使用 Tomcat 和 MyEclipse 进行开发

主要内容：

Tomcat 的主要配置

Myeclipse开发工具的使用

在Myeclipse中配置Tomcat

Tomcat：

tomcat是一个实现了JavaEE 标准的最小的Web服务器，是Apche组织开发的、免费的服务器，可以直接在网络上下载。

最新版本是7.0X ，但是为了稳定本次使用 tomcat6.0讲解开发。

下载时有两种版本：

·手工安装版

·windows安装版

使用Windows安装版可以直接在服务中进行注册，通过服务器启动tomcat，而手工版必须单独配置服务

Tomcat 的安装

安装之前必须对本机的环境进行配置，在tomcat运行时必须使用JDK，本身必须要有对J2SE的支持。

配置过程：

在环境变量中配置JAVA\_HOME，值为本机JDK的安装路径

一般服务器都会在指定的端口上进行监听，默认的监听端口是8080，要注意一点，如果你本机安装了Oracle服务器的话，则这个端口可能会被占用。

安装好之后，tomcat目录下面有以下的文件件：

·bin：所有的可执行文件

·lib:里面存放的是\*。jar文件

·conf:配置文件，如果想要修改服务器，都需要在此文件夹中进行配置

·logs：日志文件，如果服务器出错，会自动记录到此文件夹 下

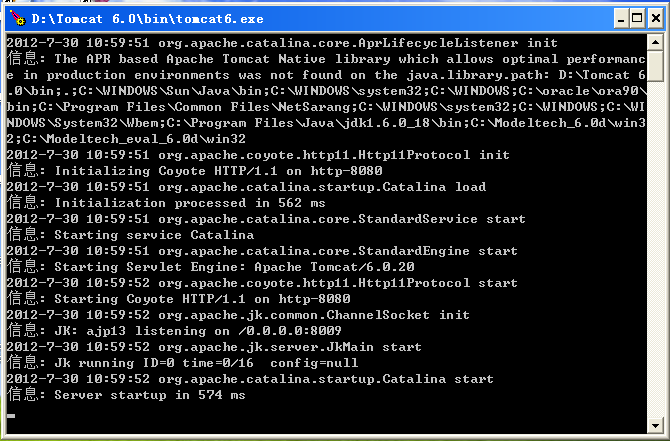
·webapps:所有的可执行的Web项目都会放在这里

·work：此文件夹中保存了所有的临时文件，窍门：如果在开发中，当一个程序无论怎么修改，都无法正确显示，此时就可以考虑删除work文件夹下面的所有内容，如果还没有正确显示，则考虑删除浏览器中的所有临时文件和缓存

