tomcat_百度百科

tomcat编辑

Tomcat是Apache 软件基金会（Apache Software Foundation）的Jakarta 项目中的一个核心项目，由Apache、Sun 和其他一些公司及个人共同开发而成。由于有了Sun 的参与和支持，最新的Servlet 和JSP 规范总是能在Tomcat 中得到体现，Tomcat 5支持最新的Servlet 2.4 和JSP 2.0 规范。因为Tomcat 技术先进、性能稳定，而且免费，因而深受Java 爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的Web 应用服务器。

Tomcat 服务器是一个免费的开放源代码的Web 应用服务器，属于轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试JSP 程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好Apache 服务器，可利用它响应HTML（标准通用标记语言下的一个应用）页面的访问请求。实际上Tomcat 部分是Apache 服务器的扩展，但它是独立运行的，所以当你运行tomcat 时，它实际上作为一个与Apache 独立的进程单独运行的。

诀窍是，当配置正确时，Apache 为HTML页面服务，而Tomcat 实际上运行JSP 页面和Servlet。另外，Tomcat和IIS等Web服务器一样，具有处理HTML页面的功能，另外它还是一个Servlet和JSP容器，独立的Servlet容器是Tomcat的默认模式。不过，Tomcat处理静态HTML的能力不如Apache服务器。目前Tomcat最新版本为9.0。

中文名
汤姆猫
外文名
Apache Tomcat
性　　质
Java Web服务器
属　　于
Apache 软件基金会
最新版本
9.0

目录

名称由来

编辑

Tomcat最初是由Sun的软件构架师詹姆斯·邓肯·戴维森开发的。后来他帮助将其变为开源项目，并由Sun贡献给Apache软件基金会。由于大部分开源项目O'Reilly都会出一本相关的书，并且将其封面设计成某个动物的素描，因此他希望将此项目以一个动物的名字命名。因为他希望这种动物能够自己照顾自己，最终，他将其命名为Tomcat（英语公猫或其他雄性猫科动物）。而O'Reilly出版的介绍Tomcat的书籍（ISBN 0-596-00318-8）[1]的封面也被设计成了一个公猫的形象。而Tomcat的Logo兼吉祥物也被设计成了一只公猫。

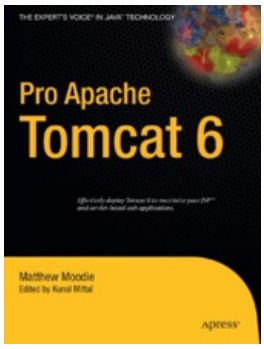
版本差异

编辑

Apache Tomcat 7.x

是目前的开发焦点。它在汲取了Tomcat 6.0.x优点的基础上，实现了对于Servlet 3.0、JSP 2.2和EL 2.2等特性的支持。除此以外的改进列表如下：

- Web应用内存溢出侦测和预防
- 增强了管理程序和服务器管理程序的安全性
- 一般 CSRF保护



Apache Tomcat

Apache Tomcat (18张)

- 支持web应用中的外部内容的直接引用
- 重构 (connectors, lifecycle)及很多核心代码的全面梳理

Apache Tomcat 6.x

在汲取 Tomcat 5.5.x优点的基础上，实现了Servlet 2.5和JSP 2.1等特性的支持。除此以外的改进列表如下：

- 内存使用优化
- 更大的IO容量
- 重构聚类

Apache Tomcat 5.x

Apache Tomcat 5.5.x 和Apache Tomcat 5.0.x 对于Servlet和JSP的支持是一样的。大量底层代码里的重大修改，带来性能的提升、稳定性的提升及整体成本。详请参照Apache Tomcat 5.5的更新日志。

Apache Tomcat 5.0.x在Apache Tomcat 4.1的基础上做了很多改动，包括：

- 性能优化和减少垃圾回收动作
- 重构程序部署，通过一个可选的独立部署程序，允许在将一个web应用放进产品前验证和编译它
- 基于JMX的服务器全面监视及web程序管理
- 提高Taglibs的支撑能力，包括改进的数据池和tag插件
- 改进平台集成性，包括Windows和Unix
- 基于JMX的嵌入
- 增强的安全管理支撑
- 集成session集群
- 文档扩充

最新版本

编辑

Servlet/JSP_ 规范版本 Apache Tomcat 版本。

3.1/2.3____ 9.0.X[1]

