

基于java的web平台构建-tomcat

By: Ivy cui
Time: 2015.12.09
Mail:cuihuahua@uplooking.com

版权声明:

本文遵循“署名-非商业性使用-相同方式共享 2.5 中国大陆”协议

您可以自由复制、发行、展览、表演、放映、广播或通过信息网络传播本作品

您可以根据本作品演绎自己的作品

您必须按照作者或者许可人指定的方式对作品进行署名。

您不得将本作品用于商业目的。

如果您改变、转换本作品或者以本作品为基础进行创作，您只能采用与本协议相同的许可协议发布基于本作品的演绎作品。

对任何再使用或者发行，您都必须向他人清楚地展示本作品使用的许可协议条款。

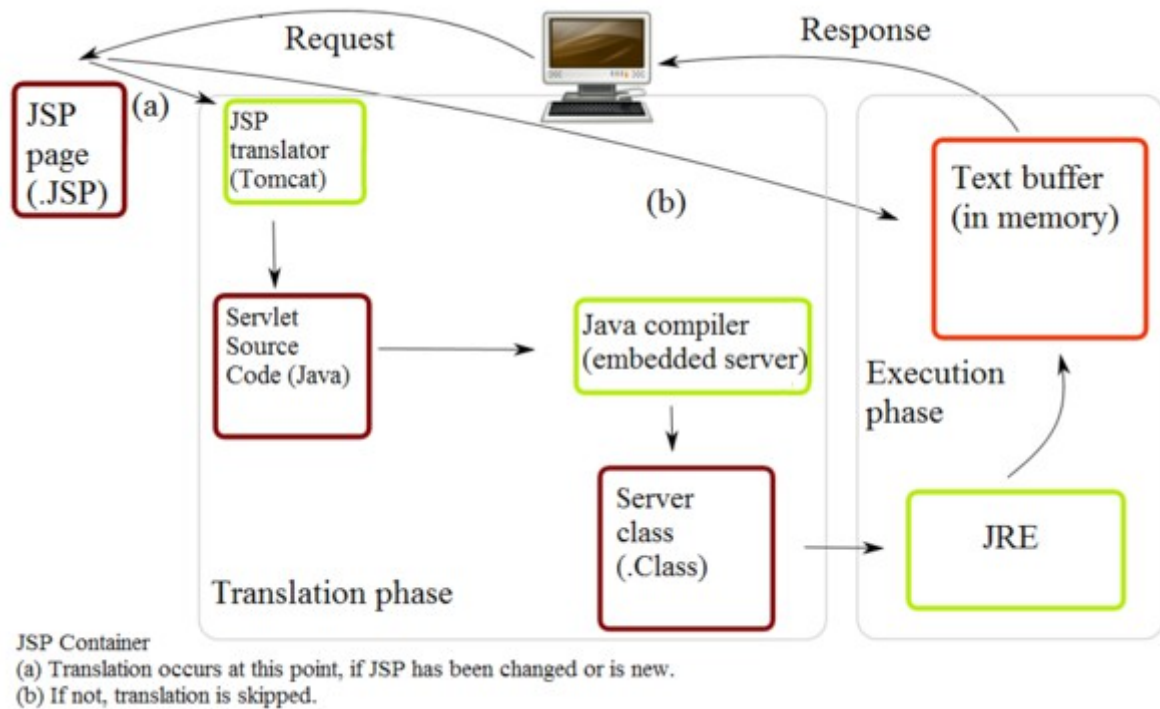
如果得到著作权人的许可，您可以不受任何这些条件的限制。

第 1 章 tomcat 简介.....	3
第 1.1 节 tomcat 简介.....	3
第 1.1 节 获取 tomcat.....	3
第 2 章 tomcat 的安装和启动.....	4
第 2.1 节 tomcat 服务器基础搭建操作步骤.....	4
第 2.2 节 优化 tomcat 服务启动.....	5
第 2.3 节 tomcat 简单操作.....	8
第 3 章 tomcat 虚拟主机配置.....	10
第 3.1 节 虚拟主机基本配置.....	10
第 3.1 节 nginx 代理访问 tomcat 服务.....	11
第 4 章 Tomcat+db.....	13
第 4.1 节 cgi 连接 db 过程.....	13
第 4.1 节 配置过程.....	13

第 5 章 Tomcat 程序复制.....	15
第 5.1 节 程序复制过程.....	15
第 5.1 节 session 问题.....	16
第 6 章 Tomcat 多实例.....	18
第 3.1 节 tomcat 多实例配置.....	18
第 3.3 节 tomcat 多实例优化.....	20

第1章 tomcat 简介

第 1.1 节 tomcat 简介



Tomcat 不能独立运行，因为真正用来解释 java 页面的是 jvm 程序，由 tomcat 来调用 jvm。

第 1.1 节 获取 tomcat

最初被发布出来的版本是 Tomcat 3.0.x，当前的最新开发版本是 9.0.课程环境中使用 8.0 版本为例。

tomcat 官网: <http://tomcat.apache.org/>

第2章 tomcat 的安装和启动

第 2.1 节 tomcat 服务器基础搭建操作步骤

将 serverc 作为 tomcat 服务器

(1) 下载并安装 jdk 软件包

```
[root@serverc ~]# mount 172.25.254.250:/content /mnt/
[root@serverc ~]# cd /mnt/item/tomcat/
[root@serverc tomcat]# rpm -ivh jdk-7u79-linux-x64.rpm
```

(2) 添加 tomcat 用户。tomcat 为系统用户，为维护 tomcat 服务正常运行而存在。故 UID 最好指定为 1-1000 之间，并且登录 shell 为 nologin。

```
[root@serverc tomcat]# groupadd -g 666 tomcat
[root@serverc tomcat]# useradd -u 666 -g tomcat tomcat
[root@serverc tomcat]# id tomcat
uid=666(tomcat) gid=666(tomcat) groups=666(tomcat)
```

