# 生产环境配置tomcat以daemon方式运行

## 一、为什么生产环境中要以daemon方式运行tomcat

通常我们在开发环境（例如windows）使用tomcat时可以直接调用$CATALINA\_HOME/bin/startup.sh来启动tomcat，调用$CATALINA\_HOME/bin/shutdown.sh来关闭tomcat。然而在生产环境（通常是linux）中我们需要将tomcat配置成以daemon守护进程来运行，主要是因为

以daemon方式运行tomcat可以使tomcat不受终端影响，不会因为退出终端而停止运行。可以让tomcat以普通用户身份运行,可以让tomcat在系统启动时自动运行

## 二、如何将tomcat配置成守护进程

将tomcat配置成守护进程需要借助apache-commons-daemon项目的jsvc工具（http://commons.apache.org/proper/commons-daemon/jsvc.html），通常该工具会包含在我们下载的tomcat包中（$CATALINA\_HOME/bin/commons-daemon-native.tar.gz）

### 配置JAVA\_HOME，CATALINA\_HOME两个环境变量

|  |
| --- |
| export JAVA\_HOME=/opt/jdk  export CATALINA\_HOME=/opt/tomcat |

### 编译安装jsvc

|  |
| --- |
| cd $CATALINA\_HOME/bin  tar -xvf commons-daemon-native.tar.gz  cd commons-daemon-1.0.x-native-src/unix  ./configure  make  cp jsvc ../.. |

### 创建运行tomcat的普通用户tomcat

|  |
| --- |
| useradd -s /sbin/nologin tomcat |

### 修改$CATALINA\_HOME的所属用户和所属组，以便可以以tomcat这个普通用户来运行

|  |
| --- |
| chown -R $CATALINA\_HOME tomcat  chgrp -R $CATALINA\_HOME tomcat |

### 5.修改制作tomcat启动脚本

$CATALINA\_HOME/bin/daemon.sh是tomcat官方为我们提供的一个脚本模版，简单修改即可

cp $CATALINA\_HOME/bin/daemon.sh /etc/init.d/tomcat

在注释位置加入chkconfig配置，使得该脚本可以被chkconfig命令识别，从而可以设置成开机启动tomcat

＃ chkconfig: 2345 85 15

由于service命令会去除系统环境变量，因此在脚本开始位置加入以下代码，配置该脚本运行时需要的几个变量

JAVA\_HOME=/opt/jdk

CATALINA\_HOME=/opt/tomcat

TOMCAT\_USER=tomcat

### 6.设置tomcat开机启动

chkconfig tomcat on

## 三、结论

经过以上配置后tomcat便能够以守护进程方式运行，并且可以随系统启动而启动

运行以下命令开启tomcat

|  |
| --- |
| service tomcat start |

运行以下命令结束tomcat

|  |
| --- |
| service tomcat stop |

# Tomcat 安全配置与性能优化

Tomcat 是 Apache软件基金会下的一个免费、开源的WEB应用服务器，它可以运行在 Linux 和 Windows 等多个平台上，由于其性能稳定、扩展性好、免费等特点深受广大用户喜爱。目前，很多互联网应用和企业应用都部署在 Tomcat 服务器上，比如我们公司的线上业务。

之前我们 tomcat 都采用的是默认的配置，因此在安全方面还是有所隐患的。上周对测试环境的所有服务器的tomcat都做了安全优化，其间也粗略做了一些性能优化，这里就简单记录分享下！

## 一、版本安全

升级当前的tomcat版本为最新稳定版本。故名思议，最新稳定版本就要兼顾最新和稳定这两个概念。一个稳定的版本，是需要时间沉淀的，而最新又是相对于稳定版而言的最新。因此我们一般会选择当前大版本中，最新版本往前推几个版本或者往前推几个月出的版本。

目前，企业常用的tomcat大版本为6.0和7.0版本，8.0版本虽然已经出了很久了，但是仍然不建议使用。

在升级版本中，需要注意的事情有两点:

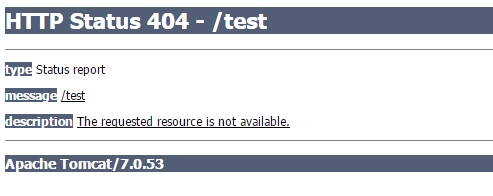
1、尽量避免跨大版本的升级

2、将当前老版本 tomcat 的server.xml、catalina.sh、web.xml和tomcat-users.xml文件进行备份，然后部署完新版本的 tomcat 之后，将这些配置文件覆盖过去即可，然后停掉旧版本，启动新版本即可完成升级操作。

