# Tomcat详细说明

[Tomcat详细说明 1](#_Toc268607942)

[1. Tomcat背景 1](#_Toc268607943)

[2. Tomcat的目录结构 1](#_Toc268607944)

[3. tomcat类加载 2](#_Toc268607945)

[4. Server.Xml配置简介 2](#_Toc268607946)

[5. Web.xml配置管理 5](#_Toc268607947)

[6. 管理 6](#_Toc268607948)

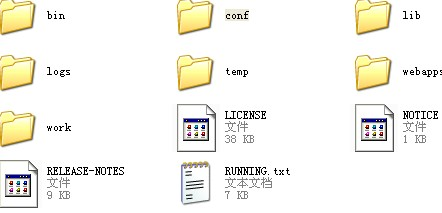
[7. Web应用部署的原理及其方式： 7](#_Toc268607949)

[8. 常见错误 9](#_Toc268607950)

# Tomcat背景

自从JSP发布之后，推出了各式各样的JSP引擎。Apache Group在完成GNUJSP1.0的开发以后，开始考虑在SUN的JSWDK基础上开发一个可以直接提供Web服务的JSP服务器，当然同时也支持Servlet, 这样Tomcat就诞生了。Tomcat是jakarta项目中的一个重要的子项目，，同时它又是sun公司官方推荐的servlet和jsp容器。servlet和jsp的最新规范都可以在tomcat的新版本中得到实现。其次，Tomcat是完全免费的软件，任何人都可以从互联网上自由地下载。Tomcat与Apache的组合相当完美。

# Tomcat的目录结构



tomcat   
|---bin Tomcat：存放启动和关闭tomcat脚本；   
|---conf Tomcat：存放不同的配置文件（server.xml和web.xml）；   
|---doc：存放Tomcat文档；   
|---lib/japser/common：存放Tomcat运行需要的库文件（JARS）；   
|---logs：存放Tomcat执行时的LOG文件；   
|---src：存放Tomcat的源代码；   
|---webapps：Tomcat的主要Web发布目录（包括应用程序示例）；   
|---work：存放jsp编译后产生的class文件；

# tomcat类加载

Bootstrap($JAVA\_HOME/jre/lib/ext/\*.jar)   
System($CLASSPATH/\*.class和指定的jar)   
Common($CATALINA\_HOME/common 下的classes,lib,endores三个子目录)   
Catalina ($CATALINA\_HOME/server/下的classes和lib目录仅对Tomcat可见)   
&Shared($CATALINA\_HOME/shared/下的classes和lib目录以及$CATALINA\_HOME/lib目录)仅对Web应用程序可见,对Tomcat不可见WebApp($WEBAPP/Web-INF/\*仅对该WEB应用可见classes/\*.class lib/\*.jar)   
  
  
加载类和资源的顺序为：   
1、/Web-INF/classes   
2、/Web-INF/lib/\*.jar   
3、Bootstrap   
4、System   
5、$CATALINA\_HOME/common/classes   
6、$CATALINA\_HOME/common/endores/\*.jar   
7、$CATALINA\_HOME/common/lib/\*.jar   
8、$CATALINA\_HOME/shared/classes   
9、$CATALINA\_HOME/shared/lib/\*.jar

# Server.Xml配置简介

Server   
A Server element represents the entire Catalina servlet container. (Singleton)  
          1、port 指定一个端口，这个端口负责监听关闭tomcat的请求   
          2、shutdown 指定向端口发送的命令字符串



  service:

A Service element represents the combination of one or more Connector components that share a single Engine   
Service是这样一个集合：它由一个或者多个Connector组成，以及一个Engine，负责处理所有Connector所获得的客户请求  
 name 指定service的名字



Connector

一个Connector将在某个指定端口上侦听客户请求，并将获得的请求交给Engine来处理，从Engine处获得回应并返回客户   
TOMCAT有两个典型的Connector，一个直接侦听来自browser的http请求，一个侦听来自其它WebServer的请求   
Coyote Http/1.1 Connector 在端口8080处侦听来自客户browser的http请求 (你可能辉修改的地方)

