|  |
| --- |
| **tomcat6.0配置** |
| http://blog.chinaunix.net/templates/default/images/right_line.gif |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **第一步**：下载jdk和tomcat：[JDK下载](http://java.sun.com/)  [Tomcat下载](http://tomcat.apache.org/)     最新的jdk为1.6.10，tomcat为6.0，建议jdk1.4以上，tomcat4.0以上 **第二步**：安装和配置你的jdk和tomcat：执行jdk和tomcat的安装程序，然后设置按照路径进行安装即可。 1.安装jdk以后，需要配置一下环境变量，在我的电脑->属性->高级->环境变量->系统变量中添加以下环境变量(假定你的jdk安装在C:\Program Files\Java）：  JAVA\_HOME=C:\Program Files\Java\jdk1.6.0\_10  classpath=.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar;（.;一定不能少，因为它代表当前路径)  path=%JAVA\_HOME%\bin    接着可以写一个简单的java程序来测试JDK是否已安装成功：   |  | | --- | | public class Test{  public static void main(String args[]){  System.out.println("This is a test program.");  }  } |   将上面的这段程序保存为文件名为Test.java的文件。  然后打开命令提示符窗口，cd到你的Test.java所在目录，然后键入下面的命令   javac Test.java  java Test   此时如果看到打印出来This is a test program.的话说明安装成功了，如果没有打印出这句话，你需要仔细检查一下你的配置情况。   2.安装Tomcat后，在我的电脑->属性->高级->环境变量->系统变量中添加以下环境变量(假定你的tomcat安装在c:\tomcat）：  CATALINA\_HOME：c:\tomcat CATALINA\_BASE：c:\tomcat TOMCAT\_HOME: C:\Tomcat 然后修改环境变量中的classpath，把tomat安装目录下的common\lib下的servlet.jar追加到classpath中去，修改后的classpath如下：  classpath=.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar;%CATALINA\_HOME%\common\lib\servlet-api.jar;  *【注意最新版本的Tomcat中可能没有common这个文件夹了。所以servlet-api.jar应该的路径为*  *%CATALINA\_HOME%\lib\servlet-api.jar;请根据自己的情况自己修改！】*  接着可以启动tomcat，在IE中访问[http://localhost:8080](http://localhost:8080/)，如果看到tomcat的欢迎页面的话说明安装成功了。  **第三步**：建立自己的jsp app目录  1.到Tomcat的安装目录的webapps目录，可以看到ROOT，examples, tomcat-docs之类Tomcat自带的的目录；  2.在webapps目录下新建一个目录，起名叫myapp；  3.myapp下新建一个目录WEB-INF，注意，目录名称是区分大小写的；  4.WEB-INF下新建一个文件web.xml，内容如下：   |  | | --- | | <?xml version="1.0" encoding="ISO-8859-1"?>   <!DOCTYPE web-app  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"  "http://java.sun.com/dtd/web-app\_2\_3.dtd">   <web-app>  <display-name>My Web Application</display-name>  <description>  A application for test.  </description>  </web-app> |   5.在myapp下新建一个测试的jsp页面，文件名为index.jsp，文件内容如下：   |  | | --- | | <html> <body> <center>  Now time is: <%=new java.util.Date()%>  </center> </body> </html> |   6.重启Tomcat  7.打开浏览器，输入<http://localhost:8080/myapp/index.jsp> 看到当前时间的话说明就成功了。  **第四步**：建立自己的Servlet： 写入你的第一个Servlet: 在你新建的Application myapp/WEB-INF/classes/test目录下新建HelloWorld.java   |  | | --- | | package test;  import java.io.\*;  import javax.servlet.\*;  import javax.servlet.http.\*;  public class HelloWorld extends HttpServlet   {     public void doGet(HttpServletRequest request,HttpServletResponse response)throws ServletException,IOException    {     response.setContentType("text/html");      PrintWriter out = response.getWriter();     out.println("<html><head><title>");     out.println("This is my first Servlet");     out.println("</title></head><body>");     out.println("<h1>Hello,World!</h1>");     out.println("</body></html>");        }    } |   然后照样用javac HelloWorld.java来编译这个文件，如果出现无法import javax.servl et.