使用maven的自动部署功能可以很方便的将maven工程自动部署到远程tomcat服务器，节省了大量时间。

本文章适用于tomcat的7.x ,8.x, 9.x版本。

下面是自动部的步骤

**1，首先，配置tomcat的manager**

编辑远程tomcat服务器下的conf/tomcat-users.xml,在末尾增加（其实只要拉到文件末尾，去掉注释改一下就可以了）

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<user username="admin" password="password" roles="manager-script"/>

<user username="root" password="password" roles="manager-gui"/>

将上面的password改为自己的密码，注意对于tomcat9来说，不能同时赋予用户manager-script和manager-gui角色。

保存tomcat-users.xml。

在tomcat服务器的conf/Catalina/localhost/目录下创建一个manager.xml文件，写入如下值:

<?xml version="1.0" encoding="UTF-8"?>

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

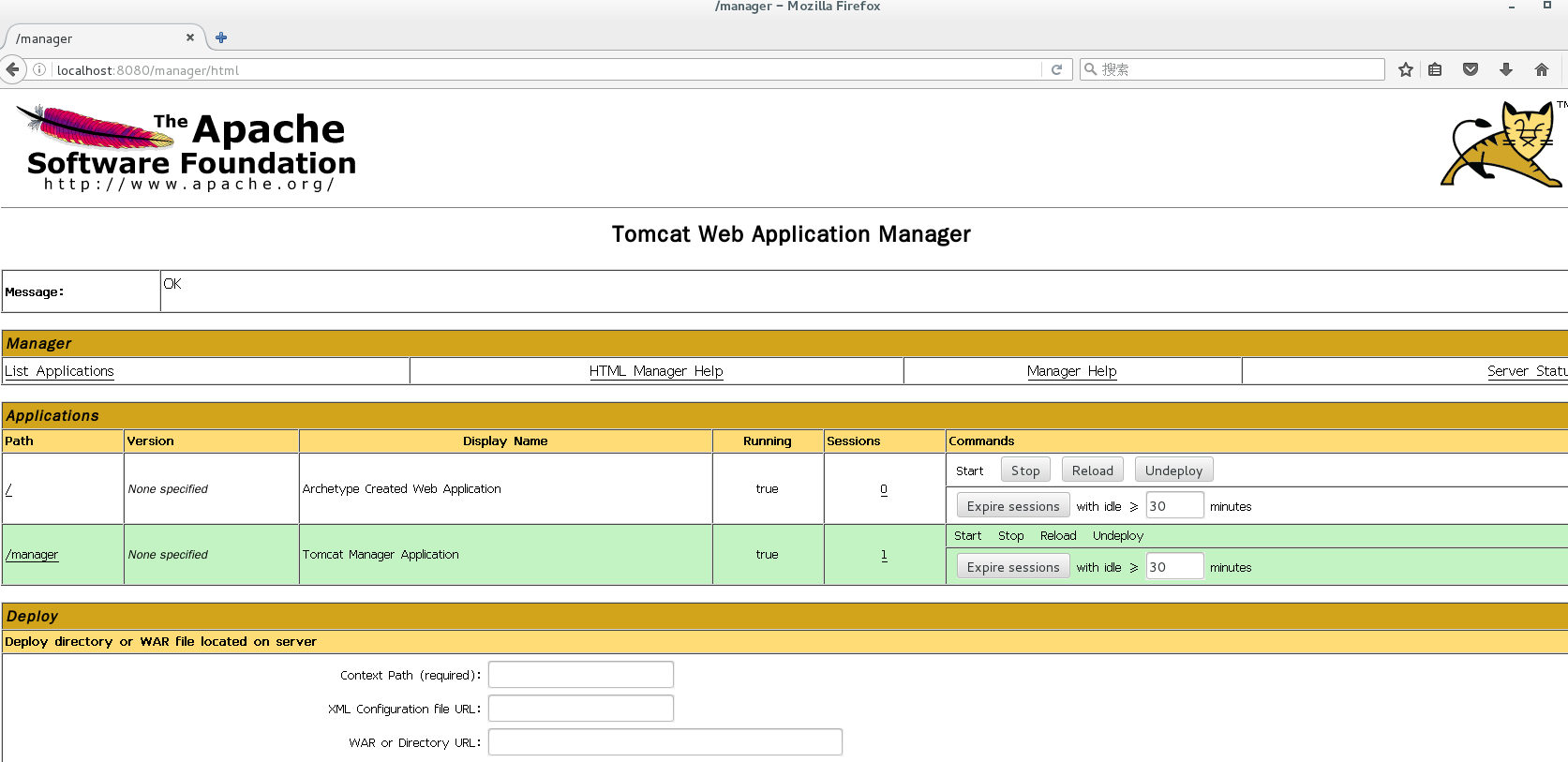
docBase="${catalina.home}/webapps/manager">

<Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="^.\*$" />

</Context>

保存退出。

然后在浏览器中输入http://serverip:port/manager/html，此时会弹出要求输入用户名和密码对话框，输入manager-gui对应的用户和密码登录管理控制台(其中serverip为服务器ip，如果服务器在本地就是localhost或者127.0.0.1，端口为tomcat端口，默认8080)。以此确认manager是否配置正确。正确结果示例如下：



**2，在maven项目中添加配置**

在pom.xml文件中，在plugins节点下添加如下plugin节点

[复制代码](javascript:void(0);)

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

<configuration>

<url>http://serverip:port/manager/text</url>

<username>admin</username>

<password>password</password>

<update>true</update>

<path>/webapp</path>

</configuration>

</plugin>

[复制代码](javascript:void(0);)

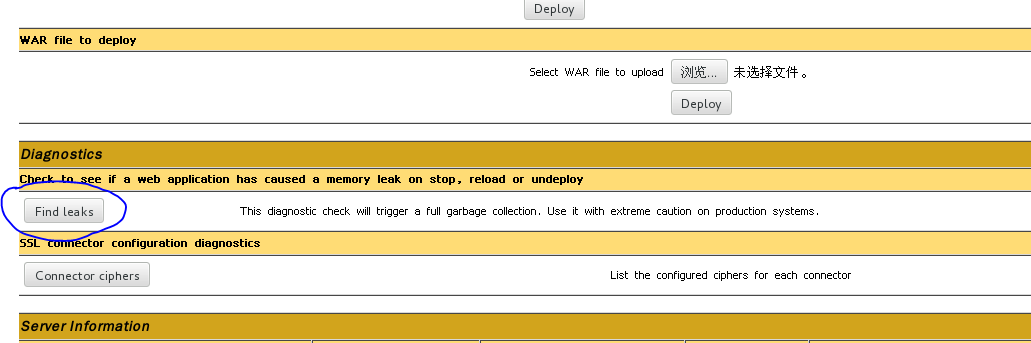
将上面的serverip和port换成自己tomcat服务器的ip和端口。密码换成上面配置的manager-script角色的密码。path改为**项目在tomcat服务器中的部署路径。**

然后进行部署，如果是第一次部署，运行mvn tomcat7:deploy进行自动部署(对于tomcat8,9，也是使用tomcat7命令)，如果是更新了代码后重新部署更新，运行mvn tomcat7:redeploy，如果第一次部署使用mvn tomcat7:redeploy，则只会执行上传war文件，服务器不会自动解压部署。**如果路径在tomcat服务器中已存在并且使用mvn tomcat7:deploy命令的话，上面的配置中一定要配置<update>true</update>，不然会报错。**