(3) 下载安装 tomcat 软件包。(因为版本问题，tomcat 没有 rpm 包) 该 Tar 包解压时推荐指定解压目录为 tomcat 家目录。因为该目录只有 tomcat 用户有完整权限，其他用户无任何权限，可增加安全性。(一般第三程序安装在 /usr/local/ 目录或者 /home/\$user/ 目录，因为 /usr/local/ 下程序太多，且任何人都可读，故放置在 /home/\$user 目录下安全性更高。)

```
[root@serverc tomcat]# tar xf apache-tomcat-8.0.24.tar.gz -C /home/tomcat/
[root@serverc tomcat]# cd /home/tomcat/apache-tomcat-8.0.24/
[root@serverc apache-tomcat-8.0.24]# ls -l
drwxr-xr-x 2 root root 4096 Dec 11 14:22 bin #存放命令
drwxr-xr-x 2 root root 4096 Jul 2 04:23 conf #存放配置文件
drwxr-xr-x 2 root root 4096 Dec 11 14:22 lib #存放库文件，java 的库文件后缀为 jar
-rw-r--r-- 1 root root 57011 Jul 2 04:23 LICENSE
drwxr-xr-x 2 root root 6 Jul 2 04:20 logs #存放相关日志
-rw-r--r-- 1 root root 1444 Jul 2 04:23 NOTICE
-rw-r--r-- 1 root root 6741 Jul 2 04:23 RELEASE-NOTES
-rw-r--r-- 1 root root 16204 Jul 2 04:23 RUNNING.txt
drwxr-xr-x 2 root root 29 Dec 11 14:22 temp #存放临时文件
drwxr-xr-x 7 root root 76 Jul 2 04:21 webapps #默认网站网页文件根目录
drwxr-xr-x 2 root root 6 Jul 2 04:20 work #工作目录
```

(4) 声明变量。JAVA_HOME：声明 jdk 安装路径。CATALINA_HOME：声明 tomcat 程序中命令和库文件所在位置。CATALINA_BASE：声明 tomcat 程序中配置文件、网站根目录等所在位置。

```
[root@serverc ~]# export JAVA_HOME="/usr/java/jdk1.7.0_79/"
[root@serverc ~]# export CATALINA_HOME="/home/tomcat/apache-tomcat-8.0.24/"
[root@serverc ~]# export CATALINA_BASE="/home/tomcat/apache-tomcat-8.0.24/"
```

(5) bin 目录下有服务的状态控制脚本。启动 tomcat 服务的脚本名称为 startup.sh。执行该脚本可启动

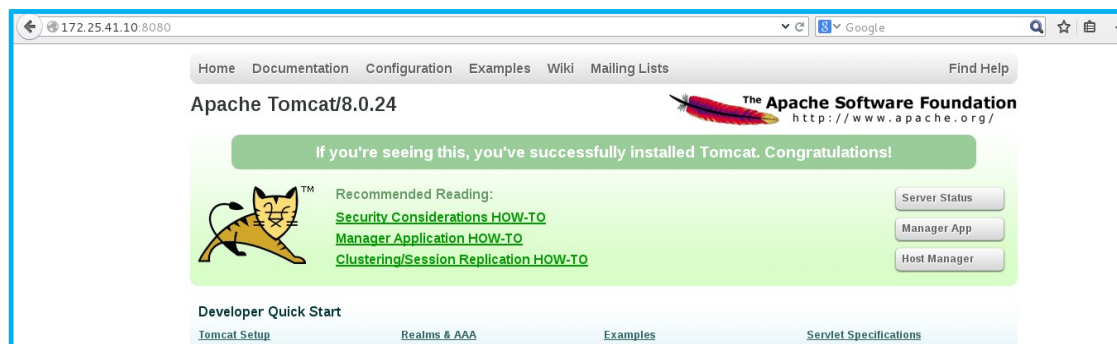
tomcat 服务。该服务默认监听 tcp/8080 端口。

```
[root@serverc ~]# cd /home/tomcat/apache-tomcat-8.0.24/bin/
[root@serverc bin]# ./startup.sh
[root@serverc bin]# ps -ef | grep java
[root@serverc bin]# netstat -ltunp | grep 8080
tcp6      0      0 :::8080          :::*              LISTEN     1102/java
```

(6) 在 servera 机器上添加防火墙规则。将目标地址为 servera 机器 8080 端口的访问转给 serverc 内网地址的 8080 端口。

```
[root@servera ~]# iptables -t nat -A PREROUTING -d 172.25.41.10 -i eth0 -p tcp --dport 8080 -j DNAT --to-destination 192.168.0.12:8080
[root@servera ~]# iptables-save > /etc/sysconfig/iptables
```

(7) 在 client 端 workstation 机器上访问 tomcat 服务器。注意端口需指定为 8080 端口。出现的页面为 tomcat 的默认首页。



第 2.2 节 优化 **tomcat** 服务启动

上述启动方法存在以下问题 (1) 启动和关闭 tomcat 服务需进入到 tomcat 主程序中 bin 目录找 startup.sh 和 shutdown.sh 脚本, 不方便。(2) tomcat 进程是由 root 用户打开并维护的, 从安全角度考虑存在缺陷。

如果希望可以通过 service 命令等方式方便的启动, 并且希望是 tomcat 用户维护 tomcat 服务正常运行, 需安装 jsvc 命令。

(1) 进入 tomcat 主程序所在目录的子目录 bin, 该目录下有 common-daemon-native 软件包, 通过该软件安装 jsvc 命令。该软件需通过源代码安装方式安装。