\*    那么就是应该把C:\Tomcat\common\lib里面的servlet-api.jar文件拷贝到C:\JDK\jre\lib\ext中，再次编译，就没有问题了！    然后在Tomcat目录里面的C:\Tomcat\webapps\myapp里面按如下的文件结构：    myapp\index.jsp    myapp\WEB-INF\classes\test\HelloWorld.class(把上面生成的HelloWorld.class文件放在这个 里面)  然后在浏览器中输入<http://localhost:8080/myapp/HelloWorld>,于是Server众望所归的报错了:Error 404--Not Found   怎么回事呢？ Servlet必须使用C:\Tomcat\webapps\myapp\WEB-INF这个目录下面的web.xml文件进行注册， 用EditPlus打开这个web.xml文件， 在<web-app></web-app>添加下面这段程序:   |  | | --- | | <servlet>     <servlet-name>HelloWorld</servlet-name>     <servlet-class>test.HelloWorld</servlet-class>  </servlet>  <servlet-mapping>     <servlet-name>HelloWorld</servlet-name>     <url-pattern>/HelloWorld</url-pattern>    </servlet-mapping> |    为什么呢？  因为 这样的结构    <servlet>    <servlet-name>HelloWorld</servlet-name>    <servlet-class>test.HelloWorld</servlet-class>//类的路径    </servlet> 表示指定包含的servlet类. 而以下的结构    <servlet-mapping>      <servlet-name>HelloWorld</servlet-name>     <url-pattern>/HelloWorld</url-pattern>    </servlet-mapping>  表示指定HelloServlet应当映射到哪一种URL模式。  在修改web.xml完毕过后，重新启动Server，然后再输入<http://localhost:8080/myapp/HelloWorld>,，那么偌大一个Hello,World!等 着你呢，恭喜你!  **第五步**：建立自己java Bean  1. 在你新建的Application myapp/WEB-INF/classes/test目录下新建TestBean.java   |  | | --- | | package test; public class TestBean {     private String name =null;     public TestBean(String nameInit){         this.name = nameInit;     }     public void setName(String newName){         this.name=newName;     }     public String getName(){         return this.name;     } } |   然后照样用javac TestBean.java来编译这个文件。  2.然后在你新建的应用程序目录myapp下新建一个新的jsp文件：testBean.jsp   |  | | --- | | <%@ page import="test.TestBean" %> <html> <head> <title>Test Bean</title> </head> <body> <center> <%    TestBean testBean = new TestBean("Http://yexin218.cublog.cn");  %> Java Bean Test:     The author's blog address is<%=testBean.getName()%> </center> </body> </html> |   好了，确定各个文件的位置：   |  | | --- | | myapp\index.jsp myapp\testBean.jsp myapp\WEB-INF\web.xml myapp\WEB-INF\classes\test\HelloWorld.class myapp\WEB-INF\classes\test\TestBean.class |   3.重启Tomcat如果需要的话，在浏览器输入:[**http://localhost:8080/myapp/testBean.jsp**](http://localhost:8080/myapp/testBean.jsp)幸运的话就会看到:  **Java Bean Test: The author's blog address isHttp://yexin218.cublog.cn**  这样就完成了整个Tomcat下的jsp、servlet和javabean的配置。  **第六步：**配置虚拟目录  打开 Tomcat6.0\conf\server.xml 文件，在 <Host> 和 </Host> 之间加入   |  | | --- | | <Context path="/myapp" docBase="D:\myapp" debug="0" reloadable="true" crossContext="true" /> |   以上配置的源代码：   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **http://control.cublog.cn/fileicon/rar.gif** | |  |  | | --- | --- | | 文件: | myapp.rar | | 大小: | 3KB | | 下载: | [下载](http://blogimg.chinaunix.net/blog/upfile2/080411120833.rar) | |   附注：  2008年4月11日－－－－－－－－－全面整理  2008年9月6日--------------修改J2sdk为JDK，避免读者误解。  **Tomcat 6.0服务器server.xml配置文件**  使用Tomcat，如果对它的配置不能深入地理解，在实际工作中可能会遇到一些问题。  当安装完Tomcat 6.0以后，默认的conf/server.xml文件，去掉注释以后，如下所示：  <?xml version='1.0' encoding='utf-8'?