## 二、隐藏版本信息

为了避免黑客针对某些版本进行攻击，因此我们需要隐藏或者伪装 Tomcat 的版本信息。

默认 Tomcat 的版本信息如下：



针对该信息的显示是由一个jar包控制的，该jar包存放在 Tomcat 安装目录下的lib目录下，名称为 catalina.jar。

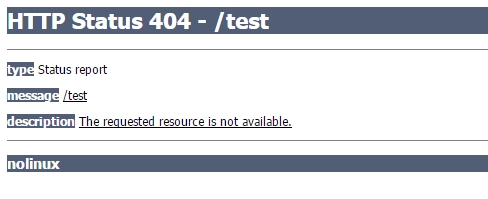
我们可以通过 jar xf 命令解压这个 jar 包会得到两个目录 META-INF 和 org ,通过修改 org/apache/catalina/util/ServerInfo.properties 文件中的 serverinfo 字段来实现来更改我们tomcat的版本信息。

文件信息如下：

|  |
| --- |
| [root@localhost ~]# cat org/apache/catalina/util/ServerInfo.properties |grep -v '^$|#'  server.info=Apache Tomcat/7.0.53  server.number=7.0.53.0  server.built=Mar 25 2014 06:20:16  当然，还有另外一种方法来实现隐藏或伪装Tomcat的版本信息，不过本质和上面一样，操作如下：  [root@localhost ~]# cd /usr/local/apache-tomcat-7.0.53/lib  [root@localhost lib]# mkdir -p org/apache/catalina/util  [root@localhost lib]# cd org/apache/catalina/util  [root@localhost util]# vim ServerInfo.properties  server.info=nolinux # 如果想修改成其它版本号，把这个地方的值改成其它值就行了 |

修改完毕之后，重启 Tomcat即可看到效果！

效果如下：



## **三**、优化 web.xml

servlet与其它适用于整个Web应用程序设置的配置文件，必须符合servlet规范的标准格式。通过它可以配置你web应用的相关选项，tomcat在启动的时候会读取这个文件，完成你开发的系统的一些初始化操作。

它可以做如下事情：

1、提供基于 servlet 的相关配置

2、增加监听器，监控session或在tomcat启动时，加载一些你希望加载的资源。比如创建数据库连接池等等

3、设置session过期时间，tomcat默认是30分钟

4、更改应用的默认网页，常用为index.html/index.jsp等

5、增加过滤器，做一些你希望的过滤操作，比如敏感词汇的过滤

6、增加一些 jstl（标准标签库）的定义，方便在jsp中直接includ进来，直接使用这些标签

7、struts，spring或hibernate的一些配置等等

下面摘录下O'REILLY 的《Tomcat 权威指南》中的一段话：

web.xml 的文件格式定义在 Servlet 规范中，因此所有符合 Servlet 规范的 Java Servlet Container 都会用到它。当 Tomcat 部署应用程序时（在激活过程中，或加载应用程序后），它都会读取通用的conf/web.xml，然后再读取web应用程序中的WEB-INF/web.xml。其实根据他们的位置，我们就可以知道，conf/web.xml文件中的设定会应用于所有的web应用程序，而某些web应用程序的WEB-INF/web.xml中的设定只应用于该应用程序本身。

如果没有WEB-INF/web.xml文件，tomcat会输出找不到的消息，但仍然会部署并使用web应用程序，servlet规范的作者想要实现一种能迅速并简易设定新范围的方法，以用作测试，因此，这个web.xml并不是必要的，不过通常最好还是让每一个上线的web应用程序都有一个自己的WEB-INF/web.xml，即使它只用做识别，但我想这是一个好的习惯。

由于Servlet规范主要是对于web程序员，而非系统管理员使用的。因此，对于运维来讲，我们可能更关心的是站点的默认网页、自定义错误页面、禁止列目录等功能。

由于，正常生产环境中，肯定不会直接由tomcat对公网提供服务，前端肯定放的有apache或者nginx。因此，针对站点的默认主页和自定义错误页面，我们均在前端的apache或者nginx中做。另外，公司也可能交由程序猿在项目内的WEB-INF/web.xml中去做定义。

