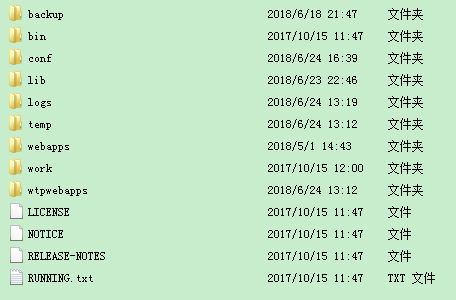
Tomcat

简介：

# 安装完成后的文件解读



bin：

存放启动和关闭tomcat的脚本。

conf：

存放Tomcat的各种配置文件，其中比较主要的是server.xml和web.xml

server.xml：Tomcat的全局配置文件，是配置的核心文件

web.xml：在Tomcat中配置不同的关系环境

lib：

存放tomcat服务器运行时使用的jar文件

logs：

存放Tomcat每次运行后产生的日志文件

temp：

存放应用运行过程中产生的临时文件

webapps：存放应用程序示例，web发布时发布到这个目录下

wtpwebapps：之前是在webapps下，随着tomcat的升级改到了这个目录下，这个目录就是只有用户的发布文件了，而不再有示例文件了。使用eclipse发布时，可以执行更改的

work：

存放有JSP生成的Servlet源文件和字节码文件，这个是由tomcat自动生成 的。

## 访问端口配置

默认是使用的8080 这里我们是可以修改的，同时也可以再添加一个端口

<Connector URIEncoding="UTF-8" connectionTimeout="20000" port="8088" protocol="HTTP/1.1" redirectPort="8443"/>

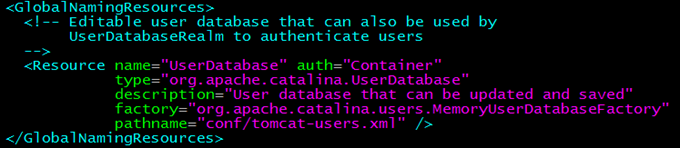
## 用户配置

Tomcat7以后的版本中

$CATALINA\_HOME/conf/tomcat-user.xml(如果更改了tomcat-user.xml文件但没有重启tomcat，则除非重启tomcat，否则tomcat不会重启读入该文件)

tomcat-user.xml文件是使用该领域的关键。它包含一份可访问web应用程序的用户清单。该文件是一个简单的xml文件，根元素是tomcat-users，且只能使用role和user元素。  
每个role元素只有一个属性：rolename。  
而每个user元素则有3个属性：username、password及roles

**默认是读取tomcat-users.xml**（可通过Resource中的pathname属性配置为其它文件）里面配置的用户角色信息。



在tomcat-users.xml中 多了一些属性

<tomcat-users version="1.0" xmlns="http://tomcat.apache.org/xml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd">

这里面的属性是可以不写的 但是不能没有这个

新增用户和角色时的配置

<role rolename="tomcat"/>

<role rolename="role1"/>

<user username="tomcat" password="<must-be-changed>" roles="tomcat"/>

<user username="both" password="<must-be-changed>" roles="tomcat,role1"/>

<user username="role1" password="<must-be-changed>" roles="role1"/>

实际上，Tomcat已经为我们定义了4种不同的角色——也就是4个rolename，我们只需要使用Tomcat为我们定义的这几种角色就足够满足我们的工作需要了。

以下是Tomcat Manager 4种角色的大致介绍(下面URL中的\*为通配符)：

manager-gui

允许访问html接口(即URL路径为/manager/html/\*)

manager-script

允许访问纯文本接口(即URL路径为/manager/text/\*)

manager-jmx

允许访问JMX代理接口(即URL路径为/manager/jmxproxy/\*)

manager-status

允许访问Tomcat只读状态页面(即URL路径为/manager/status/\*)

admin-gui 允许访问html接口

admin-script 允许访问纯文本接口

从Tomcat Manager内部配置文件中可以得知，manager-gui、manager-script、manager-jmx均具备manager-status的权限，也就是说，manager-gui、manager-script、manager-jmx三种角色权限无需再额外添加manager-status权限，即可直接访问路径/manager/status/\*。