> <Server port="8005" shutdown="SHUTDOWN"> <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" /> <Listener className="org.apache.catalina.core.JasperListener" /> <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener" /> <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" /> <GlobalNamingResources>     <Resource name="UserDatabase" auth="Container"               type="org.apache.catalina.UserDatabase"               description="User database that can be updated and saved"               factory="org.apache.catalina.users.MemoryUserDatabaseFactory"               pathname="conf/tomcat-users.xml" /> </GlobalNamingResources> <Service name="Catalina">     <Connector port="8085" protocol="HTTP/1.1"                 connectionTimeout="20000"                 redirectPort="8443" />     <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />     <Engine name="Catalina" defaultHost="localhost">       <Realm className="org.apache.catalina.realm.UserDatabaseRealm"              resourceName="UserDatabase"/>       <Host name="localhost" appBase="webapps"             unpackWARs="true" autoDeploy="true"             xmlValidation="false" xmlNamespaceAware="false">       </Host>     </Engine> </Service> </Server>  下面基于这样一个基础的配置并结合Tomcat的体系结构，来说明各个配置元素的含义。关于Tomcat的体系结构，可以参考文档docs/architecture/overview.html。  **<Server> 元素**  <Server>元素表示的是整个Catalina Servlet容器，在conf/server.xml中是最顶层的配置元素，也就是说，使用Tomcat WEB服务器是基于这样一个容器的，能够容纳各种资源、连接、虚拟主机等等。当然，如果你实现了org.apache.catalina.Server接口，也可以在server.xml中配置<Server>元素，是启动容器为自己实现的，这需要在<Server>元素中配置指定的属性。  实现org.apache.catalina.Server接口，必须支持该接口定义的三个属性：className、port、shutdown。  **className属性**  指明实现org.apache.catalina.Server接口的类，如果在<Server>元素中没有指定className的值，就会默认指定className="org.apache.catalina.core.StandardServer"，使用标准的实现类。  **port属性**  指定TCP/IP端口号，通过该端口号可以与Tomcat服务器进行通信，比如执行shutdown命令来关闭Tomcat服务器。  **shutdown属性**  通过为sutdown指定一个字符串，可以通过发送该字符串作为消息来关闭Tomcat服务器。  对于上面的配置文件中，<Server>元素配置如下：  <Server port="8005" shutdown="SHUTDOWN">  显然省略了className属性，使用默认的org.apache.catalina.core.StandardServer，端口号为8005，通过建立的TCP/IP连接，可以使用命令向端口8085发送消息，来影响Tomcat服务器的行为，例如关闭服务器。shutdown="SHUTDOWN"，如果Tomcat在启动以后，接收到端口8085发送的消息为SHUTDOWN，则开始执行关闭的动作。  **<Service>元素**  <Server>元素是<Service>元素的顶层配置，从<Server>元素的含义来看，<Service>元素表示存活于一个顶层的Catalina Servlet容器中的组件，其中，一个<Server>元素中可以配置一个或者多个<Service>元素。作为组件，应该通过某种方式来提供一定的服务。  其中，作为一个服务组件，必须实现org.apache.catalina.Service 接口。默认，Tomcat实现了一个标准的服务器组件，具体实现类为org.apache.catalina.core.StandardService。一个<Server>元素可以配置两个属性：  **className属性**  默认使用Tomcat实现了标准服务组件，当然可以定义自己的服务组件，并在<Server>元素中配置指定。  **name属性**  用来标识在<Server>元素中唯一的一个服务组件。  从上面的配置文件可以看到，<Server>元素的配置块：  <Service name="Catalina">     <Connector port="8085" protocol="HTTP/1.1"                 connectionTimeout="20000"                 redirectPort="8443" />     <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />     <Engine name="Catalina" defaultHost="localhost">       <Realm className="org.apache.catalina.realm.UserDatabaseRealm"              resourceName="UserDatabase"/>       <Host name="localhost" appBase="webapps"             unpackWARs="true" autoDeploy="true"             xmlValidation="false" xmlNamespaceAware="false">       </Host>     </Engine> </Service>  **<Connector>元素**  <Connector>元素表示连接器，是容器与外部进行交互信的桥梁。