  
Coyote JK2 Connector 在端口8009处侦听来自其它WebServer(Apache)的servlet/jsp代理请求

   
 Engine

The Engine element represents the entire request processing machinery associated with a particular Service   
It receives and processes all requests from one or more Connectors   
and returns the completed response to the Connector for ultimate transmission back to the client   
Engine下可以配置多个虚拟主机Virtual Host，每个虚拟主机都有一个域名   
当Engine获得一个请求时，它把该请求匹配到某个Host上，然后把该请求交给该Host来处理   
Engine有一个默认虚拟主机，当请求无法匹配到任何一个Host上的时候，将交给该默认Host来处理



Host

代表一个Virtual Host，虚拟主机，每个虚拟主机和某个网络域名Domain Name相匹配   
每个虚拟主机下都可以部署(deploy)一个或者多个Web App，每个Web App对应于一个Context，有一个Context path   
当Host获得一个请求时，将把该请求匹配到某个Context上，然后把该请求交给该Context来处理   
匹配的方法是“最长匹配”，所以一个path==""的Context将成为该Host的默认Context   
所有无法和其它Context的路径名匹配的请求都将最终和该默认Context匹配

         1、name 指定主机名   
          2、appBase 应用程序基本目录，即存放应用程序的目录   
          3、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   
    Logger (表示日志，调试和错误信息)：   
          1、className 指定logger使用的类名，此类必须实现org.apache.catalina.Logger 接口   
          2、prefix 指定log文件的前缀   
          3、suffix 指定log文件的后缀   
          4、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   
   Realm (表示存放用户名，密码及role的数据库)：   
          1、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)：   
          1、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）：   
    1、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



Context   
一个Context对应于一个Web Application，一个Web Application由一个或者多个Servlet组成   
Context在创建的时候将根据配置文件$CATALINA\_HOME/conf/web.xml和$WEBAPP\_HOME/WEB-INF/web.xml载入Servlet类   
当Context获得请求时，将在自己的映射表(mapping table)中寻找相匹配的Servlet类   
如果找到，则执行该类，获得请求的回应，并返回  
          1、docBase 应用程序的路径或者是WAR文件存放的路径   
          2、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/\*\*\*\*   
          3、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

# Web.xml配置管理

1、默认(欢迎)文件的设置   
 在tomcat4\conf\web.xml中，<welcome-file-list>与IIS中的默认文件意思相同。   
 <welcome-file-list>   
 <welcome-file>index.html</welcome-file>   
 <welcome-file>index.htm</welcome-file>   
 <welcome-file>index.jsp</welcome-file>   
 </welcome-file-list>   
  
2、报错文件的设置   
<error-page>   
<error-code>404</error-code>   
<location>/notFileFound.jsp</location>   
</error-page>   
<error-page>   
<exception-type>java.lang.NullPointerException</exception-type>   
<location>/null.jsp</location>   
</error-page>   
如果某文件资源没有找到，服务器要报404错误，按上述配置则会调用\webapps\ROOT\notFileFound.jsp。   
如果执行的某个JSP文件产生NullPointException ，则会调用\webapps\ROOT\null.jsp   
  
3、会话超时的设置   
设置session 的过期时间，单位是分钟；   
<session-config>   
<session-timeout>30</session-timeout>   
</session-config>   
  
4、过滤器的设置   
<filter>   
<filter-name>FilterSource</filter-name>   
<filter-class>project4. FilterSource </filter-class>   
</filter>   
<filter-mapping>   
<filter-name>FilterSource</filter-name>   
<url-pattern>/WwwServlet</url-pattern>   
(<url-pattern>/haha/\*</url-pattern>)   
</filter-mapping>   
  
过滤：   
1) 身份验证的过滤Authentication Filters   
2) 日志和审核的过滤Logging and Auditing Filters   
3) 图片转化的过滤Image conversion Filters   
4) 数据压缩的过滤Data compression Filters   
5) 加密过滤Encryption Filters   
6) Tokenizing Filters   
7) 资源访问事件触发的过滤Filters that trigger resource access events XSL/T 过滤XSL/T filters   
9) 内容类型的过滤Mime-type chain Filter 注意监听器的顺序，如：先安全过滤，然后资源，   
然后内容类型等，这个顺序可以自己定。