除了密码限制之外，还可以通过添加或来限制远程IP地址或主机对Manager Web应用程序的访问权限，特别是在公网上的主机

# vim /usr/local/tomcat/webapps/manager/META-INF/context.xml

# vim /usr/local/tomcat/webapps/host-manager/META-INF/context.xml

添加允许访问的IP范围，否则只能本机访问

<Context antiResourceLocking="false" privileged="true" >

<Valve className="org.apache.catalina.valves.RemoteAddrValve"

allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1|192\.168\.10\.\d+" />

<Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number|String)|org\.apache\.catalina\.f

ilters\.CsrfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>

</Context>

增加了**“用户组”**的概念，即用户除了有所属角色外，还可以有所属**“用户组”，用户组可以关联其它多个角色，例如它可以使用下面的tomcat-users.xml配置：**

  <role rolename="tomcat"/>

  <role rolename="admin"/>

  <user username="tomcat" password="tomcat" roles="tomcat"/>

  <group groupname="one"  roles="tomcat"/>

  <user username="admin" password="admin" roles="admin" groups="one"/>

**admin用户由于关联了“one”这个用户组，“one”用户组包含角色“tomcat”，所以admin用户就拥有了“tomcat”角色。**

## 部署web项目（webapps|| wtpwebapps）

三种方法：直接拖过来、修改server.xml进行部署、创建配置文件进行部署

### 直接拖放

直接拿过来是可以的，也可以将web工程进行打包，之后放到这个部署文件的位置，tomcat是会自动解压的。

### 修改server.xml进行配置

<Context docBase=" " path=" " reloadable=" " source=" "/>

参数说明：

path是虚拟路径；

docBase 是应用程序的物理路径；

workDir 是这个应用的工作目录，存放运行时生成的与这个应用相关的文件；

debug 则是设定debug level, 0表示提供最少的信息，9表示提供最多的信息

privileged设置为true的时候，才允许Tomcat的Web应用使用容器内的Servlet

reloadable 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，可以在不重起tomcat的情况下，自动装载改变后的应用程序，实现热部署

antiResourceLocking和antiJARLocking 热部署是需要配置的参数，默认false避免更新了某个webapp，有时候Tomcat并不能把旧的webapp完全删除，通常会留下WEB-INF/lib下的某个jar包，必须关闭Tomcat才能删除，这就导致自动部署失败。设置为true，Tomcat在运行对应的webapp时，会把相应的源文件和jar文件复制到一个临时目录里。

### 创建一个Context文件单独设置一个应用

在conf目录中，新建 Catalina＼localhost目录，在该目录中新建一个xml文件，名字可以随意取，要和path后的那个名字一致，按照下边这个path的配置，xml的名字应该就应该是vc（vc.xml），该xml文件的内容为如下：

<Context path="/vc" docBase="D:\Dev\appliction\WebRoot" debug="0" privileged="true" reloadable="true"></Context>

文件内容其实和上面第二点其实是一样的，只是单独以一个xml文件形式部署web应用，一个文件表示一个web应用，文件名是虚拟路径。

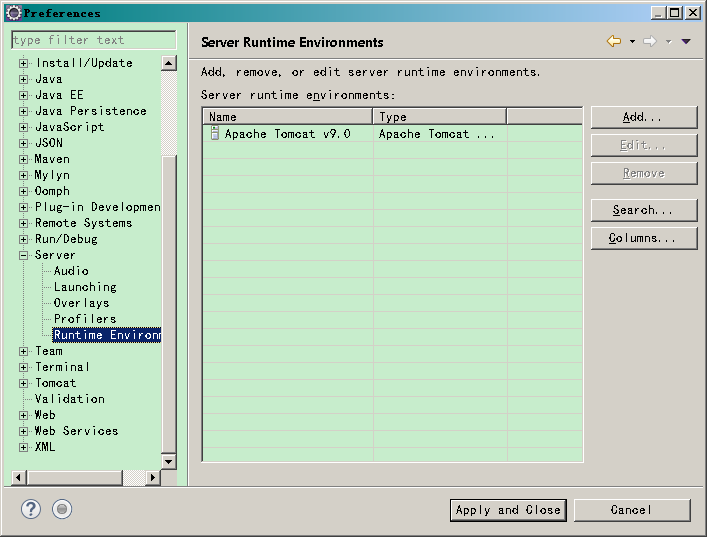
想两个都用时就不可以任意了，一般只有一个单独的工程时是可以的，多个工程就不可以

[**Setting property 'source' to 'org.eclipse.jst.jee.server的问题**](https://blog.csdn.net/z69183787/article/details/19911935)