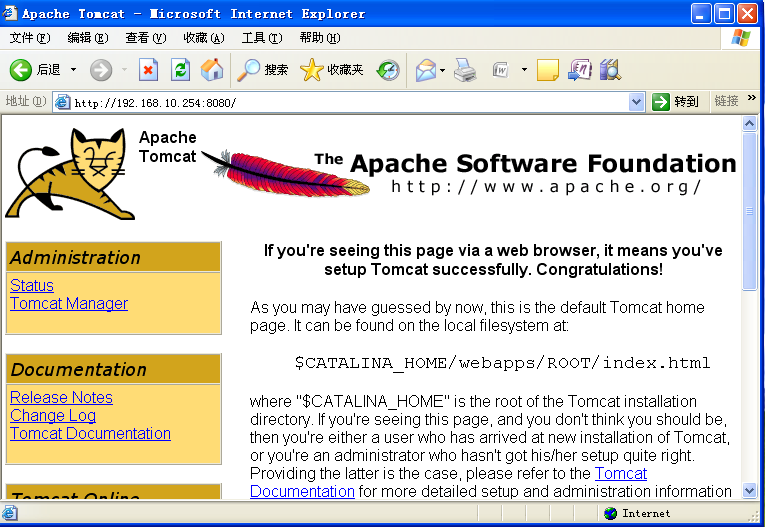
启动Tomcat：

直接使用bin路径下的tomcat6.exe即可

如果出现以下界面就表示服务器启动成功：



出现以下几面就表示服务器访问成功：



启动完成之后再浏览器中输入：<http://192.168.10.254:8080/>就可以访问tomcat。

如果是访问本机的tomcat，输入<http://localhost:8080/>

<http://www.wanczy.com/——>>通过域名访问，通过DNS映射到IP地址

配置Tomcat服务器：

配置服务器是每个开发人员都必须会的。

我们之前已经说过，tomcat所有的配置文件都放在conf文件夹下面，里面的server.xml是配置的核心文件

此时我想要修改服务器的监听端口，则可以在server.xml文件中进行配置，将端口改为80

|  |
| --- |
| <Connector port="80" protocol="HTTP/1.1"  connectionTimeout="20000"  redirectPort="8443" /> |

每次对tomcat服务器进行修改，都需要重新启动tomcat

配置虚拟目录：

tomcat中配置虚拟目录是必须会的，所有的开发都是在其他项目中完成的，如果想要通过tomcat访问项目，则需要配置虚拟目录，配置虚拟目录也有严格的要求，虚拟目录必须有以下目录结构：

|-Web ROOT 一般情况下虚拟目录配置的路径都在此文件夹中

|-WEB-INF

|-web.xml:工程的配置文件

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app version=*"2.5"*  xmlns=*"http://java.sun.com/xml/ns/javaee"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee*  *http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*>  <welcome-file-list>  <welcome-file>index.jsp</welcome-file>  </welcome-file-list>  <display-name>Welcome to Tomcat</display-name>  <description>Welcome to Tomcat</description>  </web-app> |

虽然 工程中的web.xml配置好了，但是服务器并不知道在此工程下面的web.xml文件有配置，所以要在tomcat中的conf下面的server.xml中配置此工程目录

修改server.xml文件，在文件中加入如下代码，但是，编写时，必须注意要求，必须在</host>之前配置

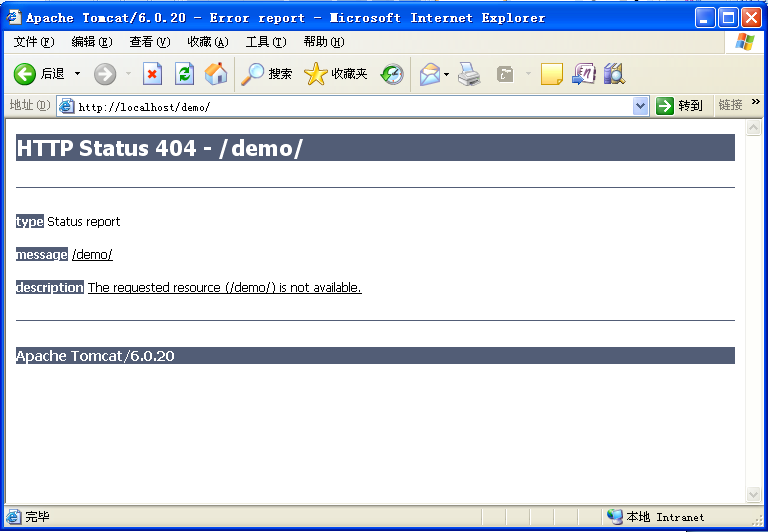
|  |
| --- |
| <Context path=”/demo” docBase=”工程目录”/> |

Context 表示上下文，表示配置一个新的上下文

path:表示在浏览器中输入的路径 必须要有“/”