# 管理

    1、用户配置   
      在进行具体Tomcat管理之前，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="user" password="user" roles="standard,manager"/>   
    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。   
  
    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   
     上面显示的信息分别为：应用程序的路径、当前状态、连接这个程序的session数   
  
   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      
这表示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，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。   
  
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   
  
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。

# Web应用部署的原理及其方式：

### ·原理：

1：将可部署目录拷贝到tomcat/webapps目录中，由虚拟主机自动部署（部署方式1， 3， 4）

2: 部署时，可部署目录不需要拷贝到webapps目录中，通过context上下文动态查找( 部署方式2，3 )。

### ·部署方式：

1.将web应用的可部署目录，如hello.jsp(内容：This is my Hello page)，拷贝到tomcat/webapps/ROOT目录中，由tomcat自动完成部署（虚拟主机 autoDeployee=”true”）。，在IE地址栏输入[http://localhost：8080/hello.jsp](http://localhost/hello.jsp)，是否显示以上的内容

2 . 法一：

通过manager进行web应用部署，目录：D:\Program Files\Apache Software Foundation\Tomcat 5.0\conf\Catalina\localhost 下的manager.xml文件，修改如下：

<Context docBase="/F:/1202web/WebRoot" path="/1202web">

法二：

修改：D:\Program Files\Apache Software Foundation\Tomcat 5.0\conf下的server.Xml文件，修改如下：

<Engine defaultHost="localhost" name="Catalina">

<Host appBase="F:\web" name="www.sina.com.cn">

<Logger className="org.apache.catalina.logger.FileLogger" prefix="localhost\_log." suffix=".txt" timestamp="true"/>

</Host>

<Host appBase="webapps" name="localhost">

<Logger className="org.apache.catalina.logger.FileLogger" prefix="localhost\_log." suffix=".txt" timestamp="true"/>

<Context docBase="/F:/1202web/WebRoot" path="/demo"/>

</Host>

<Logger className="org.apache.catalina.logger.FileLogger" prefix="catalina\_log." suffix=".txt" timestamp="true"/>

<Realm className="org.apache.catalina.realm.UserDatabaseRealm"/>

</Engine>

path :上下文Context root URL

path :上下文Context名 ，Context名是随便起的

·在IE地址栏输入：http://localhost:8080/demo/，是否出现index.jsp页面的内容：

· 新建的上下文Context名demo，会在在localhost文件下有个demo.xml文件，目录：D:\Program Files\Apache Software Foundation\Tomcat 5.0\conf\Catalina\localhost，在demo.xml文里可以知道上下文demo的信息

### 3. 部署war（原理1、2）

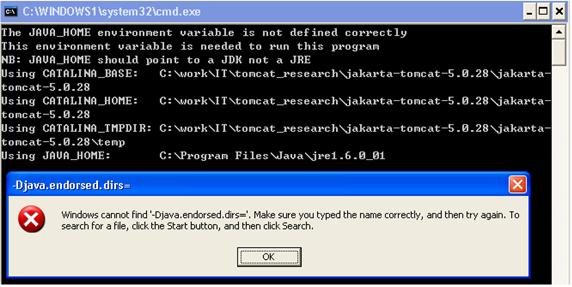
### 4 .工具部署

# 常见错误

3.1 环境变量设置问题

如果机器上没有安装JDK或者环境变量JAVA\_HOME或CATALINA\_HOME没有设置正确，Tomcat就不能正常启动。

3.1.1 下面这个错误对话框的意思是Windows不能发现“-Djava.endorsed.dirs=”命令，下面背景的黑窗口的错误是JAVA\_HOME环境变量应该指向JDK而不是JRE的根目录（本例故意把把JAVA\_HOME指向JRE的根目录，所以产生这个错误）。在这种情况下，我们在可以重新设置环境变量JAVA\_HOME并指向JDK的安装目录即可。