1.1 解压该源代码压缩包至当前目录, 进入解压目录中子目录 unix 目录, 执行 configure 检测程序。

```
[root@serverc bin]# cd /home/tomcat/apache-tomcat-8.0.24/bin/
[root@serverc bin]# tar -xf commons-daemon-native.tar.gz
[root@serverc bin]# cd commons-daemon-1.0.15-native-src/unix/
[root@serverc unix]# ./configure
```

1.2 检测过程中可能会出现很多问题, 需逐一解决。例如: 下图中由于没有 c 语言编译器导致检测未完成, 出现 error。可通过 yum 程序安装 gcc 软件包解决。

```
[root@serverc unix]# ./configure
*** Current host ***
```

```
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking cached host system type... ok
*** C-Language compilation tools ***
checking for gcc... no
checking for cc... no
checking for cc... no
checking for cl... no
configure: error: no acceptable C compiler found in $PATH
```

See `config.log' for more details.

```
[root@serverc unix]# yum -y install gcc
```

1.3 修改上述问题后重新检测，检测已通过。但是中途出现 warning，出现 warning 不会中断检测程序，故可以忽略。如果想解决，可查看报错，找除解决方法。例如：下图中由于没有 libcap 相关程序导致警告。

```
[root@serverc unix]# ./configure
*** C-Language compilation tools ***
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking for sys/capability.h... no
configure: WARNING: cannot find headers for libcap
*** Writing output files ***
configure: creating ./config.status
config.status: creating Makefile
config.status: creating Makedefs
config.status: creating native/Makefile
*** All done ***
Now you can issue "make"
```

1.4 安装 libcap-devel 软件包解决该问题。一般在源代码安装软件过程中，需安装的大多为 devel 包，即程序的开发包。

```
[root@serverc unix]# yum -y install libcap-devel
```

1.5 解决该报警之后，再次执行 configure 检测程序完成检测。检测通过后会生成 Makefile 文件以记录检测结果。（此处省略）

1.6 执行 make 程序编译，将源代码编译成二进制。编译过程中会调用 Makefile 文件。

```
[root@serverc unix]# make
(cd native; make all)
make[1]: Entering directory `/home/tomcat/apache-tomcat-8.0.24/bin/commons-daemon-1.0.15-native-src/unix/native'
gcc -g -O2 -DOS_LINUX -DDSO_DLFCN -DCPU=\"amd64\" -Wall -Wstrict-prototypes -DHAVE_LIBCAP -I/usr/java/jdk1.7.0_79//include -I/usr/java/jdk1.7.0_79//include/linux -c jsvc-unix.c -o jsvc-unix.o
```

1.7 编译完成后，该目录下会安装 jsvc 命令，将该命令拷贝至 /home/tomcat/apache-tomcatxxx/bin 目录下。因为该目录下有 daemon.sh 文件，执行该文件时会自动调用 jsvc 命令。

```
[root@serverc unix]# cp jsvc /home/tomcat/apache-tomcat-8.0.24/bin/
[root@serverc unix]# cd /home/tomcat/apache-tomcat-8.0.24/bin/
[root@serverc bin]# grep jsvc daemon.sh
# Setup parameters for running the jsvc
# If not explicitly set, look for jsvc in CATALINA_BASE first then CATALINA_HOME
JSVC="$CATALINA_BASE/bin/jsvc"
JSVC="$CATALINA_HOME/bin/jsvc"
```

1.8 编辑 daemon.sh 文件，将之前临时声明的 JAVA_HOME、CATALINA_HOME、CATALINA_BASE 变量再该文件中声明，以便永久生效，并且可以在启动 tomcat 服务时自动声明。

```
[root@serverc ~]# vim /home/tomcat/apache-tomcat-8.0.24/bin/daemon.sh
```

```
# resolve links - $0 may be a softlink

export JAVA_HOME="/usr/java/jdk1.7.0_79/"
export CATALINA_HOME="/home/tomcat/apache-tomcat-8.0.24/"
export CATALINA_BASE="/home/tomcat/apache-tomcat-8.0.24/"

ARG0="$0"
```

(2) 将文件复制到 /etc/init.d/ 目录下，以后就可以方便使用 rhel6 版本中服务的状态控制方式控制 tomcat 服务。

前一章节 tomcat 服务器基础搭建操作步骤中我们启动过 tomcat 服务，该进程已经存在，所以在重启之前先将 tomcat 进程停掉。

再通过 /etc/init.d/tomcat start 方式或者 service tomcat start 方式启动该服务。

对比发现，通过这种方式启动之后，tomcat 进程的拥有者为 tomcat 用户。（当前，root 用户需要先打开 tomcat 进程，然后再由 tomcat 用户创建子进程。root 用户打开的 tomcat 进程不参与服务运行。）