docBase:表示的是此项目硬盘中的实际目录

通过“HTTP://localhost/demo” 就可以访问F:\dist文件夹中的内容，配置完成之后，必须重启服务器

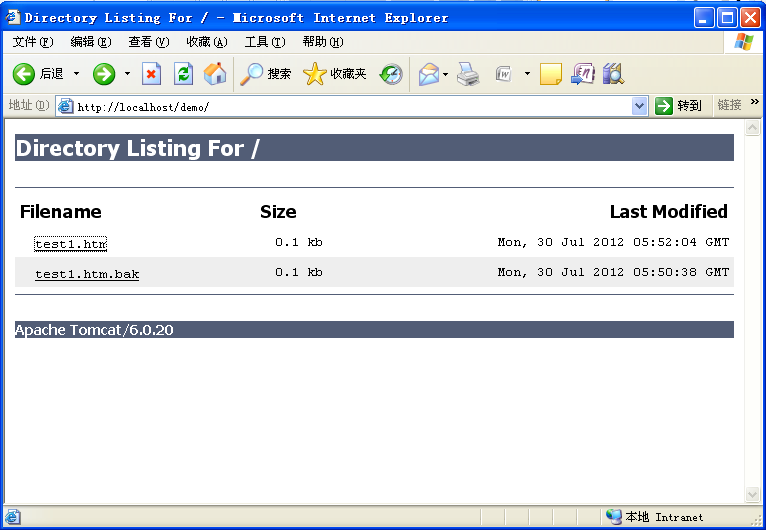


虚拟目录配置成功，但是访问时出现了404错误，404错误表示客户端错误，表示的是路径出错，之所以会有这样的错误，原因之一：工程里面没有index.jsp文件。如果里面放置index.jsp文件的话，会默认打开此文件，如果没有index.jsp文件，解决方法就是打开Tomcat中的功能列表，就是说在页面上列出所有的此文件夹下面的全部内容，需要修改 conf/web.xml文件：

|  |
| --- |
| <servlet>  <servlet-name>default</servlet-name>  <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>  <init-param>  <param-name>debug</param-name>  <param-value>0</param-value>  </init-param>  <init-param>  <param-name>listings</param-name>  <param-value>**true**</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet> |

但是有一点需要注意，在项目发布的时候，最好关闭列表功能，为了安全。

重启服务器之后再次访问，得到如下图所示的列表界面：



第一个程序：

还是由最经典的程序：“Hello World”开始

jsp = java+html ，就是在html的基础之上添加了java代码

|  |
| --- |
| <html>  <head>  <title>Hello World To JSP</title>    </head>  <body>  <%  out.print("Hello World!!");//向浏览器输出  %>  </body>  </html> |

运行程序，发现，第一次运行会比第二次慢，这是因为第一次执行的时候程序需要编译。编译成字节码文件。

tomcat的执行流程：

\*\_jsp.class

\*.jsp

浏览器

\*\_jsp.java

We容器

第一次访问时的执行流程。

·客户端通过网络连接到服务器，因为是动态请求，所以要交给Web容器处理

·在容器中找到\*。jsp文件，

·转换成java文件

·编译成class 文件

·执行形成的class 文件

第二次执行（第三次。。。。）：

·因为已经存在了class文件，就不需要再转化和编译了（文件内容没有改变）

·如果jsp文件内容有修改，第二次执行就要重新转换和编译，tomcat中有一个work文件夹，就是存放了这些转换和编译的临时文件。

我们在执行的时候发现out.println("Hello World!!");并没有换行，这里的换行只是针对源文件，会被html忽略掉，所以想要换行的话，可以直接编写html代码