'800')this.width='800';if(this.height>'600')this.height='600';" border=0>

3.1.2 下面这个错误是因为CATALINA\_HOME环境变量设置不正确，从而造成Tomcat不能正常启动。

C:\carl>startup

CATALINA\_HOME C:\

The CATALINA\_HOME environment variable is not defined correctly

This environment variable is needed to run this program

3.1.3 下面这个错误是由于错误地设置了Java启动参数xms256M，正确的写法应该是-Xms256M，请注意大小写。大家知道，在Java命令行中，-Xms表示JVM开始启动时所分配的内存大小，而-Xmx表示JVM运行时最大所能占用的的内存大小，如果您的应用程序所需的内存超过Xmx的值，JVM就会抛出Out of Memeory(内存不足)的异常而停止。当然，Xmx的最大值不应超过物理内存的70%。

D:\back\blog>start\_tomcat\_debug.bat

D:\back\blog>set JAVA\_OPTS=-xms256M

D:\back\blog>rem 请将JAVA\_HOME环境变量修改到您的JDK安装目录

D:\back\blog>set JAVA\_HOME=C:\Program Files\Java\jdk1.5.0\_09

D:\back\blog>rem 请将CATALINA\_HOME环境变量修改到您的Tomcat安装目录

D:\back\blog>set CATALINA\_HOME=C:\carl\it\tomcat\_research\jakarta-tomcat-5.0.28

D:\back\blog>rem 开始调用catalina.bat文件

D:\back\blog>call C:\carl\it\tomcat\_research\jakarta-tomcat-5.0.28\bin\catalina.bat debug

Using CATALINA\_BASE:   C:\carl\it\tomcat\_research\jakarta-tomcat-5.0.28

Using CATALINA\_HOME:   C:\carl\it\tomcat\_research\jakarta-tomcat-5.0.28

Using CATALINA\_TMPDIR: C:\carl\it\tomcat\_research\jakarta-tomcat-5.0.28\temp

Using JAVA\_HOME:   C:\Program Files\Java\jdk1.5.0\_09

debug

无效的选项： -xms256M

用法：jdb <选项> <类> <参数> ...

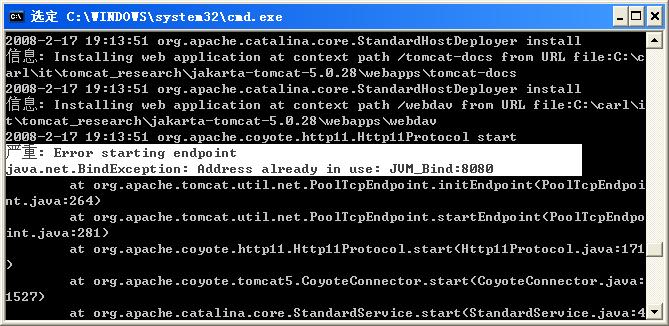
如果遇到类似问题，可以用下面的方法重新设置一下JVM启动时的内存参数，在startup.bat脚本中加上下面这一行：

set JAVA\_OPTS=-Xms256m -Xmx512m

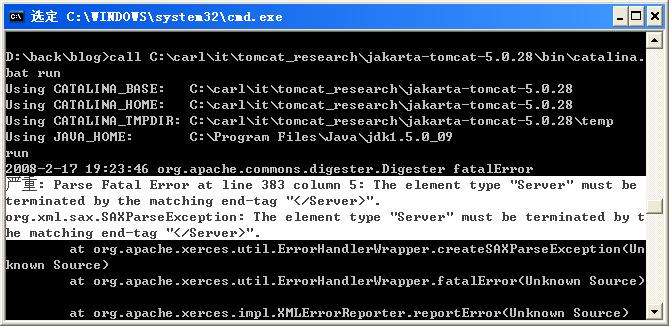
上面这行表示Tomcat初始启动内存至少需要256M，而最大可能占用内存为为512M。有兴趣的读者可以在startup.bat里加上这行参数，启动Tomcat，您会从Windows任务管理器中看到您使用的内存迅速飙升。