3.0/2.2____ 7.0.X

2.5/2.1____ 6.0.X

2.4/2.0____ 5.5.X

2.3/1.2____ 4.1.X

2.2/1.1____ 3.3.X

配置方法

编辑

启动内存参数的配置

tomcat/bin/catalina.bat 如果是linux 就是 catalina.sh

在rem 的后面增加如下参数

set JAVA_OPTS= -Xms256m -Xmx256m -XX:MaxPermSize=64m

修改Tomcat的JDK目录

打开tomcat/bin/catalina.bat

在最后一个rem后面增加

set JAVA_HOME=C:\Program Files\Java\jdk1.8.0

增加虚拟目录

/tomcat/conf/server.xml

第一行是以前默认存在的，第二行是新增的

使用默认配置的tomcat,另外虚拟目录也可这设置:

因为默认情况下,tomcat启动过程中配置虚拟目录的时候会从 webapps目录下查找webContent应用。

这样配置好了，即使以后从一台服务器移植到另一台服务器，不做任何修改也能运行起来。

GET方式URL乱码问题解决

打开 tomcat/conf/server.xml

查找下面这部分，在最后增加一段代码就可以了。

.....

URIEncoding="UTF-8" useBodyEncodingForURI="true"

.....

/>

其中的UTF-8 请根据你的需要自己修改，比如GBK

虚拟主机配置文件

tomcat/conf/server.xml

unpackWARs="true" autoDeploy="true"

xmlValidation="false" xmlNamespaceAware="false">

...

unpackWARs="true" autoDeploy="true"

xmlValidation="false" xmlNamespaceAware="false">

unpackWARs="true" autoDeploy="true"

xmlValidation="false" xmlNamespaceAware="false">

数据源配置

比较复杂，各个版本都有所不同，请直接查看 http://java2000. net/p1906，包括tomcat5.0,tomcat5.5x,tomcat6.0的各个版本的配置方法。

更多关于Tomcat的使用，请看参考资料

十个技巧

编辑

配置系统管理

（Admin Web Application）
大多数商业化的JavaEE服务器都提供一个功能强大的管理界面，且大都采用易于理解的Web应用界面。Tomcat按照自己的方式，同样提供一个成熟的管理工具，并且丝毫不逊于那些商业化的竞争对手。Tomcat的Admin Web Application最初在4.1版本时出现，当时的功能包括管理context、data source、user和group等。当然也可以管理像初始化参数，user、group、role的多种数据库管理等。在后续的版本中，这些功能将得 到很大的扩展，但现有的功能已经非常实用了。
Admin Web Application被定义在自动部署文件：CATALINA_BASE/webapps/admin.xml 。
必须编辑这个文件，以确定Context中的docBase参数是绝对路径。也就是说， CATALINA_BASE/webapps/admin.xml 的路径是绝对路径。作为另外一种选择，也可以删除这个自动部署文件，而在server.xml文件中建立一个Admin Web Application的context，效果是一样的。不能管理Admin Web Application这个应用，换而言之，除了删除CATALINA_BASE/webapps/admin.xml ，可能什么都做不了。
如果使用UserDatabaseRealm（默认），将需要添加一个user以及一个role到CATALINA_BASE/conf/tomcat-users.xml 文件中。你编辑这个文件，添加一个名叫“admin”的role 到该文件中，如下：
同样需要有一个用户，并且这个用户的角色是“admin”。象存在的用户那样，添加一个用户（改变密码使其更加安全）：
当完成这些步骤后，请重新启动Tomcat，访问http://localhost:8080/admin，将看到一个登录界面。Admin Web Application采用基于容器管理的安全机制，并采用了Jakarta Struts框架。一旦作为“admin”角色的用户登录管理界面，将能够使用这个管理界面配置Tomcat。