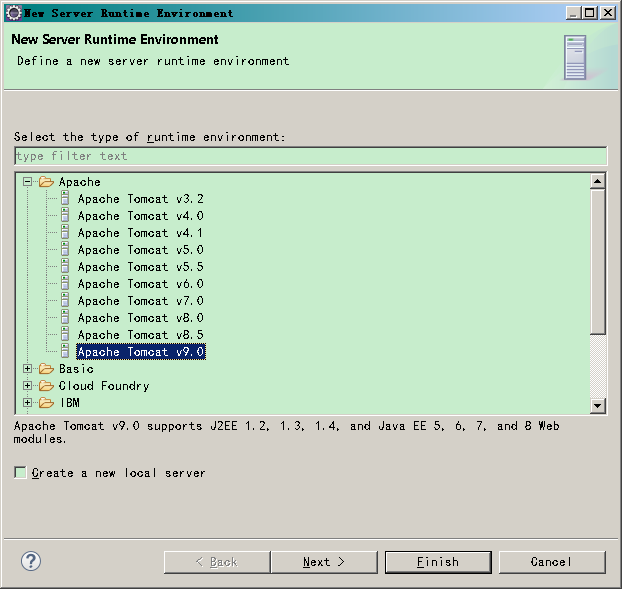
# Eclipse中集成tomcat

配置的流程示意图：

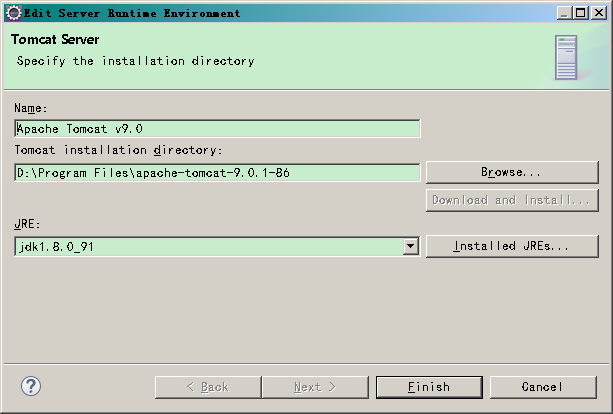
Window--->preference--->



Add-🡪

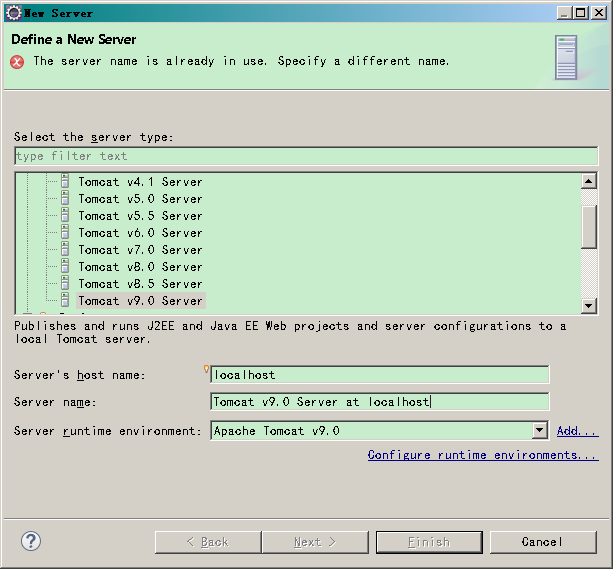


这里我们选择最新的

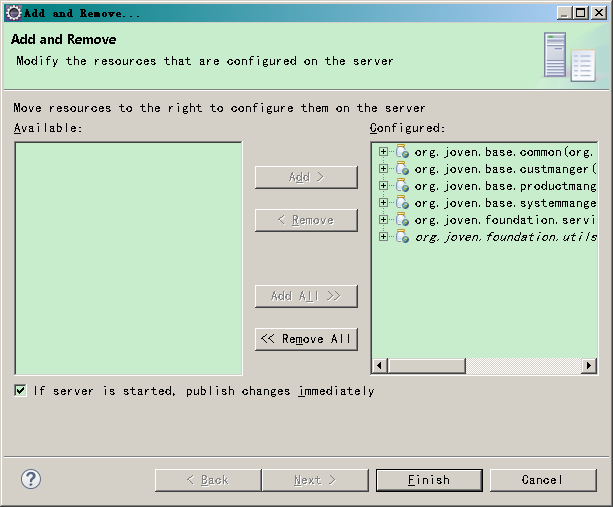


这是就要选择咱们安装的tomcat的路径和Jdk的路径了

接下来就是将我们的web工程部署到tomcat了



完成选择后 点击 next ，若是点错了变成finish后，可以选择tomcat后右键-🡪选择 add and remove 也是可以弹出下面的界面的。



# Log4j

tomcat日志 tomcat日志配置

在server.xml里的<host>标签下加上

<Valve className="org.apache.catalina.valves.AccessLogValve"