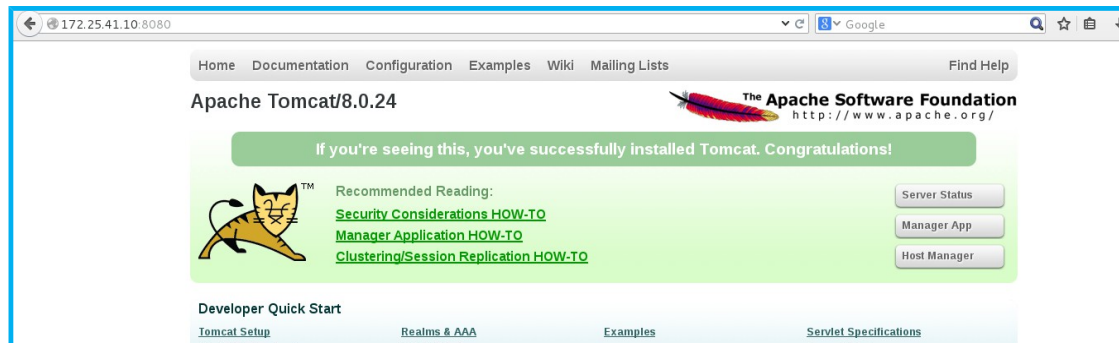
```
[root@serverc ~]# cp /home/tomcat/apache-tomcat-8.0.24/bin/daemon.sh /etc/init.d/tomcat
[root@serverc ~]# ps -ef | grep java
root          1102          1    0  14:24  ?                00:00:06 /usr/java/jdk1.7.0_79/bin/java
-Djava.util.logging.config.file=/home/tomcat/apache-tomcat-8.0.24/conf/logging.properties
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Djava.endor ...
[root@serverc ~]# kill 1102
[root@serverc ~]# ps -ef | grep java
root    4146  3979  0 14:40 pts/1    00:00:00 grep --color=auto java
```

```
[root@serverc ~]# /etc/init.d/tomcat start
[root@serverc ~]# ps -ef | grep java
root    4157    1  0 14:40 ?        00:00:00 jsvc.exec -java-home /usr/java/jdk1.7.0_79/ -user tomcat -pidfile
/home/tomcat/apache-tomcat-8.0.24/logs/catalina-daemon.pid -wait 10 -outfile ...
tomcat  4158  4157  58 14:40 ?        00:00:02 jsvc.exec -java-home /usr/java/jdk1.7.0_79/ -user tomcat -pidfile
/home/tomcat/apache-tomcat-8.0.24/logs/catalina-daemon.pid -wait 10 -outfile /home/tomcat/apache-tomcat-
8.0.24/logs/catalina-daemon.out -errfile &1 -classpath /ho...
```

(3) 在 client 端 workstation 机器上访问 tomcat 服务器。访问后发现出现空白页（图省略）。此处是由于 tomcat 主程序所在目录下 UGO 权限导致的，该目录需要 tomcat 用户有完整权限。可以通过改变目录

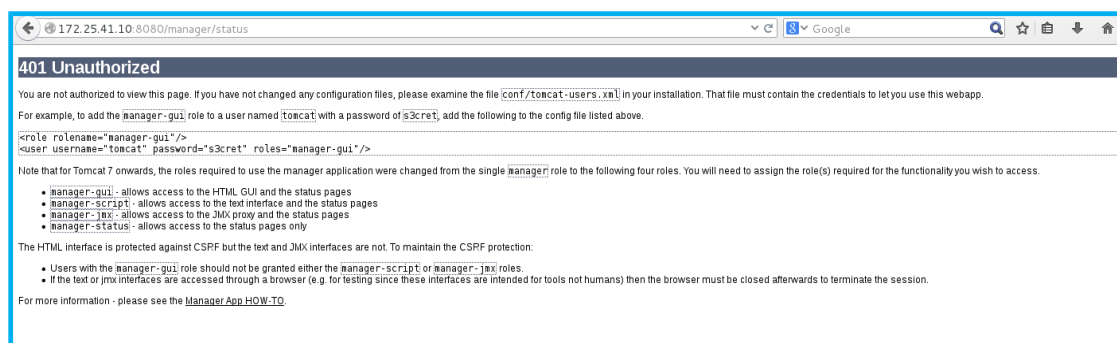
拥有者方式达到此要求。重启 tomcat 服务后，再次测试，即看到 tomcat 默认首页文件。

```
[root@serverc ~]# cd /home/tomcat/
[root@serverc tomcat]# chown tomcat * -R
[root@serverc tomcat]# /etc/init.d/tomcat stop
[root@serverc tomcat]# /etc/init.d/tomcat start
```



第 2.3 节 tomcat 简单操作

(1) 在 client 端点击 tomcat 首页上 server status 图标，会提示输入用户名和密码方可看到内容。如果选择不输入或输错了，即弹出下图报错。报错信息中给出了解决方法。



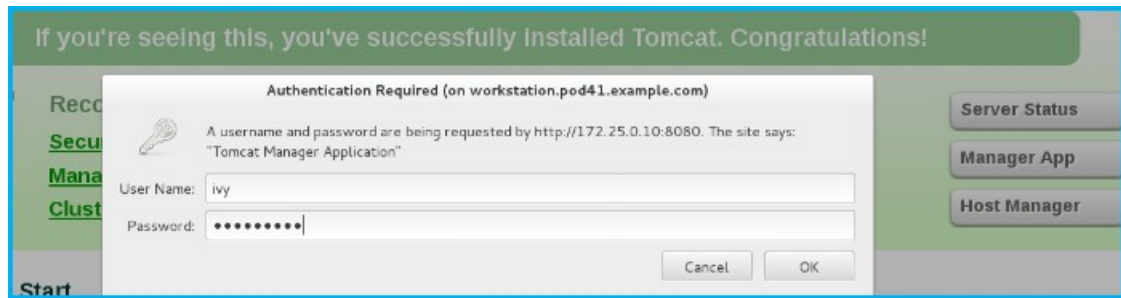
(2) 按照报错信息提示，修改 tomcat 主程序所在目录下子目录 conf 中 tomcat-users.xml 文件,在 </tomcat-users> 结束符前做添加。

```
<role rolename="manager-gui"/>
<user username="ivy" password="uplooking" roles="manager-gui"/>
</tomcat-users>
```



(3) 修改完上述文件后，重启 tomcat 服务。

```
[root@serverc ~]# /etc/init.d/tomcat stop
[root@serverc ~]# /etc/init.d/tomcat start
```

(4) 再次点击首页上 servers status 按钮，输入自己指定的用户名和密码，即可看到服务器状态。



172.25.41.10:8080/manager/status Google



Server Status

Manager

[List Applications](#) [HTML Manager Help](#) [Manager Help](#) [Complete Server Status](#)