配置应用管理

Manager Web Application让你通过一个比Admin Web Application更为简单的用户界面，执行一些简单的Web应用任务。
Manager Web Application被定义在一个自动部署文件中：
CATALINA_BASE/webapps/manager.xml 。
必须编辑这个文件，以确保context的docBase参数是绝对路径，也就是说CATALINA_HOME/server/webapps/manager的绝对路径。
如果使用的是UserDatabaseRealm，那么需要添加一个角色和一个用户到CATALINA_BASE/conf/tomcat-users.xml文件中。接下来，编辑这个文件，添加一个名为“manager”的角色到该文件中：
同样需要有一个角色为“manager”的用户。像已经存在的用户那样，添加一个新用户（改变密码使其更加安全）：
然后重新启动Tomcat，访问http://localhost/manager/list，将看到一个很朴素的文本型管理界面，或者访问http://localhost/manager/html/list，将看到一个HMTL的管理界面。不管是哪种方式都说明你的Manager Web Application现在已经启动了。
Manager application可以在没有系统管理特权的基础上，安装新的Web应用，以用于测试。如果我们有一个新的web应用位于 /home/user/hello下在，并且想把它安装到 /hello下，为了测试这个应用，可以这么做，在第一个文件框中输入“/hello”（作为访问时的path），在第二个文本框中输入“file: /home/user/hello”（作为Config URL）。
Manager application还允许停止、重新启动、移除以及重新部署一个web应用。停止一个应用使其无法被访问，当有用户尝试访问这个被停止的应用时，将 看到一个503的错误——“503 - This application is not currently available”。
移除一个web应用，只是指从Tomcat的运行拷贝中删除了该应用，如果重新启动Tomcat，被删除的应用将再次出现（也就是说，移除并不是指从硬盘上删除）。

部署一个应用

有两个办法可以在系统中部署web服务。
1> 拷贝WAR文件或者web应用文件夹（包括该web的所有内容）到$CATALINA_BASE/webapps目录下。
2> 为web服务建立一个只包括context内容的XML片断文件，并把该文件放到$CATALINA_BASE/webapps目录下。这个web应用本身可以存储在硬盘上的任何地方。
如果有一个WAR文件，想部署它，则只需要把该文件简单的拷贝到CATALINA_BASE/webapps目录下即可，文件必须以“.war”作 为扩展名。一旦Tomcat监听到这个文件，它将（缺省的）解开该文件包作为一个子目录，并以WAR文件的文件名作为子目录的名字。接下来，Tomcat 将在内存中建立一个context，就好象在server.xml文件里建立一样。当然，其他必需的内容，将从server.xml中的 DefaultContext获得。
部署web应用的另一种方式是写一个Context XML片断文件，然后把该文件拷贝到CATALINA_BASE/webapps目录下。一个Context片断并非一个完整的XML文件，而只是一个 context元素，以及对该应用的相应描述。这种片断文件就像是从

server.xml中切取出来的context元素一样，所以这种片断被命名为 “context片断”。
举个例子，如果我们想部署一个名叫MyWebApp.war的应用，该应用使用realm作为访问控制方式，我们可以使用下面这个片断：
debug=“0” privileged=“true”>
resourceName=“UserDatabase”/>
把该片断命名为“MyWebApp.xml”，然后拷贝到CATALINA_BASE/webapps目录下。
这种context片断提供了一种便利的方法来部署web应用，不需要编辑server.xml，除非想改变缺省的部署特性，安装一个新的web应用时不需要重启动Tomcat。

配置虚拟主机

（Virtual Hosts）
关于server.xml中“Host”这个元素，只有在设置虚拟主机的才需要修改。虚拟主机是一种在一个web服务器上服务多个域名的机制，对每个域 名而言，都好象独享了整个主机。实际上，大多数的小型商务网站都是采用虚拟主机实现的，这主要是因为虚拟主机能直接连接到Internet并提供相应的带 宽，以保障合理的访问响应速度，另外虚拟主机还能提供一个稳定的固定IP。
基于名字的虚拟主机可以被建立在任何web服务器上，建立的方法就是通过在域名服务器（DNS）上建立IP地址的别名，并且告诉web服务器把去往不同域 名的请求分发到相应的网页目录。
在Tomcat中使用虚拟主机，需要设置DNS或主机数据。为了测试，为本地IP设置一个IP别名就足够了，接下来，你需要在server.xml中添加几行内容，如下：
port=“8080” minProcessors=“5” maxProcessors=“75”
enableLookups=“true” redirectPort=“8443”/>
port=“8443” minProcessors=“5” maxProcessors=“75”
acceptCount=“10” debug=“0” scheme=“https” secure=“true”/>
clientAuth=“false” protocol=“TLS” />
unpackWARs=“true” autoDeploy=“true”>
reloadable=“true” crossContext=“true”>
Tomcat的server.xml文件，在初始状态下，只包括一个虚拟主机，但是它容易被扩充到支持多个虚拟主机。在前面的例子中展示的是一个简单的 server.xml版本，其中粗体部分就是用于添加一个虚拟主机。每一个Host元素必须包括一个或多个context元素，所包含的context元 素中必须有一个是默认的context，这个默认的context的显示路径应该为空（例如，path=“”）。