directory="logs" prefix="localhost\_access\_log." suffix=".txt"

pattern="common" resolveHosts="false"/>

|  |  |
| --- | --- |
| className | 官方文档上说了This MUST be set to org.apache.catalina.valves.AccessLogValve to use the default access log valve. &<60; 想配置访问日志？这就必须得写成这样。 |
| Directory | 这个东西是日志文件放置的目录，在tomcat下面有个logs文件夹，那里面是专门放置日志文件的，当然你也可以修改，我就给改成了/opt (E:\) |
| prefix | 这个是日志文件的名称前缀，我的日志名称为localhost\_access\_log.2017-11-06.txt，前面的前缀就是这个localhost\_access\_log |
| suffix | 这就是后缀名啦，可以改成别的 |
| pattern | 这个是最主要的参数了，具体的咱们下面讲，这个参数的内容比较丰富。 |
| resolveHosts | 如果这个值是true的话，tomcat会将这个服务器IP地址通过DNS转换为主机名，如果是false，就直接写服务器IP地址啦 |

|  |  |
| --- | --- |
| rotatable | 缺省值为true，默认的设置使得你的tomcat生成的文件命为prefix（前缀）+.+时间（一般是按天算）+.+suffix（后缀），参照我的日志名就知道了：localhost\_access\_log.2017-11-06.txt  使用这个需要谨慎，因为你将其设置为false的话，tomcat会忽略时间，不会新生成文件，最后导致你的文件超级大，这样生成的文件名就是：localhost\_access\_log.txt |

pattern可以设置成两种方式，第一种是pattern="common"，第二种是pattern="combined",这就可以控制日志里面的格式，各位说了，pattern就这两种？common和combined又是什么意思，具体是什么格式呢？

咱们一点一点看，其实pattern是可以设置的，common和combined只是集成了一些显示方式，就是将显示方式给组合了，pattern的实际值有如下几种，都是后面一个字母，前面一个%百分号

common的值：%h %l %u %t %r %s %b

combined的值：%h %l %u %t %r %s %b %{Referer}i %{User-Agent}i

至于combined的值的最后两个：

    %{Referer}i：从那个页面链接跳转到的此页面  
    %{User-agent}i：用户的User-Agent

|  |  |
| --- | --- |
| %a | 这是记录访问者的IP，在日志里是127.0.0.1 |
| %A | 这是记录本地服务器的IP，在日志里是192.168.254.108 |
| %b | 这是发送信息的字节数，不涵括http头，如果字节数为0的话，显示为- |
| %B | 看tomcat的解释，没看出来与b%的区别，但我这里显示为-1，没想明白，望知道者告知，我把官方解释贴出来吧 Bytes sent, excluding HTTP headers(发送的字节数，不包括HTTP头) |
| %h | 远端主机名(如果resolveHost=false，远端的IP地址） |
| %H | 访问者使用的协议，这里是HTTP/1.1 |
| %l | 官方说这个always return '-' 官方解释：Remote logical username from identd (可能这样翻译：记录浏览者进行身份验证时提供的名字){从identd返回的远端逻辑用户名（总是返回 '-'）}; |
| %m | 访问的方式，是GET还是POST，我这是GET |
| %p | 收到请求的本地端口号，这里的是80。 |
| %q | 查询字符串(如果存在，以 '?'开始)。   比如你访问的是aaa.jsp?bbb=ccc，那么这里就显示?bbb=ccc，明白了吧，这个q是querystring的意思 |
| %r | 官方解释：First line of the request (method and request URI)——请求的第一行，包含了请求的方法和URI |
| %s | 这个是响应http的状态码，这里返回的是304，咱们经常看见访问某个网页报错误500什么的，那也会返回500 |
| %S | 用户的session ID,这个session ID大家可以另外查一下详细的解释，反正每次都会生成不同的session ID |
| %t | 日志和时间，使用通常的Log格式 |
| %u | 认证以后的远端用户（如果存在的话，否则为'-'） |
| %U | 请求访问的URL地址，我这里是/seces/commonService?wsdl |
| %v | 本地服务器名称，可能就是你url里面写的那个吧，我这里是localhost |
| %D | 官方解释：Time taken to process the request, in millis，处理请求的时间，以毫秒为单位 |
| %T | 官方解释：Time taken to process the request, in seconds，处理请求的时间，以秒为单位 |