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture	Hostname	IP Address
Apache Tomcat/8.0.24	1.7.0_79-b15	Oracle Corporation	Linux	3.10.0-229.el7.x86_64	amd64	serverc.pod41.example.com	192.168.0.12

JVM

Free memory: 6.62 MB Total memory: 30.75 MB Max memory: 121.81 MB

Memory Pool	Type	Initial	Total	Maximum	Used
Eden Space	Heap memory	2.06 MB	8.50 MB	33.62 MB	3.38 MB (10%)
Survivor Space	Heap memory	0.25 MB	1.06 MB	4.18 MB	1.06 MB (25%)
Tenured Gen	Heap memory	5.12 MB	21.18 MB	84.00 MB	19.67 MB (23%)
Code Cache	Non-heap memory	2.43 MB	2.43 MB	48.00 MB	1.40 MB (2%)
Perm Gen	Non-heap memory	20.75 MB	22.62 MB	82.00 MB	22.40 MB (27%)

"ajp-nio-8009"

Max threads: 200 Current thread count: 0 Current thread busy: 0 Kept alive sockets count: 0

第3章 tomcat 虚拟主机配置

第 3.1 节 虚拟主机基本配置

(1) 修改 tomcat 服务的主配置文件。该文件位于 tomcat 主程序下 conf 目录中，文件名称为 server.xml。

```
[root@serverc ~]# cd /home/tomcat/apache-tomcat-8.0.24/conf/
[root@serverc conf]# ls -l server.xml
-rw----- 1 tomcat root 6458 Jul  2 04:23 server.xml
```

(2) 编辑该文件，添加虚拟主机。host 字段为 tomcat 服务的虚拟主机配置字段。下图中配置了两个虚拟主机，www.tomcat1.com 和 www.tomcat2.com，两台虚拟主机有自己的 appbase（即网页文件根目录，下图中使用的是相对路径表示虚拟主机网页根目录，该路径是相对于 tomcat 服务主程序所在目录来说，该路径现不存在，后续再创建）和相关日志前缀。注意：严格遵循该文件的语法格式，每个虚拟主机的配置都是<host>开始，</host>结束。

```
<Host name="www.tomcat1.com" appBase="tomcat1.com"
    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>
<Host name="www.tomcat2.com" appBase="tomcat2.com"
    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>
```

(3) 进入 tomcat 服务主程序所在目录，创建上述步骤中虚拟主机所指定的 appBase，分别进入每一个虚拟主机的 appBase 目录下创建 ROOT 目录，在 ROOT 目录写每一个虚拟主机对应的首页文件。。

```
[root@serverc conf]# cd /home/tomcat/apache-tomcat-8.0.24/
[root@serverc apache-tomcat-8.0.24]# mkdir tomcat1.com/
[root@serverc apache-tomcat-8.0.24]# cd tomcat1.com/
[root@serverc tomcat1.com]# mkdir ROOT
[root@serverc tomcat1.com]# echo tomcat1 > ROOT/index.html
[root@serverc tomcat1.com]# cd ../
[root@serverc apache-tomcat-8.0.24]# mkdir tomcat2.com
[root@serverc apache-tomcat-8.0.24]# cd tomcat2.com
[root@serverc tomcat2.com]# mkdir ROOT
[root@serverc tomcat2.com]# echo tomcat2 > ROOT/index.html
```

(4) 重启 tomcat 服务

```
[root@serverc tomcat2.com]# /etc/init.d/tomcat stop
```

```
[root@serverc tomcat2.com]# /etc/init.d/tomcat start
```

(5) 在 client 端 workstation 机器上测试

```
[root@workstation ~]# echo 172.25.41.10 www.tomcat1.com >> /etc/hosts
```

```
[root@workstation ~]# echo 172.25.41.10 www.tomcat2.com >> /etc/hosts
```



tomcat1



tomcat2

第 3.1 节 *nginx* 代理访问 *tomcat* 服务

通过上述配置，我们会发现，在客户端访问过程中，需要在\$host后添加上需要访问的端口号，略麻烦，我们希望客户端访问的都是 http 协议默认的 80 端口，但是可以访问到后台多台服务器。这个时候我们可以引入 *nginx*，让 *nginx* 作为代理服务器去代理用户请求。也就是说客户端找的是 *nginx* 服务，*nginx* 服务默认监听 80 端口，然后 *nginx* 将用户请求转交给后台监听不同端口的 *tomcat* 虚拟主机。

```
[root@serverb ~]# cd /etc/nginx/conf.d/
[root@serverb conf.d]# cp default.conf www.tomcat.com.conf
[root@serverb conf.d]# vim /etc/nginx/conf.d/www.tomcat.com.conf
server {
    listen    80;
    server_name ~www\.tomcat.*\.com;

    #charset koi8-r;
```

```
#access_log /var/log/nginx/log/host.access.log main;
```

```
location / {
```

```
    index index.html index.htm;
```

```
    proxy_pass http://172.25.0.12;
```

```
    proxy_set_header Host $host;
```

```
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
    proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504 http_404;
```

```
    proxy_set_header X-Real-IP $remote_addr;
```

```
    proxy_redirect off;
```

```
}
```

第4章 Tomcat+db

第 4.1 节 *cgi* 连接 *db* 过程

- 有关数据库服务器信息写到文件中（php 程序需要写到 php 代码文件中，jsp 可以写到代码文件中，也可以写到配置文件中）：数据库服务器的 ip 地址，主机名、用户名、密码、库名
- 驱动：php+db 使用 php-mysql 驱动，tomcat+db 使用 java-connector-java 驱动
- db 授权：允许 cgi 所在服务器读取 db 中信息

第 4.1 节 配置过程

- (1) 进入服务器公共目录，下载 ejforum 软件包

```
[root@serverc ~]# cd /mnt/items/tomcat/
[root@serverc tomcat]# cp ejforum-2.3.zip /opt/
```