该元素定义的连接器是在一个<Service>元素表示的组件中，一个<Service>元素中可以配置多个连接器<Connector>元素，表示提供服务的组件通过不同的连接方式与容器外部进行通信，从而完成一定的操作。  通过上面配置可以看到，配置了两个连接器，一个是基于HTTP协议的连接器，一个是基于AJP协议的连接器。  上面<Connector>元素属性包括port、protocol、redirectPort、connectionTimeout，分别表示端口号、通信协议、重定向端口号、连接超时时间。  <Connector>元素可以配置的属性非常多，这是由连接的特点所决定的，可以对连接的状态进行详细设置。可以参考Tomcat文档docs/config/http.html。  **<Engine>元素**  <Engine>元素表示Catalina Servlet容器中针对</Service>元素定义的服务，执行请求处理的一个引擎。当然作为这样一个引擎，它可能将通过<Connector>元素配置的不同连接方式，接收容器外部请求的数据，经过处理，将处理后的数据，再通过连接器传送给容器外部执行请求的载体，例如用户。  可见，存在一个或者多个连接器绑定在一个引擎上，相互协调完成特定服务定义的功能。  <Engine>元素可以配置的属性有：  **className属性**  className属性指定引擎的实现类，可以实现自定义引擎，但是必须实现org.apache.catalina.Engine 接口，Tomcat同样实现了默认的标准引擎，通过org.apache.catalina.core.StandardEngine类定义的。  **defaultHost属性**  指定默认主机。在一个<Engine>元素中可以配置多个虚拟主机，每个虚拟主机都通过name属性来识别，但是在<Engine>元素中必须指定一个在其中存在的主机配置，通过defaultHost来默认指定。  **name属性**  用来标识该引擎的逻辑名称。  通过从上面的配置可以看到，name="Catalina"，defaultHost="localhost"，而且localhost已经在<Engine>元素中存在的<Host>元素中配置。没有指定className属性，表示默认使用org.apache.catalina.core.StandardEngine引擎。  **<Host>元素**  <Host>元素表示对虚拟主机进行配置，可以在一个<Engine>元素中配置多个<Host>元素，也就是指定多个虚拟主机。虚拟主机就是通过域名这样的逻辑名，来定位到物理存在的主机，或者主机上的目录。  同样，如果想自定义主机实现类，可以实现org.apache.catalina.Host 接口，否则使用默认的标准主机实现org.apache.catalina.core.StandardHost类。  上面对<Host>元素的配置如下：        <Host name="localhost" appBase="webapps"             unpackWARs="true" autoDeploy="true"             xmlValidation="false" xmlNamespaceAware="false">  属性说明如下：  **className属性**  指定主机实现类。这里使用默认的标准主机，可以省略。  **name属性**  虚拟主机名称，也就是一个逻辑域名标识符，例如localhost，如果连接器<Connector>配置元素指定了HTTP协议和8085端口，则可以通过浏览器使用前缀http://localhost:8085来进行通信；如果你将本地IP地址127.0.0.1绑定到任意一个域名，例如www.shirdrn.org上，可以通过浏览器使用前缀http://www.shirdrn.org:8080来与容器进行本地交互。  **appBase属性**  appBase属性表示应用程序的根目录，也就是通过虚拟主机可以访问的资源所在的基路径，例如，上面appBase="webapps"，因为使用Tomcat部署应用程序，默认部署到webapps目录下面，这种情况是，你只能将应用程序部署后存放到webapps目录下，其实通过<Host>元素的子元素<Context>元素的docBase来指定实际应用程序的位置，也就是在物理文件系统中的位置（不一定就是在webapps目录下）。  <Host>元素还有很多属性，可以参考docs/config/host.html。  **<Context>元素**  在<Host>元素中可以配置一个或者多个<Context>元素。如果配置多个的话，必须使得每个<Context>元素对应一个唯一的path。每个<Context>元素表示一个Web应用程序，可以在该元素中来对指定的Web应用程序详细配置。  例如，存在这样的一个主机配置中，包含了<Context>元素：  <Host name="beta.shirdrn.org" appBase="/usr/local/tomcat/webapps" unpackWARs="true" autoDeploy="true" xmlValidation="false" xmlNamespaceAware="false"> <Context path="/" docBase="/home/javaApp/myApps" workDir="/usr/local/tomcat/work" reloadable="true">     <Logger className="org.apache.catalina.logger.FileLogger"      directory="logs" prefix="appLog." suffix=".txt" timestamp="true" />     <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"      prefix="test\_access\_log." suffix=".txt" pattern="common" resolveHosts="false"/> </Context> </Host>  虚拟主机逻辑名称为beta.shirdrn.org，实际我的项目应用程序目录docBase为/home/javaApp/myApps，应用程序部署基路径appBase为/usr/local/tomcat/webapps，path设置为“/”表示前缀是“/”，从根目录请求，通过HTTP连接器，指定端口为8085，可以通过浏览器使用前缀http://beta.shirdrn.org:8085来访问虚拟主机上的资源。 | |