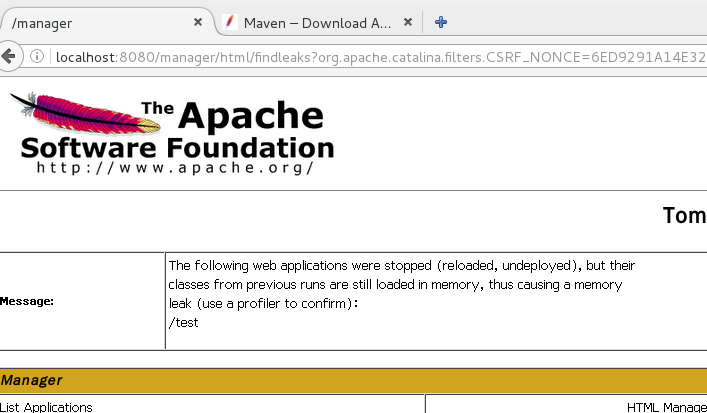
如果IDE是eclipse,就在runas->run configurations中配置一个maven build，intellij类似。

**3. 内存泄漏**

 使用上面的方法进行部署后会出现严重的内存泄漏现象。tomcat的manager提供了诊断在部署时是否产生内存泄漏的功能，在上面提到的http://serverip:port/manager/html这个页面底部有一个“Find leaks”的按钮，如下：



点击按钮，网页头部出现如下信息说明在部署的时候有内存泄漏：



上面的消息显示部署的test项目存在内存泄漏，如果同一项目多次重新部署，则一个项目名可能会出现多次。

部署时产生内存泄漏的原因是每次（重新）部署时，Tomcat会为项目新建一个类加载器，而旧的类加载器没有被GC回收。maven的库classloader-leak-prevention-servlet可以用来解决这个问题。具体方案为：

（1）添加maven依赖：

<dependency>

<groupId>se.jiderhamn.classloader-leak-prevention</groupId>

<artifactId>classloader-leak-prevention-servlet</artifactId>

<version>2.1.0</version>

</dependency>

（2）在项目的web.xml中添加一个Listener（**必须让此Listener成为web.xml中的第一个Listener，否则不起作用**）

<listener>

<listener-class>se.jiderhamn.classloader.leak.prevention.ClassLoaderLeakPreventorListener</listener-class>

</listener>

这样部署时的内存泄漏就解决了。

注意：

1） 添加这个Listener后，默认在tomcat关闭5s后jvm会进行内存回收的操作，具体时间设置可在下面的第三个参考链接中找到，所以，在关闭后的5s内，再次启动tomcat，可能会存在问题，导致启动无效（如果出现tomcat重启后日志显示正常但是服务器不工作的话考虑一下是不是这个问题）。

2）这个Listener只解决部署的内存泄漏，其他问题（如jdbc等）产生的内存泄漏还需要自己解决。

参考：

http://stackoverflow.com/questions/7788280/memory-leak-when-redeploying-application-in-tomcat#answer-36295683

http://java.jiderhamn.se/2011/12/11/classloader-leaks-i-how-to-find-classloader-leaks-with-eclipse-memory-analyser-mat/

https://github.com/mjiderhamn/classloader-leak-prevention

**4，错误排除。**

(1) 执行tomcat7:deploy显示Build Success成功但是没有效果，也没有在本地生成war包，检查一下maven配置文件中packaging标签是否设置为war。即：

<packaging>war</packaging>

如果不是（比如说是pom），那么改成war应该就可以了。

(2) 如果出现在本地tomcat服务器自动部署没有任何问题，部署到远程服务器出现下面的Cannot invoke Tomcat manager: Connection reset by peer: socket write error 错误：

[复制代码](javascript:void(0);)