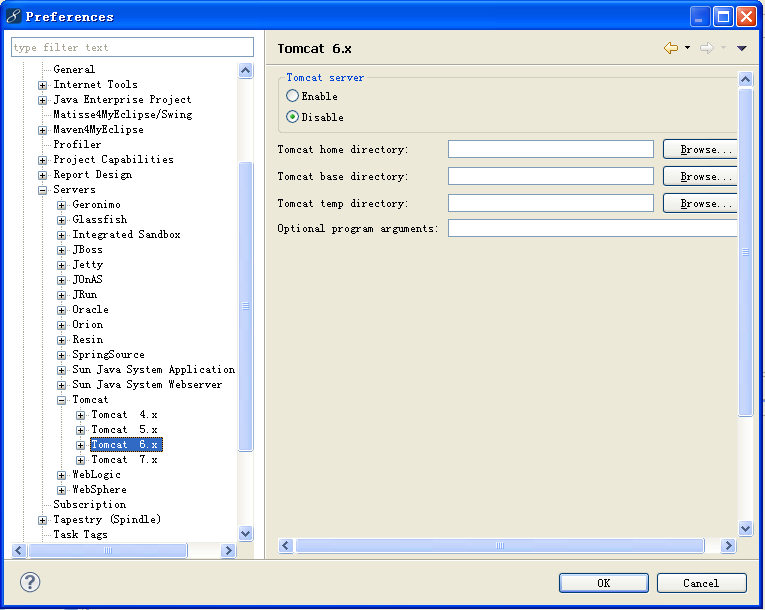
使用Myeclipse 进行WEB项目开发：

怎么在MyEclipse中运行WEB项目呢？

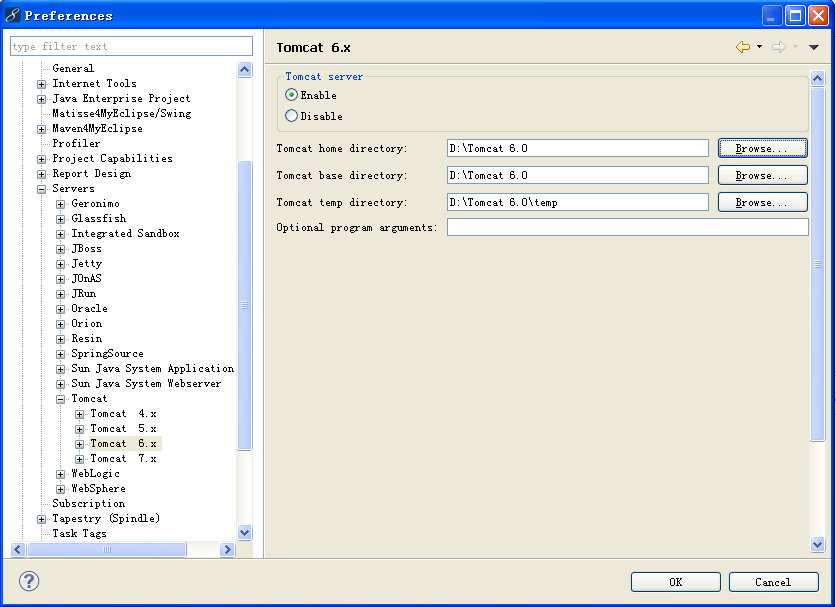
第一种方式：我们可以向最早的那种方式，通过配置虚拟路径来部署项目（手工部署），此种部署方式为推荐部署方式。