3.2 Tomcat应用服务器的配置问题及其中部署的某个Web应用问题

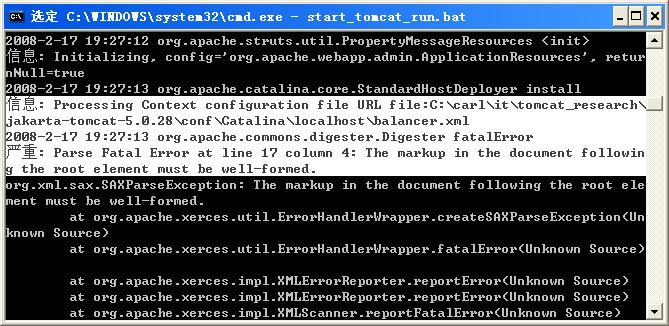
3.2.1   Tomcat端口冲突，具体错误如下窗口所示。这表明缺省的8080端口已经被占有，所以Tomcat不能启动，我们可以通过修改Tomcat的配置文件server.xml来重新定义端口号即可启动，或者停止已经使用8080端口的程序。有时我们并没有启动Tomcat，也没有其它应用程序占用8080端口，但是Tomcat还抛出这个错误。这可能是因为上次我们关闭Tomcat时，并没有真正关闭Tomcat。在这种情况下，请从Windows任务管理器中检查一下有无一个java.exe的程序正在运行，如果发现，强行kill java.exe，然后重启试试。

'800')this.width='800';if(this.height>'600')this.height='600';" border=0>

3.2.2   Tomcat配置文件有错误，产生异常。有时我们在修改server.xml文件时，不小心多加或者多删了一个</>标签，可能造成Tomcat不能正常启动。下面的窗口说明了这个问题。

'800')this.width='800';if(this.height>'600')this.height='600';" border=0>

3.2.3   Tomcat中部署的某个Web应用发生异常。这时Tomcat能正常启动，但在Tomcat启动窗口里发现一大堆异常，如data source没有正确定义，程序抛出异常等等都有关系。下面的这个例子是Tomcat自带的balancer web应用程序启动时产生错误。该问题是因为我们错误的修改了balancer.xml文件造成的。

'800')this.width='800';if(this.height>'600')this.height='600';" border=0>

3.2.4   Web应用程序的jar文件丢失，如JDBC数据库连接文件class12.jar, mysql.jar没有拷贝到Webapp的WEB-INF\lib目录下就会产生异常。尤其要注意的是在Tomcat5以前的版本中，公共jar文件包通常放在Tomcat安装目录下的common\lib子目录中，但在Tomcat6.0中，这个公共目录改为lib子目录。

3.3 未知错误

3.3.1 有时Tomcat的启动窗口一闪而过，根本就看不出启动过程中发生了什么错误。这中间的原因有好多种，最常见的解决办法就是使用run命令，打开startup.bat文件，找到下面这行：

call "%EXECUTABLE%" start %CMD\_LINE\_ARGS%

并将它修改为：

call "%EXECUTABLE%" run %CMD\_LINE\_ARGS%

这样，Tomcat启动时就不会弹出新窗口，我们就可以从容不迫地观察Tomcat的启动错误，并解决问题，请参考上面关于run命令的解释。

3.3.2 另外，阅读Tomcat的启动日志文件也是我们解决问题的重要办法，缺省的Tomcat日志是放在Tomcat安装目录的logs子目录下。例如下面这段日志说明JspServletViewer这个Web应用缺少Map Object的相关jar文件包，从而找不到com/esri/mo2/map/core/Layout这个Java类。

2008-02-23 11:19:30 StandardContext[/JspServletViewer]Exception sending context initialized event to listener instance of class com.esri.svr.cmn.FileRewriterContextListener

java.lang.NoClassDefFoundError: com/esri/mo2/map/core/Layout

at com.esri.svr.cat.ServiceXMLHandler.startElement ServiceXMLHandler.java:47)

at org.apache.xerces.parsers.AbstractSAXParser.startElement(Unknown Source)

at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanStartElement(Unknown Source)