[ERROR] Failed to execute goal org.apache.tomcat.maven:tomcat7-maven-plugin:2.2:deploy (default-cli) on project webapp: Cannot invoke Tomcat manager: Connection reset by peer: socket write error -> [Help 1]

org.apache.maven.lifecycle.LifecycleExecutionException: Failed to execute goal org.apache.tomcat.maven:tomcat7-maven-plugin:2.2:deploy (default-cli) on project clyf\_wechat: Cannot invoke Tomcat manager

at org.apache.maven.lifecycle.internal.MojoExecutor.execute(MojoExecutor.java:212)

at org.apache.maven.lifecycle.internal.MojoExecutor.execute(MojoExecutor.java:153)

at org.apache.maven.lifecycle.internal.MojoExecutor.execute(MojoExecutor.java:145)

at org.apache.maven.lifecycle.internal.LifecycleModuleBuilder.buildProject(LifecycleModuleBuilder.java:116)

at org.apache.maven.lifecycle.internal.LifecycleModuleBuilder.buildProject(LifecycleModuleBuilder.java:80)

at org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder.build(SingleThreadedBuilder.java:51)

at org.apache.maven.lifecycle.internal.LifecycleStarter.execute(LifecycleStarter.java:128)

at org.apache.maven.DefaultMaven.doExecute(DefaultMaven.java:307)

at org.apache.maven.DefaultMaven.doExecute(DefaultMaven.java:193)

at org.apache.maven.DefaultMaven.execute(DefaultMaven.java:106)

at org.apache.maven.cli.MavenCli.execute(MavenCli.java:863)

at org.apache.maven.cli.MavenCli.doMain(MavenCli.java:288)

at org.apache.maven.cli.MavenCli.main(MavenCli.java:199)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:497)

at org.codehaus.plexus.classworlds.launcher.Launcher.launchEnhanced(Launcher.java:289)

at org.codehaus.plexus.classworlds.launcher.Launcher.launch(Launcher.java:229)

at org.codehaus.plexus.classworlds.launcher.Launcher.mainWithExitCode(Launcher.java:415)

at org.codehaus.plexus.classworlds.launcher.Launcher.main(Launcher.java:356)

Caused by: org.apache.maven.plugin.MojoExecutionException: Cannot invoke Tomcat manager

at org.apache.tomcat.maven.plugin.tomcat7.AbstractCatalinaMojo.execute(AbstractCatalinaMojo.java:141)

at org.apache.tomcat.maven.plugin.tomcat7.AbstractWarCatalinaMojo.execute(AbstractWarCatalinaMojo.java:68)

at org.apache.maven.plugin.DefaultBuildPluginManager.executeMojo(DefaultBuildPluginManager.java:134)

at org.apache.maven.lifecycle.internal.MojoExecutor.execute(MojoExecutor.java:207)

... 20 more

Caused by: java.net.SocketException: Connection reset by peer: socket write error

at java.net.SocketOutputStream.socketWrite0(Native Method)

at java.net.SocketOutputStream.socketWrite(SocketOutputStream.java:109)

at java.net.SocketOutputStream.write(SocketOutputStream.java:153)

at org.apache.http.impl.io.AbstractSessionOutputBuffer.write(AbstractSessionOutputBuffer.java:181)

at org.apache.http.impl.io.ContentLengthOutputStream.write(ContentLengthOutputStream.java:115)

at org.apache.tomcat.maven.common.deployer.TomcatManager$RequestEntityImplementation.writeTo(TomcatManager.java:880)

at org.apache.http.entity.HttpEntityWrapper.writeTo(HttpEntityWrapper.java:89)

at org.apache.http.impl.client.EntityEnclosingRequestWrapper$EntityWrapper.writeTo(EntityEnclosingRequestWrapper.java:108)

at org.apache.http.impl.entity.EntitySerializer.serialize(EntitySerializer.java:117)

at org.apache.http.impl.AbstractHttpClientConnection.sendRequestEntity(AbstractHttpClientConnection.java:265)

at org.apache.http.impl.conn.ManagedClientConnectionImpl.sendRequestEntity(ManagedClientConnectionImpl.java:203)

at org.apache.http.protocol.HttpRequestExecutor.doSendRequest(HttpRequestExecutor.java:236)

at org.apache.http.protocol.HttpRequestExecutor.execute(HttpRequestExecutor.java:121)