配置基础验证

（Basic Authentication）
容器管理验证方法控制着当用户访问受保护的web应用资源时，如何进行用户的身份鉴别。当一个web应用使用了Basic Authentication（BASIC参数在web.xml文件中auto-method元素中设置），而有用户访问受保护的web应用时， Tomcat将通过HTTP Basic Authentication方式，弹出一个对话框，要求用户输入用户名和密码。在这种验证方法中，所有密码将被以64位的编码方式在网络上传输。
注意：使用Basic Authentication通常被认为是不安全的，因为它没有强健的加密方法，除非在客户端和服务器端都使用HTTPS或者其他密码加密码方式（比如， 在一个虚拟私人网络中）。若没有额外的加密方法，网络管理员将能够截获（或滥用）用户的密码。但是，如果是刚开始使用Tomcat，或者你想在你的 web应用中测试一下基于容器的安全管理，Basic Authentication还是非常易于设置和使用的。只需要添加和两个元素到web应用的web.xml文件中，并且在CATALINA_BASE/conf/tomcat-users.xml 文件中添加适当的和即可，然后重新启动Tomcat。

配置单点登录

（Single Sign-On）
一旦设置了realm和验证的方法，就需要进行实际的用户登录处理。一般说来，对用户而言登录系统是一件很麻烦的事情，必须尽量减少用户登录验证的 次数。作为缺省的情况，当用户第一次请求受保护的资源时，每一个web应用都会要求用户登录。如果运行了多个web应用，并且每个应用都需要进行单独的 用户验证，那这看起来就有点像在用户搏斗。用户们不知道怎样才能把多个分离的应用整合成一个单独的系统，所有用户也就不知道他们需要访问多少个不 同的应用，只是很迷惑，为什么总要不停的登录。
Tomcat 4的“single sign-on”特性允许用户在访问同一虚拟主机下所有web应用时，只需登录一次。为了使用这个功

能，只需要在Host上添加一个SingleSignOn Valve元素即可，如下所示：

debug=“0”/>

在Tomcat初始安装后，server.xml的注释里面包括SingleSignOn Valve配置的例子，只需要去掉注释，即可使用。那么，任何用户只要登录过一个应用，则对于同一虚拟主机下的所有应用同样有效。

使用single sign-on valve有一些重要的限制：

1> value必须被配置和嵌套在相同的Host元素里，并且所有需要进行单点验证的web应用（必须通过context元素定义）都位于该Host下。

2> 包括共享用户信息的realm必须被设置在同一级Host中或者嵌套之外。

3> 不能被context中的realm覆盖。

4> 使用单点登录的web应用最好使用一个Tomcat的内置的验证方式（被定义在web.xml中的中），这比自定 义的验证方式强，Tomcat内置的的验证方式包括basic、digest、form和client-cert。

5> 如果你使用单点登录，还希望集成一个第三方的web应用到你的网站中来，并且这个新的web应用使用它自己的验证方式，而不使用容器管理安全，那你基本上 就没招了。用户每次登录原来所有应用时需要登录一次，并且在请求新的第三方应用时还得再登录一次。

6> 单点登录需要使用cookies。

用户定制目录

一些站点允许个别用户在服务器上发布网页。例如，一所大学的学院可能想给每一位学生一个公共区域，或者是一个ISP希望给一些web空间给他的客户，但这又不是虚拟主机。在这种情况下，一个典型的方法就是在用户名前面加一个特殊字符（~），作为每位用户的网站，比如：

提供两种方法在主机上映射这些个人网站，主要使用一对特殊的Listener元素。Listener的className属性应该是org.apache.catalina.startup.UserConfig，userClass属性应该是几个映射类之一。如果电脑系统是 Unix，它将有一个标准的/etc/passwd文件，该文件中的帐号能够被运行中的Tomcat很容易的读取，该文件指定了用户的主目录，使用PasswdUserDatabase 映射类。

Tomcat

directoryName=“public_html”

userClass=“org.apache.catalina.startup.PasswdUserDatabase”/>

web文件需要放置在像/home/users/ian/public_html 或者 /users/jbrittain/public_html一样的目录下面。当然你也可以改变public_html 到其他任何子目录下。