第二种方式：通过Myeclipse进行自动部署

·将tomcat配置到Myeclipse中



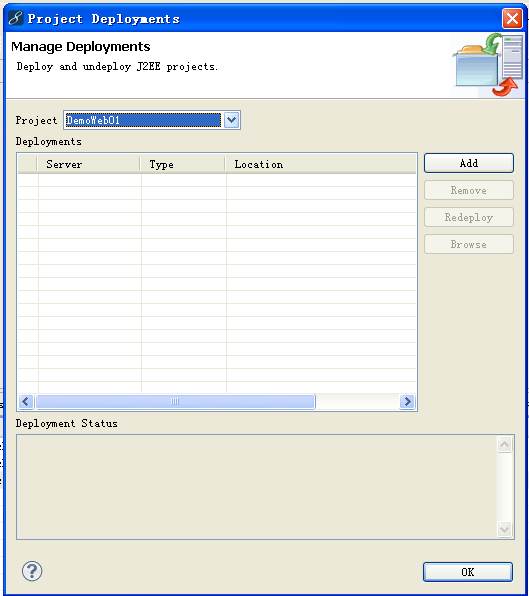
选择tomcat安装路径：



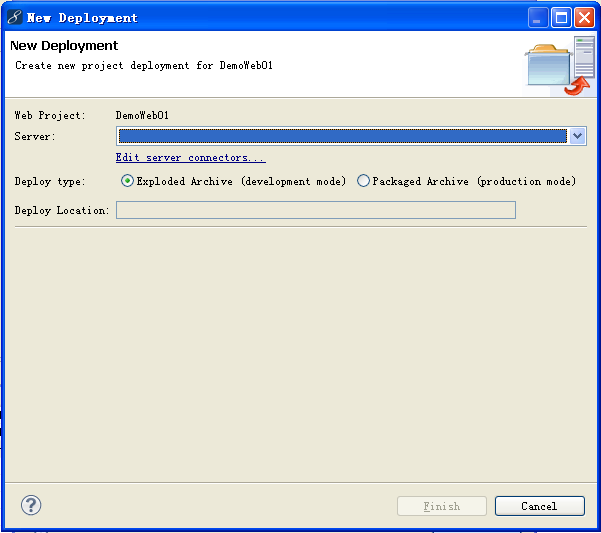
此时tomcat已经配置到myeclipse中，能够正常启动了，

那怎么去自动部署项目呢？

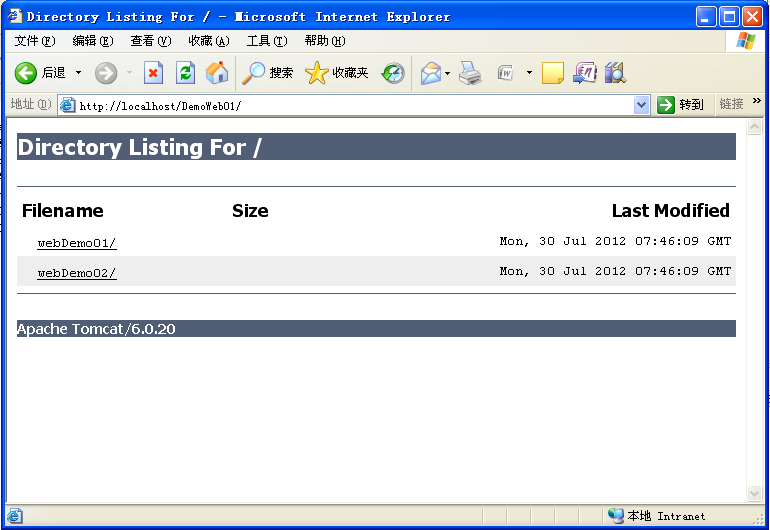
第一步：将项目加入服务器



选择你要加入的项目



将项目添加到指定的服务器中



tomcat启动之后，通过工程名称访问<http://localhost/>工程名称

在开发过程中，最好不要使用Myeclipse自带的服务器和浏览器，因为本身可能有问题，Myeclipse最作为开发用的工具，部署时最好使用 手动部署方式，即配置虚拟目录

第一个交互程序：

现在想要实现程序的交互，要依赖于表单，所以我们演示一个表单提交的交互程序，用户输入内容，然后服务器显示内容

*MyHtml01。html*

|  |
| --- |
| <html>  <head>  <title>zhongwen</title>  </head>  <body>  <form method=*"post"* action=*"MyHtml01.jsp"*>  输入信息：<input type=*"text"* name=*"input"* size=*"20"*/>  <input type=*"submit"* value=*"提交"* >  </form>  </body>  </html> |

*在MyHtml01。jsp* 显示表单输入的信息 通过方法 request.getParamete(“参数名称”) 取得表单页面提交的String类型的值

|  |
| --- |
| <%  String str = request.getParameter("input");  out.print("<h2>"+str+"</h2>");  %> |

这是我们的第一个交互程序，表单页面输入了什么内容，jsp就显示什么内容。

现在对于这个交互程序，为了保证前台输入的页面不能为空，就可以加入一些javascript的判断

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <title>zhongwen</title>  </head>  <body>  <form method=*"post"* action=*"MyHtml01.jsp"* onsubmit=*"return validate(this)"*>  输入信息：<input type=*"text"* name=*"input"* size=*"20"*/>  <input type=*"submit"* value=*"提交"* >  </form>  </body>  <script type=*"text/javascript"*>  **function** **validate**(f){  **var** s = f.input.value;  **if**(s=="" || s==**null**){  alert("输入信息不能为空");  f.input.focus();  **return** **false**;  }  **return** **true**;  }    </script>  </html> |