# 一、目录结构

Bin：存放启动和关闭tomcat脚本

Conf：包含不同的配置文件,server.xml(Tomcat的主要配置文件)和web.xml

Work：存放jsp编译后产生的class文件

Webapp：存放应用程序示例，以后你要部署的应用程序也要放到此目录

Logs：存放日志文件

lib/japser/common：这三个目录主要存放tomcat所需的jar文件

# 二、server.xml配置简介

## 2.1、属性值

Port：指定一个端口，这个端口负责监听关闭tomcat的请求

Shutdown：指定向端口发送的命令字符串

Name：指定service的名字

## 2.2、Connector

Connector标签表示客户端和service之间的连接

Port：指定服务器端要创建的端口号，并在这个断口监听来自客户端的请求

minProcessors：服务器启动时创建的处理请求的线程数

maxProcessors：最大可以创建的处理请求的线程数

enableLookups：如果为true，则可以通过调用request.getRemoteHost()进行DNS查询来得到远程客户端的实际主机名，若为false则不进行DNS查询，而是返回其ip地址

redirectPort：指定服务器正在处理http请求时收到了一个SSL传输请求后重定向的端口号

acceptCount：指定当所有可以使用的处理请求的线程数都被使用时，可以放到处理队列中的请求数，超过这个数的请求将不予处理

connectionTimeout：指定超时的时间数(以毫秒为单位)

## 2.3、Engine

Engine标签表示指定service中的请求处理机，接收和处理来自Connector的请求

defaultHost：指定缺省的处理请求的主机名，它至少与其中的一个host元素的name属性值是一样的

## 2.4、Context

Context标签表示一个web应用程序，通常为WAR文件，关于WAR的具体信息见servlet规范

docBase：应用程序的路径或者是WAR文件存放的路径

path：表示此web应用程序的url的前缀，这样请求的url为

http://localhost:8080/path/\*\*\*\*

reloadable：这个属性非常重要，如果为true，则tomcat会自动检测应用程序的/WEB-INF/lib 和/WEB-INF/classes目录的变化，自动装载新的应用程序，我们可以在不重起tomcat的情况下改变应用程序

## 2.5、Host

Host标签表示一个虚拟主机

Name：指定主机名

appBase：应用程序基本目录，即存放应用程序的目录

unpackWARs：如果为true，则tomcat会自动将WAR文件解压，否则不解压，直接从WAR文件中运行应用程序

## 2.6、Logger

Logger标签表示日志，调试和错误信息

className：指定logger使用的类名，此类必须实现org.apache.catalina.Logger 接口

prefix：指定log文件的前缀

suffix：指定log文件的后缀

timestamp：如果为true，则log文件名中要加入时间，如下例:localhost\_log.2001-10-04.txt

## 2.7、Realm

Realm标签表示存放用户名，密码及role的数据库

className 指定Realm使用的类名，此类必须实现org.apache.catalina.Realm接口

Valve

(功能与Logger差不多，其prefix和suffix属性解释和Logger 中的一样)

className 指定Valve使用的类名，如用org.apache.catalina.valves.AccessLogValve类可以记录应用程序的访问信息

directory 指定log文件存放的位置

pattern 有两个值，common方式记录远程主机名或ip地址，用户名，日期，第一行请求的字符串，HTTP响应代码，发送的字节数。combined方式比common方式记录的值更多

## 2.8、注意

1：经过我测试，我设置Context 的path="",reloadable=true，然后放一个WAR文件到webapps目录，结果tomcat不能检测出此文件(重起tomcat可以)，而把此文件解压，则tomcat会自动检测出这个新的应用程序。如果不能自动检测WAR文件，我们可以利用下面管理中讲的方法来部署应用程序。

2：默认的server.xml中，Realm元素只设置了一个className属性，但此文件中也包含几个通过JDBC连接到数据库进行验证的示例(被注释掉了)，通过Realm元素我们可以实现容器安全管理(Container Managed Security)。

3：还有一些元素我们没有介绍，如Parameter，loader，你可以通过tomcat的文档获取这些元素的信息。

# 三、管理

## 3.1、配置

在进行具体的管理之前，我们先给tomcat添加一个用户，使这个用户有权限来进行管理。

打开conf目录下的tomcat-users.xml文件，在相应的位置添加下面一行：

<user name="zf" password="zf" roles="standard,manager"/>

注意：这一行的最后部分一定是/>,tomcat的文档掉了/符号，如果没有/符号的话，tomcat重起时将无法访问应用程序。通过logs/catalina.out文件你可以看到这个错误的详细信息。