- (2) 解压 ejforum 软件包，并将 jsp 文件拷贝到网站根目录下。

```
[root@serverc opt]# unzip ejforum-2.3.zip
[root@serverc opt]# cd ejforum-2.3/
[root@serverc ejforum-2.3]# cp -r ejforum/* /home/tomcat/apache-tomcat-8.0.24/tomcat1.com/ROOT/
```

- (3) 设置 UGO 权限，使 tomcat 用户对网站根目录下文件具有完整权限。

```
[root@serverc ejforum-2.3]# chown -R tomcat /home/tomcat/apache-tomcat-8.0.24/tomcat1.com/ROOT/
```

- (4) 安装驱动。mysql-connector-java 程序解压后，把 mysql-connector-java-5.1.36-bin.jar 文件拷贝到 tomcat 安装主目录下的 lib 目录。

```
[root@serverc ~]# cd /mnt/items/tomcat/
[root@serverc tomcat]# cp mysql-connector-java-5.1.36.tar.gz /opt/
[root@serverc opt]# cd mysql-connector-java-5.1.36/
[root@serverc mysql-connector-java-5.1.36]# cp mysql-connector-java-5.1.36-bin.jar /home/tomcat/apache-tomcat-8.0.24/lib/
```

- (5) 配置数据库相关信息

```
[root@serverc ~]# cd /home/tomcat/apache-tomcat-8.0.24/tomcat1.com/ROOT/WEB-INF/
[root@serverc WEB-INF]# vim conf/config.xml

<database maxActive="10" maxIdle="10" minIdle="2" maxWait="10000"
    username="javaadmin" password="uplooking"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://172.25.0.18:3306/javabbs?
characterEncoding=gbk&autoReconnect=true&autoReconnectForPools=true&zeroDateTimeBehavior=convertToNull"
    sqlAdapter="sql.MySqlAdapter"/>
```

- (6) 将 ejforum 软件包中 easyjforum_mysql 中的数据导入到数据库服务器 server1 的 javabbs 库中。

```
[root@serveri ~]#mysqladmin -u root password "uplooking"
[root@serveri ~]# mysqladmin -uroot -puplooking create javabbs
[root@serveri ~]# mysql -uroot -puplooking javabbs < /tmp/install/script/easyjforum_mysql.sql
```

(7) 在数据库服务器上给 tomcat 服务器授权

```
[root@serveri ~]# echo "grant all on javabbs.* to javaadmin@'172.25.0.12' identified by 'uplooking';" | mysql -uroot -puplooking
[root@serveri ~]# mysqladmin -uroot -puplooking flush-privileges
```

(8) 重启 tomcat 服务

```
[root@serverc ~]# /etc/init.d/tomcat stop
[root@serverc ~]# /etc/init.d/tomcat start
```

(9) 客户端访问 tomcat 服务器，可以看到 ejforum 论坛。（图省略）

第5章 Tomcat 程序复制

第 5.1 节 程序复制过程

- (1) 在新的 tomcat 服务器上安装 jdk 程序

```
[root@servere ~]# cd /mnt/items/tomcat/  
[root@servere tomcat]# rpm -ivh jdk-7u79-linux-x64.rpm
```

- (2) 添加 tomcat 用户

```
[root@servere tomcat]# groupadd -g 666 tomcat  
[root@servere tomcat]# useradd -u 666 -g tomcat tomcat
```

- (3) 将 serverc 机器上 tomcat 安装目录下的所有文件和程序启动控制脚本复制到 servere 机器

```
[root@serverc ~]# tar cf /tmp/data.tar /home/tomcat/apache-tomcat-8.0.24/  
[root@serverc ~]# scp /tmp/data.tar 172.25.0.14:/tmp/  
[root@serverc ~]# scp /etc/init.d/tomcat 172.25.0.14:/etc/init.d/  
[root@servere tomcat]# tar xf /tmp/data.tar -C /
```

- (4) 启动 servere 的 tomcat 服务

```
[root@servere ~]# /etc/init.d/tomcat start
```

- (5) nginx 做代理，将用户请求转交给 serverc 和 servere 机器。

```
[root@serverb ~]# vim /etc/nginx/nginx.conf  
upstream java_pools {  
    server 172.25.0.12:8080;  
    server 172.25.0.14:8080;  
}
```

```
[root@serverb ~]#vim /etc/nginx/conf.d/www.tomcat.com.conf  
location / {  
    index index.html index.htm;  
    proxy_pass http://java_pools;  
  
    proxy_set_header Host $host;  
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504 http_404;  
    proxy_set_header X-Real-IP $remote_addr;  
    proxy_redirect off;  
  
}
```

第 5.1 节 session 问题

多个 tomcat 要一起协同工作有几种办法，可以考虑的方案有以下几个：

1. 使用 tomcat 自带的 cluster 方式，多个 tomcat 见自动实时复制 session 信息，配置起来很简单。但这个方案的效率比较低，在大并发下表现并不好。
2. 利用 nginx 的基于访问 ip 的 hash 路由策略，保证访问的 ip 始终被路由到同一个 tomcat 上，这个配置更简单。但是我们的应用很可能是某一个局域网大量用户同时登录，这样负载均衡就没什么作用了。
3. 利用 memcached 或 redis 等把多个 tomcat 的 session 集中管理，这是最直接的解决方案，但是操作起来也最为复杂。

我们的系统既要求性能，又要比较好的利用上负载均衡，所以第 3 个方案是首选。

memcached 或 redis 比较

如果简单地比较 Redis 与 Memcached 的区别，大多数都会得到以下观点：

- 1 Redis 不仅仅支持简单的 k/v 类型的数据，同时还提供 list, set, hash 等数据结构的存储。
 - 2 Redis 支持数据的备份，即 master-slave 模式的数据备份。
 - 3 Redis 支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用。
- 在 Redis 中，并不是所有的数据都一直存储在内存中的。这是和 Memcached 相比一个最大的区别（我个人是这么认为的）。

