN
3338/java
tcp6       0      0 :::12345                :::*                    LISTEN
3338/java
[root@localhost native]# tail -f /usr/local/tomcat8/logs/catalina.out
08-Apr-2019 12:20:20.455 信息 [main] org.apache.coyote.AbstractProtocol.start Starting
ProtocolHandler ["ajp-nio-8009"]
```

在实际生产环境中，Tomcat中跑的项目代码特别大，这时Tomcat启动会非常慢，
Tomcat在启动时，会有生成随机号的机制，
解决重启tomcat服务后，8005端口延迟启动的问题

```
[root@localhost ~]# vim /usr/local/java/jre/lib/security/java.security
117 securerand om.source=file:/ dev/urandom        #将原来的random改为urandom
```