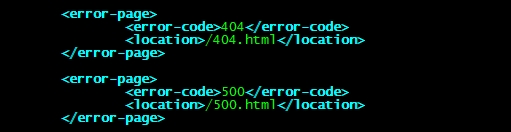
在tomcat新版本中，自动默认已经禁止列目录功能。

下面，我列出几种常见功能，在web.xml中的表现形式：

站点默认主页：



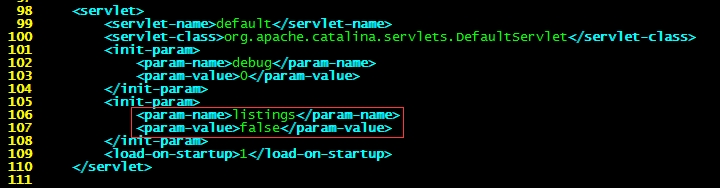
自定义错误页面：



定义会话超时时间：



禁止列目录：

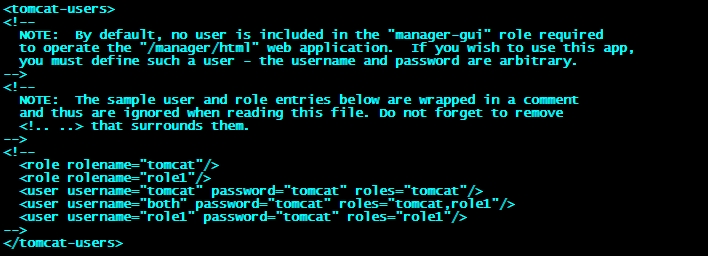


## 四、优化 tomcat-user.xml

该文件含有用户名、角色以及密码的清单文件。负责提供webapps下manager项目的登录认证管理。

在生产环境中，我们需要将该文件全部注释。

注释效果如下：

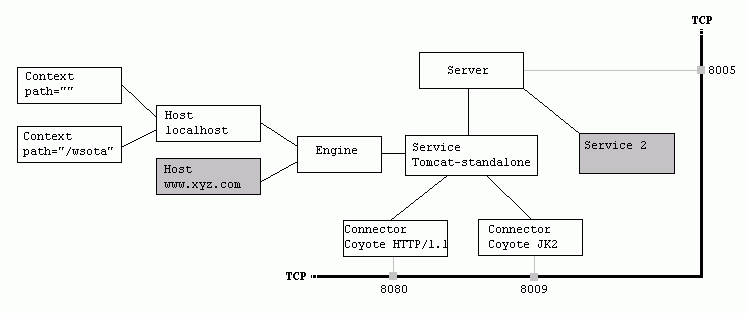


## 五、优化 server.xml

Tomcat的主配置文件，该文件中包含很多主要元素，比如Service、Connector、Host等，这些元素都会创建软件"对象"、排序及进程管道中设置的这些元素嵌套方，使我们可以执行过滤、分组等工作。

如果要对改文件做优化，我们需要先了解该文件的结构！

server.xml的结构图：



 该文件描述了如何启动Tomcat Server

|  |
| --- |
| <Server>  <Listener />  <GlobaNamingResources>  </GlobaNamingResources  <Service>  <Connector />  <Engine>  <Logger />  <Realm />  <host>  <Logger />  <Context />  </host>  </Engine>  </Service>  </Server> |

针对该文件，我们需要优化的点有如下：

1、maxThreads 连接数限制

maxThreads 是 Tomcat 所能接受最大连接数。一般设置不要超过8000以上，如果你的网站访问量非常大可能使用运行多个Tomcat实例的方法，即，在一个服务器上启动多个tomcat然后做负载均衡处理。

这里还需要注意的一点是，tomcat 和 php 不同。php可以按照cpu和内存的情况去配置连接数，上万很正常。而 java 还需要注意 jvm 的参数配置。如果不注意就会因为jvm参数过小而崩溃。

2、多虚拟主机

强烈建议不要使用 Tomcat 的虚拟主机，推荐每个站点使用一个实例。即，可以启动多个 Tomcat，而不是启动一个 Tomcat 里面包含多个虚拟主机。因为 Tomcat是多线程，共享内存，任何一个虚拟主机中的应用崩溃，都会影响到所有应用程序。虽然采用多实例的方式会产生过多的开销，但至少保障了应用程序的隔离和安全。

3、压错传输

tomcat作为一个应用服务器，也是支持 gzip 压缩功能的。我们可以在 server.xml 配置文件中的 Connector 节点中配置如下参数，来实现对指定资源类型进行压缩。

|  |
| --- |
| compression="on" # 打开压缩功能  compressionMinSize="50" # 启用压缩的输出内容大小，默认为2KB  noCompressionUserAgents="gozilla, traviata" # 对于以下的浏览器，不启用压缩  compressableMimeType="text/html,text/xml,text/javascript,text/css,text/plain"　# 哪些资源类型需要压缩 |