at org.apache.http.impl.client.DefaultRequestDirector.tryExecute(DefaultRequestDirector.java:682)

at org.apache.http.impl.client.DefaultRequestDirector.execute(DefaultRequestDirector.java:486)

at org.apache.http.impl.client.AbstractHttpClient.doExecute(AbstractHttpClient.java:863)

at org.apache.http.impl.client.CloseableHttpClient.execute(CloseableHttpClient.java:82)

at org.apache.tomcat.maven.common.deployer.TomcatManager.invoke(TomcatManager.java:742)

at org.apache.tomcat.maven.common.deployer.TomcatManager.deployImpl(TomcatManager.java:705)

at org.apache.tomcat.maven.common.deployer.TomcatManager.deploy(TomcatManager.java:388)

at org.apache.tomcat.maven.plugin.tomcat7.deploy.AbstractDeployWarMojo.deployWar(AbstractDeployWarMojo.java:85)

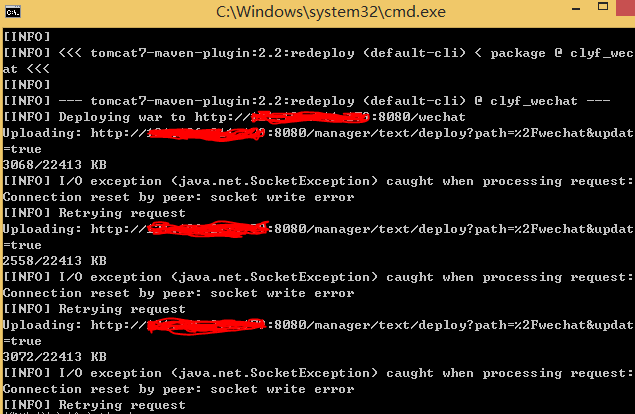
at org.apache.tomcat.maven.plugin.tomcat7.deploy.AbstractDeployMojo.invokeManager(AbstractDeployMojo.java:82)

at org.apache.tomcat.maven.plugin.tomcat7.AbstractCatalinaMojo.execute(AbstractCatalinaMojo.java:132)

... 23 more

[复制代码](javascript:void(0);)

使用mvn tomcat7:redeploy时出现如下情况



经过查询Tomcat文档后发现，这是由于Tomcat的远程地址拦截器造成的结果，默认情况下，Tomcat的Manager和Host-Manager只接受本机的请求，而要让它接受远程的请求，需要添加上面提到的manager.xml的配置（第一步配置过了就不要加了），也就是：

<?xml version="1.0" encoding="UTF-8"?>

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

docBase="${catalina.home}/webapps/manager">

<Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="^.\*$" />

</Context>

由于Tomcat的Manager可以执行项目的部署、卸载等敏感操作，如果你只想允许特定的IP地址访问Manager，可在上面的allow属性中设置规则。具体规则设置见下面的链接：

http://tomcat.apache.org/tomcat-7.0-doc/config/valve.html#Remote\_Address\_Filter

问题说明：http://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html#Configuring\_Manager\_Application\_Access

(3)自动部署显示成功，war包也上传成功，但是war不自动解压自动部署。

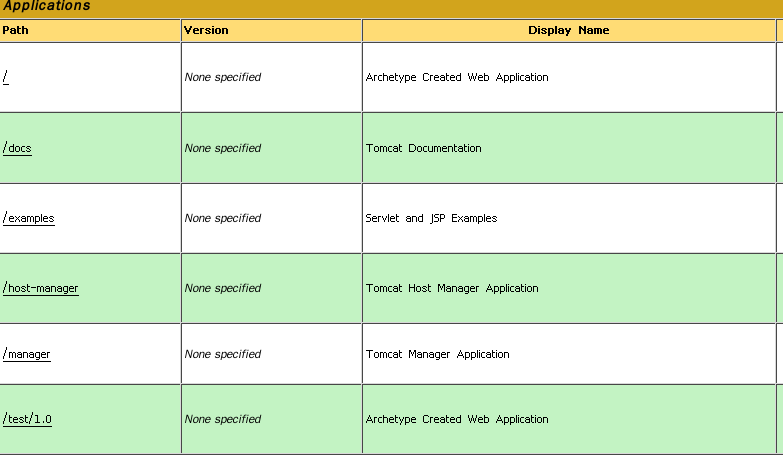
如果你在tomcat的server.xml中通过设置<Context>标签来部署相同名称的项目的话，maven发布到该服务器的war不会被自动解压，部署，更新，需要去掉server.xml中该项目的<Context>标签。