实际上，这个用户目录根本不一定需要位于用户主目录下里面。如果你没有一个密码文件，但你又想把一个用户名映射到公共的像/home一样目录的子目录里面，则可以使用HomesUserDatabase类。

directoryName=“public_html” homeBase=“/home”

userClass=“org.apache.catalina.startup.HomesUserDatabase”/>

这样一来，web文件就可以位于像/home/ian/public_html 或者 /home/jasonb/public_html一样的目录下。这种形式对Windows而言更加有利，你可以使用一个像c:\home这样的目录。

这些Listener元素，如果出现，则必须在Host元素里面，而不能在context元素里面，因为它们都用应用于Host本身。

使用CGI脚本

Tomcat主要是作为Servlet/JSP容器，但它也有许多传统web服务器的性能。支持通用网关接口（Common Gateway Interface，即CGI）就是其中之一，CGI提供一组方法在响应浏览器请求时运行一些扩展程序。CGI之所以被称为通用，是因为它能在大多数程序 或脚本中被调用，包括：Perl，Python，awk，Unix shell scripting等，甚至包括Java。不会把一个Java应用程序当作CGI来运行，毕竟这样太过原始。一般而言，开发Servlet总 要比CGI具有更好的效率，因为当用户点击一个链接或一个按钮时，不需要从操作系统层开始进行处理。

Tomcat包括一个可选的CGI Servlet，允许你运行遗留下来的CGI脚本。

为了使Tomcat能够运行CGI，必须做的几件事：

1. 把servlets-cgi.renametojar （在CATALINA_HOME/server/lib/目录下）改名为servlets-cgi.jar。处理CGI的servlet应该位于Tomcat的CLASSPATH下。

2. 在Tomcat的CATALINA_BASE/conf/web.xml 文件中，把关于 CGI的那段的注释去掉（默认情况下，该段位于第241行）。

3. 同样，在Tomcat的CATALINA_BASE/conf/web.xml文件中，把关于对CGI进行映射的那段的注释去掉（默认情况下，该

段位于第299行）。注意，这段内容指定了HTML链接到CGI脚本的访问方式。

4. 可以把CGI脚本放置在WEB-INF/cgi 目录下（注意，WEB-INF是一个安全的地方，你可以把一些不想被用户看见或基于安全考虑不想暴露的文件放在此处），或者也可以把CGI脚本放置在 context下的其他目录下，并为CGI Servlet调整cgiPathPrefix初始化参数。这就指定的CGI Servlet的实际位置，且不能与上一步指定的URL重名。

5. 重新启动Tomcat，你的CGI就可以运行了。

在Tomcat中，CGI程序缺省放置在WEB-INF/cgi目录下，正如前面所提示的那样，WEB-INF目录受保护的，通过客户端的浏览器无法窥探 到其中内容，所以对于放置含有密码或其他敏感信息的CGI脚本而言，这是一个非常好的地方。为了兼容其他服务器，尽管你也可以把CGI脚本保存在传统的 /cgi-bin目录，但要知道，在这些目录中的文件有可能被网上好奇的冲浪者看到。另外，在Unix中，请确定运行Tomcat的用户有执行CGI脚本 的权限。

改变编译器

在Tomcat 4.1（或更高版本，大概），JSP的编译由包含在Tomcat里面的Ant程序控制器直接执行。这听起来有一点点奇怪，但这正是Ant有意为之的一部 分，有一个API文档指导开发者在没有启动一个新的JVM的情况下，使用Ant。这是使用Ant进行Java开发的一大优势。另外，这也意味着你现在能够 在Ant中使用任何javac支持的编译方式，这里有一个关于Apache Ant使用手册的javac page列表。使用起来是容易的，因为你只需要在 元素中定义一个名字叫“compiler”，并且在value中有一个支持编译的编译器名字，示例如下：

```
jsp
org.apache.jasper.servlet.JspServlet
logVerbosityLevel
WARNING
compiler
jikes
3
```

当然，给出的编译器必须已经安装在你的系统中，并且CLASSPATH可能需要设置，那取决于你选择的是何种编译器。

限制主机访问

有时，可能想限制对Tomcat web应用的访问，比如，希望只有指定的主机或IP地址可以访问应用。这样一来，就只有那些指定的的客户端可以访问服务的内容了。为了实现这种效 果，Tomcat提供了两个参数供你配置：RemoteHostValve 和RemoteAddrValve。