然后重起tomcat，在浏览器中输入http://localhost:8080/manager/，会弹出对话框，输入上面的用户名和密码即可。

## 3.2、应用程序列表

在浏览器中输入http://localhost:8080/manager/list，浏览器将会显示如下的信息：

OK - Listed applications for virtual host localhost

/ex:running:1

/examples:running:1

/webdav:running:0

/tomcat-docs:running:0

/manager:running:0

/:running:0

面的信息分别为应用程序的路径，当前状态(running 或者stopped)，与这个程序相连的session数。

## 3.3、重新装载应用程序

在浏览器中输入 http://localhost:8080/manager/reload?path=/examples ，浏览器显示如下：

OK - Reloaded application at context path /examples

表示example应用程序装载成功，如果我们将server.xml的Context元素的reloadable属性设为true(见上面表格)，则没必要利用这种方式重新装载应用程序，因为tomcat会自动装载。

## 3.4、显示session信息

在浏览器中输入http://localhost:8080/manager/sessions?path=/examples，浏览器显示如下： OK - Session information for application at context path /examples Default maximum session inactive interval 30 minutes

## 3.5、启动和关闭应用程序

在浏览器中输入http://localhost:8080/manager/start?path=/examples和http://localhost:8080/manager/stop?path=/examples分别启动和关闭examples应用程序。

## 3.6、部署及撤销部署

WAR有两种组织方式，一种是按一定的目录结构组织文件，一种是一个后缀为WAR的压缩包，因此它的部署方式也有两种：

(1)：在浏览器中输入：

http://localhost:8080/manager/install?path=/examples&war=file:/c:examples

就会将按目录结构组织的WAR部署

(2)：如果输入

:http://localhost:8080/manager/install?path=/examples&war=jar:file:/c: examples.war!/

就会将按压缩包组织的WAR部署，注意此url后半部分一定要有!/号。

部署后就可以用 http://localhost:8080/examples 访问了。

在浏览器中输入：http://localhost:8080/manager/remove?path=/examples

就会撤销刚才部署的应用程序。

# 四、与Apache集成

虽然tomcat也可以作web服务器,但其处理静态html的速度比不上apache,且其作为web服务器的功能远不如apache,因此我们想把apache和tomcat集成起来。

## 4.1、Linux中Apache安装

1、从apache网站下载apache1.3.22源代码版本，然后使用如下命令配置安装apache：

mkdir /usr/local/apache

tar zxvf apache.1.32.tar.gz

cd apache.1.32

./configure --prefix=/usr/local/apache --enable-module=so

make

make install

注意：configure命令指定目标安装目录，并且加入DSO(Dynamic Shared Object)支持，注意一定不要忘了这一个选项。

2、然后下载webapp模块，将解压后mod\_webapp.so文件放入apache的libexec目录，编辑apache的conf目录下的httpd.conf，在这个文件的最后加入下面三行：

LoadModule webapp\_module libexec/mod\_webapp.so

WebAppConnection warpConnection warp localhost:8008

WebAppDeploy examples warpConnection /examples/

第一行是加入webapp模块，如果编译apache时不增加DSO支持，则无法使用LoadModule指令

第二行指定tomcat与apache的连接，第三行指定部署那个应用，这两个指令使用格式如下：

WebAppConnection [connection name] [provider] [host:port]

WebAppDeploy [application name] [connection name] [url path]

其中connection name指定连接名，provider只能是warp，port端口与你的tomcat的配置文件server.xml最后几行指定的要保持一致。文件如下：

<Service name="Tomcat-Apache">

<Connector className="org.apache.catalina.connector.warp.WarpConnector"

port="8008" minProcessors="5" maxProcessors="75" enableLookups="true"

acceptCount="10" debug="0"/>

\*\*\*\*\*\*

</Service>

application name与你在tomcat中部署的应用名一致，url path指定访问这个应用的url。例如上面的例子可以通过http://localhost/examples/来访问tomcat中的examples应用。

# 五、中文问题

一般jsp的乱码问题可以通过在jsp中加入<%@ page contentType="text/html;charset=

GB2312" %>来解决，至于servlet的乱码在可以使用servlet2.3中提供的HttpServeletRequest.setCharacterEncoding函数。更详细的中文问题请见JSP/Servlet 中的汉字编码问题。

# 六、总结

tomcat作为一个servlet(jsp也被编译为servlet执行)容器，其应用前景是非常好的，如果与jboss结合起来，则可以实现sun的j2ee规范(用jboss作ejb服务器)。jboss的官方网站也提供集成了tomcat3.2\*的jboss以供下载。另外一个开源的应用服务器(enhydra) 也是基于tomcat的，其提供了更友好的管理界面，部署应用程序也更简单，功能也更强大。