**5. 其他**

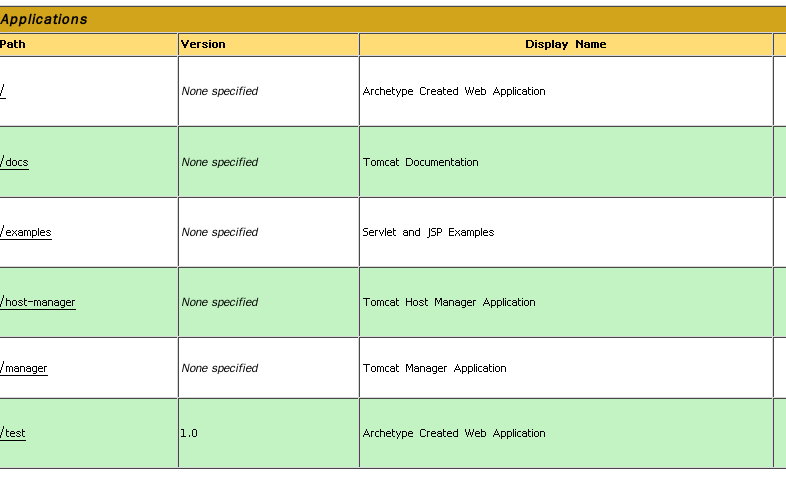
如果想要实现对部署路径加版本，可将上面tomcat7-maven-pluginconfiguration的path设置为

<path>/webapp#version</path>

的形式，部署后，当前项目在服务器端的路径就是/webapp/version。举个例子，如果path设置为 test#1.0，那么服务端项目实际的路径就是/test/1.0。如下：



如果名字和版本号之间是两个#，效果就是制定当前项目在manager网页中显示的版本，路径不变，举个例子，path设置为test##1.0，实际部署路径为/test，但是在manager网页中，显示如下，注意Version一栏的值：



参考：

http://tomcat.apache.org/tomcat-7.0-doc/config/context.html#Naming

# tomcat进行redeploy时出错：could not be completely deleted

2015年06月13日 00:20:24

阅读数：1465

在windows下使用Jenkins进行tomcat的自动部署时发现一个问题，在配置好了tomcat后每次进行自动部署时tomcat都会报一个错误：could not be completely deleted。原因是因为jenkins进行自动部署时流程如下：

**[plain]** [view plain](https://blog.csdn.net/feierky/article/details/46477039) [copy](https://blog.csdn.net/feierky/article/details/46477039)

1. Redeploying [C:\Program Files (x86)\jenkins\jobs\test one\workspace\jenkins.war]
2. Undeploying [C:\Program Files (x86)\jenkins\jobs\test one\workspace\jenkins.war]
3. Deploying [C:\Program Files (x86)\jenkins\jobs\test one\workspace\jenkins.war]

在tomcat进行undeploying时会先删除已经部署的目录再重新部署，不知原因在删除旧的目录时没办理彻底删除掉，所以才会出现上面的提示：could not be completely deleted。

在网上找了方法，说是需要修改$TOMCAT\_HOME/conf/context.xml：

<Context antiJARLocking="true" antiResourceLocking="true">

按这个方法修改了，不过在部署进还是会出现上面的错误提示，不过项目是可以继续deploying了，而且可以正常访问，具体怎么才能不出现错误提示还没有找到方法。