Redis 只会缓存所有的 key 的信息，如果 Redis 发现内存的使用量超过了某一个阈值，将触发 swap 的操作，Redis 根据 “swappiness = age*log(size_in_memory)” 计算出哪些 key 对应的 value 需要 swap 到磁盘。然后再将这些 key 对应的 value 持久化到磁盘中，同时在内存中清除。这种特性使得 Redis 可以保持超过其机器本身内存大小的数据。当然，机器本身的内存必须要能够保持所有的 key，毕竟这些数据是不会进行 swap 操作的。

同时由于 Redis 将内存中的数据 swap 到磁盘中的时候，提供服务的主线程和进行 swap 操作的子线程会共享这部分内存，所以如果更新需要 swap 的数据，Redis 将阻塞这个操作，直到子线程完成 swap 操作后才可以进行修改。

当从 Redis 中读取数据的时候，如果读取的 key 对应的 value 不在内存中，那么 Redis 就需要从 swap 文件中加载相应数据，然后再返回给请求方。这里就存在一个 I/O 线程池的问题。在默认的情况下，Redis 会出现阻塞，即完成所有的 swap 文件加载后才会相应。这种策略在客户端的数量较小，进行批量操作的时候比较合适。但是如果将 Redis 应用在一个大型的网站应用程序中，这显然是无法满足大并发的情况的。所以 Redis 运行我们设置 I/O 线程池的大小，对需要从 swap 文件中加载相应数据的读取请求进行并发操作，减少阻塞的时间。

redis、memcache、mongoDB 对比

从以下几个维度，对 redis、memcache、mongoDB 做了对比，欢迎拍砖

1、性能

都比较高，性能对我们来说应该都不是瓶颈

总体来讲，TPS 方面 redis 和 memcache 差不多，要大于 mongodb

2、操作的便利性

memcache 数据结构单一

redis 丰富一些，数据操作方面，redis 更好一些，较少的网络 IO 次数

mongodb 支持丰富的数据表达，索引，最类似关系型数据库，支持的查询语言非常丰富

3、内存空间的大小和数据量的大小

redis 在 2.0 版本后增加了自己的 VM 特性，突破物理内存的限制；可以对 key value 设置过期时间（类似 memcache）

memcache 可以修改最大可用内存,采用 LRU 算法

mongoDB 适合大数据量的存储, 依赖操作系统 VM 做内存管理, 吃内存也比较厉害, 服务不要和别的服务在一起

4、可用性（单点问题）

对于单点问题,

redis, 依赖客户端来实现分布式读写; 主从复制时, 每次从节点重新连接主节点都要依赖整个快照, 无增量复制, 因性能和效率问题,

所以单点问题比较复杂; 不支持自动 sharding, 需要依赖程序设定一致 hash 机制。

一种替代方案是, 不用 redis 本身的复制机制, 采用自己做主动复制（多份存储）, 或者改成增量复制的方式（需要自己实现）, 一致性问题 and 性能的权衡

Memcache 本身没有数据冗余机制, 也没必要; 对于故障预防, 采用依赖成熟的 hash 或者环状的算法, 解决单点故障引起的抖动问题。

mongoDB 支持 master-slave, replicaset（内部采用 paxos 选举算法, 自动故障恢复）, auto sharding 机制, 对客户端屏蔽了故障转移和切分机制。

5、可靠性（持久化）

对于数据持久化和数据恢复,

redis 支持（快照、AOF）: 依赖快照进行持久化, aof 增强了可靠性的同时, 对性能有所影响

memcache 不支持, 通常用在缓存, 提升性能;

MongoDB 从 1.8 版本开始采用 binlog 方式支持持久化的可靠性

6、数据一致性（事务支持）

Memcache 在并发场景下, 用 cas 保证一致性

redis 事务支持比较弱, 只能保证事务中的每个操作连续执行

mongoDB 不支持事务

7、数据分析

mongoDB 内置了数据分析的功能(mapreduce), 其他不支持

8、应用场景

redis: 数据量较小的更性能操作和运算上

memcache: 用于在动态系统中减少数据库负载, 提升性能; 做缓存, 提高性能（适合读多写少, 对于数据量比较大, 可以采用 sharding）

MongoDB: 主要解决海量数据的访问效率问题

第6章 Tomcat 多实例

第 3.1 节 tomcat 多实例配置

在之前虚拟主机配置章节中，tomcat 虚拟主机配置过程中停掉一个虚拟主机的运行，会导致所有虚拟主机都停止运行，因为这些虚拟主机都在同一个进程，同一个实例下操作。所以为了让多个网站之间互不影响，我们可以通过多实例的方式来操作。

(1) 先 taomcat 服务停掉

```
[root@serverc ~]# /etc/init.d/tomcat stop
```

(2) 去到/home/tomcat 下，为每一个虚拟主机创建一个目录

```
[root@serverc ~]# cd /home/tomcat/
```

```
[root@serverc tomcat]# mkdir tomcat1 tomcat2
```

(3) 将原来的安装目录下重要的配置文件拷贝至这两个新建的目录下。(不需要所有都拷贝，有些比如许可证等没有拷贝的必要)

```
[root@serverc tomcat]# cd apache-tomcat-8.0.24/
```

```
[root@serverc apache-tomcat-8.0.24]# cp -rp logs/ temp/ tomcat1.com/ work/ webapps/ conf/ ../tomcat1/
```

```
[root@serverc apache-tomcat-8.0.24]# cp -rp logs/ temp/ tomcat2.com/ work/ webapps/ conf/ ../tomcat2/
```