提示：

       Tomcat 的压缩是在客户端请求服务器对应资源后，从服务器端将资源文件压缩，再输出到客户端，由客户端的浏览器负责解压缩并浏览。相对于普通的浏览过程 HTML、CSS、Javascript和Text，它可以节省40% 左右的流量。更为重要的是，它可以对动态生成的，包括CGI、PHP、JSP、ASP、Servlet,SHTML等输出的网页也能进行压缩，压缩效率也很高。但是，压缩会增加 Tomcat 的负担，因此最好采用Nginx + Tomcat 或者 Apache + Tomcat 方式，将压缩的任务交由 Nginx/Apache 去做。

4、管理AJP端口

AJP是为 Tomcat 与 HTTP 服务器之间通信而定制的协议，能提供较高的通信速度和效率。如果tomcat前端放的是apache的时候，会使用到AJP这个连接器。由于我们公司前端是由nginx做的反向代理，因此不使用此连接器，因此需要注销掉该连接器。

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

5、更改关闭 Tomcat 实例的指令

server.xml中定义了可以直接关闭 Tomcat 实例的管理端口。我们通过 telnet 连接上该端口之后，输入 SHUTDOWN （此为默认关闭指令）即可关闭 Tomcat 实例（注意，此时虽然实例关闭了，但是进程还是存在的）。由于默认关闭 Tomcat 的端口和指令都很简单。默认端口为8005，指令为SHUTDOWN 。因此我们需要将关闭指令修改复杂一点。

当然，在新版的 Tomcat 中该端口仅监听在127.0.0.1上，因此大家也不必担心。除非黑客登陆到tomcat本机去执行关闭操作。

修改实例：

|  |
| --- |
| <Server port="8005" shutdown="9SDKJ29jksjf23sjf0LSDF92JKS9DKkjsd"> |

6、更改 Tomcat 的服务监听端口

一般公司的 Tomcat 都是放在内网的，因此我们针对 Tomcat 服务的监听地址都是内网地址。

修改实例：

|  |
| --- |
| <Connector port="8080" address="172.16.100.1" /> |

7、关闭war自动部署

默认 Tomcat 是开启了对war包的热部署的。为了防止被植入木马等恶意程序，因此我们要关闭自动部署。

修改实例：

|  |
| --- |
| <Host name="localhost" appBase=""  unpackWARs="false" autoDeploy="false"> |

## 六、禁用 Tomcat 管理页面

我们线上是不使用 Tomcat 默认提供的管理页面的，因此都会在初始化的时候就把这些页面删掉。这些页面是存放在 Tomcat 安装目录下的webapps目录下的。

我们只需要删除该目录下的所有文件即可。

当然，还有涉及管理页面的2个配置文件 host-manager.xml 和 manager.xml 也需要一并删掉。这两个文件存放在 Tomcat 安装目录下的conf/Catalina/localhost目录下。

## 七、用普通用户启动 Tomcat

为了进一步安全，我们不建议使用 root 来启动 Tomcat。这边建议使用专用用户 tomcat 或者 nobody 用户来启动 Tomcat。

在启动之前，需要对我们的tomcat 安装目录下所有文件的属主和属组都设置为指定用户。

# Tomcat 生产服务器性能优化

Tomcat内存优化

Tomcat内存优化主要是对 tomcat 启动参数优化，我们可以在 tomcat 的启动脚本 catalina.sh 中设置 JAVA\_OPTS 参数。

|  |
| --- |
| JAVA\_OPTS='-Xms1024m -Xmx2048m -XX:PermSize=256M -XX:MaxNewSize=256m -XX:MaxPermSize=256m' |

参数说明：

|  |
| --- |
| -server 启用jdk 的 server 版；  -Xms java虚拟机初始化时的最小内存；  -Xmx java虚拟机可使用的最大内存；  -XX:PermSize 内存永久保留区域  -XX:MaxPermSize 内存最大永久保留区域 |

查看Tomcat进程信息

|  |
| --- |
| [root@oracledb ~]# ps -ef |grep tomcat  root 16261 1 7 11:31 ? 00:00:25 /usr/java/jdk1.6.0\_45/bin/java -Djava.util.logging.config.file=/usr/local/tomcat6/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Xms1024m -Xmx2048m -XX:PermSize=256M -XX:MaxNewSize=256m -XX:MaxPermSize=256m -Djava.endorsed.dirs=/usr/local/tomcat6/endorsed -classpath /usr/local/tomcat6/bin/bootstrap.jar -Dcatalina.base=/usr/local/tomcat6 -Dcatalina.home=/usr/local/tomcat6 -Djava.io.tmpdir=/usr/local/tomcat6/temp org.apache.catalina.startup.Bootstrap -config /usr/local/tomcat6/conf/serverOraWdOnlyOldWay.xml -Dfile.encoding=utf-8 start |

http://ihuangweiwei.iteye.com/blog/1233941