通过配置这两个参数，可以让你过滤来自请求的主机或IP地址，并允许或拒绝哪些主机/IP。与之类似的，在Apache的httpd文件里有对每个目录的允许/拒绝指定。

可以把Admin Web application设置成只允许本地访问，设置如下：

```
allow=“127.0.0.1” deny=“” />
```

如果没有给出允许主机的指定，那么与拒绝主机匹配的主机就会被拒绝，除此之外的都是允许的。

目录结构

/bin：存放windows或Linux平台上启动和关闭Tomcat的脚本文件

/conf：存放Tomcat服务器的各种全局配置文件，其中最重要的是server.xml和web.xml

/doc：存放Tomcat文档

/server：包含三个子目录：classes、lib和webapps

/server/lib：存放Tomcat服务器所需的各种JAR文件

/server/webapps：存放Tomcat自带的两个WEB应用admin应用和 manager应用

/common/lib：存放Tomcat服务器以及所有web应用都可以访问的jar文件

/shared/lib：存放所有web应用都可以访问的jar文件（但是不能被Tomcat服务器访问）

/logs：存放Tomcat执行时的日志文件

/src：存放Tomcat的源代码

/webapps：Tomcat的主要Web发布目录，默认情况下把Web应用文件放于此目录

/work：存放JSP编译后产生的class文件

安全启动

编辑

Tomcat是一个世界上广泛使用的支持jsp和servlets的Web服务器。它在java上运行时能够很好地运行并支持Web应用部署。会因为设置不当，造成灾难性的后果。在Tomcat默认安装，Tomcat作为一个系统服务运行，如果没有将其作为系统服务运行，几乎所有Web服务器管理员都是缺省地将其以Administrator权限运行。这两种方式都允许Java运行时访问Windows系统下任意文件夹中的任何文件。缺省情况下，Java运行时授予安全权限。当 Tomcat以系统管理员身份或作为系统服务运行时，Java运行取得了系统用户或系统管理员所具有的全部权限。这样一来，Java运行时就取得了所有文件夹中所有文件的全部权限。并且Servlets(JSP在运行过程中要转换成Servlets)取得了同样的权限。所以Java代码可以调用Java SDK中的文件API、列出文件夹中的全部文件、删除任何文件，最大的危险在于以系统权限运行一个程序。当任一Servlets含有如下代码：

b4ae04fd6dYsJkr5 Runtime rt = Runtime.getRuntime();

rt.exec(”c:\SomeDirectory\SomeUnsafePRogram.exe”)

其服务是以system权限启动。根据权限最小安全原则，降低了脚本所获取的操作本地系统权限。此操作如下：

新建一个帐户

1. 用”ITOMCAT_计算机名”建立一个普通用户
2. 为其设置一个密码
3. 保证”密码永不过期”(PassWord Never Expires)被选中

修改Tomcat安装文件夹的访问权限

1. 选定环境参数CATALINA_HOME或TOMCAT_HOME指向的Tomcat安装文件夹。
2. 为”ITOMCAT_计算机名”用户赋予读、写、执行的访问权限。
3. 为”ITOMCAT_计算机名”用户赋予对WebApps文件夹的只读访问权限。
4. 如果某些Web应用程序需要写访问权限，单独为其授予对那个文件夹的写访问权限。

Tomcat作为系统服务

1. 到”控制面板”，选择”管理工具”，然后选择”服务”。
2. 找到Tomcat：比如Apache Tomcat.exe等等，打开其”属性”。
3. 选择其”登录”(Log)标签。
4. 选择”以…登录”(Log ON Using)选项。
5. 键入新建的”ITOMCAT_计算机名”用户作为用户名。
6. 输入密码。
7. 重启机器。

在DOS窗口下运行Tomcat步骤

1. 在”开始”按钮的”运行”框中键入CMD以打开一个DOS窗口。
2. 键入”RunAs /user：ITOMCAT_计算机名 CMD.exe”命令。
3. 在询问”ITOMCAT_计算机名”用户的密码时输入设置的密码。
4. 这将打开一个新的DOS窗口。
5. 在新开的DOS窗口中，转换到Tomcat的bin文件夹内。
6. 键入”catalina run”命令。
7. 关闭第一个DOS窗口。

设置一下程序

CMD.EXE NET.EXE ATTRIB.EXE At.EXE NET1.EXE FTP.EXE TELNET.EXE COMMAND. COM CAcls.EXEnetstat.exe;system 全部权限 ,其它用户无权限。

词条图册更多图册

Apache软件基金会

参考资料

- 1.Apache Tomcat. Apache [引用日期2016-01-15]

词条标签：

网站，科技产品，科学，技术，书籍