(4) 将原来的安装目录下多余的文件删掉，只保留 bin、lib、temp、work 等目录就可以了。

```
[root@serverc apache-tomcat-8.0.24]# rm -rf LICENSE NOTICE RELEASE-NOTES RUNNING.txt conf logs  
tomcat1.com tomcat2.com webapps
```

```
[root@serverc apache-tomcat-8.0.24]# ls
```

```
bin lib temp work
```

(5) 第一台虚拟主机的配置信息

```
[root@serverc apache-tomcat-8.0.24]# cd ../tomcat1/
```

```
[root@serverc tomcat1]# vim conf/server.xml
```

```
<Host name="www.tomcat1.com" appBase="tomcat1.com"  
    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>
```

(6) 第二台虚拟主机的配置信息 (注意第二台虚拟主机需要和第一台虚拟主机监听不同端口)

```
[root@serverc tomcat1]# cd ../tomcat2/
```

```
[root@serverc tomcat2]# vim conf/server.xml
```

```
<Host name="www.tomcat2.com" appBase="tomcat2.com"  
    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>
```

```
=====
==
    <Connector port="8081" protocol="HTTP/1.1"
        connectionTimeout="20000"
        redirectPort="8443" />
=====
==
    <Connector port="8010" protocol="AJP/1.3" redirectPort="8443" />
```

(7) 创建第一台虚拟主机服务状态控制脚本，修改 CATALINA_HOME 变量（声明 tomcat 程序中命令和库文件所在位置）以及 CATALINA_BASE 变量（声明 tomcat 程序中配置文件、网站根目录等所在位置）两个变量。

```
[root@serverc tomcat2]# cd /etc/init.d/
[root@serverc init.d]# mv tomcat tomcat1
[root@serverc init.d]# vim tomcat1
export CATALINA_HOME="/home/tomcat/apache-tomcat-8.0.24/"
export CATALINA_BASE="/home/tomcat/tomcat1"
```

(8) 创建第二台虚拟主机服务状态控制脚本，同上。

```
[root@serverc init.d]# cp tomcat1 tomcat2
[root@serverc init.d]# vim tomcat2
export CATALINA_HOME="/home/tomcat/apache-tomcat-8.0.24/"
export CATALINA_BASE="/home/tomcat/tomcat2"
```

(9) 添加防火墙转发规则。

```
[root@servera ~]# iptables -t nat -A PREROUTING -d 172.25.41.10 -i eth0 -p tcp --dport 8081 -j DNAT --to-destination 192.168.0.12:8081
[root@servera ~]# iptables-save > /etc/sysconfig/iptables
```

(10) 启动两台虚拟主机。

```
[root@serverc init.d]# /etc/init.d/tomcat1 start
[root@serverc init.d]# /etc/init.d/tomcat2 start
[root@serverc init.d]# netstat -ltunp | grep 8080
tcp6    0      0 :::8080          :::*              LISTEN      5749/jsvc.exec
[root@serverc init.d]# netstat -ltunp | grep 8081
tcp6    0      0 :::8081          :::*              LISTEN      5782/jsvc.exec
```

(11) 客户端 workstation 机器测试两台虚拟主机是否都可以正常访问。

← www.tomcat1.com:8080

tomcat1

← www.tomcat2.com:8081

tomcat2

第 3.3 节 *tomcat* 多实例优化

通过 3.2 的配置，我们会发现，多台虚拟主机通过多实例配置之后，每个虚拟主机都需要监听不同的端口号，那么导致在客户端访问过程中，需要在\$host 后添加上需要访问的端口号，略麻烦，我们希望客户端访问的都是 http 协议默认的 80 端口，但是可以访问到后台多台服务器。这个时候我们可以引入 nginx，让 nginx 作为代理服务器去代理用户请求。也就是说客户端找的是 nginx 服务，nginx 服务默认监听 80 端口，然后 nginx 将用户请求转交给后台监听不同端口的 tomcat 虚拟主机，

(1) 在 serverb 机器上配置虚拟主机。通过配置不同的 server_name 和 proxy_pass 配置行来代理不同的用户请求。比如如果匹配到\$host 为 www.tomcat1.com，则将请求转交给 serverc 的 8080 端口。

```
[root@serverb ~]# cd /etc/nginx/conf.d/
[root@serverb conf.d]# cp www.proxy.com.conf www.tomcat1.com.conf
```

```
[root@serverb conf.d]# vim www.tomcat1.com.conf
server {
    listen    80;
    server_name www.tomcat1.com;
    location / {
        proxy_pass http://192.168.0.12:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504 http_404;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_redirect off;
    }
}
```

(2) 如果匹配到\$host 为 www.tomcat2.com，则将请求转交给 serverc 的 8081 端口。

```
[root@serverb conf.d]# cp www.tomcat1.com.conf www.tomcat2.com.conf
[root@serverb conf.d]# vim www.tomcat2.com.conf
server {
    listen    80;
    server_name www.tomcat2.com;

    location / {
        proxy_pass http://192.168.0.12:8081;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504 http_404;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_redirect off;
    }
}
```

(3) 重启 serverb 的 nginx 服务。

```
[root@serverb conf.d]# systemctl restart nginx.service
```

(4) 因为配置了 nginx 代理，用户不需要去访问 8080 端口和 8081 端口，所以 servera 机器上相应的防火墙规则就可以删除了（图省略）

(5) 客户端测试 taomcat 虚拟主机是否还是可以正常访问。



tomcat1



tomcat2

(6) 将 tomcat1 停掉，测试是否会影响 tomcat2 虚拟主机正常运行。

```
[root@serverc init.d]# /etc/init.d/tomcat1 stop
```



An error occurred.

Sorry, the page you are looking for is currently unavailable.
Please try again later.

If you are the system administrator of this resource then you should check the [error log](#) for details.

Faithfully yours, nginx.



tomcat2