[docker安装tomcat&部署javaweb程序](https://www.cnblogs.com/kaixinyufeng/p/9689982.html)

**一。docker定制简单的java-web应用镜像**

网址：

1。jdk下载网址：[https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html](https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html" \t "_blank)

docker安装tomcat8

[基于docker安装tomcat](https://blog.csdn.net/mynameissls/article/details/80825661)

[docker操作镜像](https://blog.csdn.net/winy_lm/article/details/77980529)

**Eg1:以本地java打war包为例，构建一个镜像**

[复制代码](javascript:void(0);)

Step1:以tomcat镜像为基础镜像

docker pull tomcat //拉取tomcat镜像

Step2:定制镜像

touch Dockerfile #新建Dockerfile文件

vi Dockerfile #编辑Dockerfile

##add content start

FROM tomcat:latest

MAINTAINER shiyufeng kaixinyufeng@sina.com

COPY web-ssm-single.war /usr/local/tomcat/webapps

##add content end

Step3:构建镜像

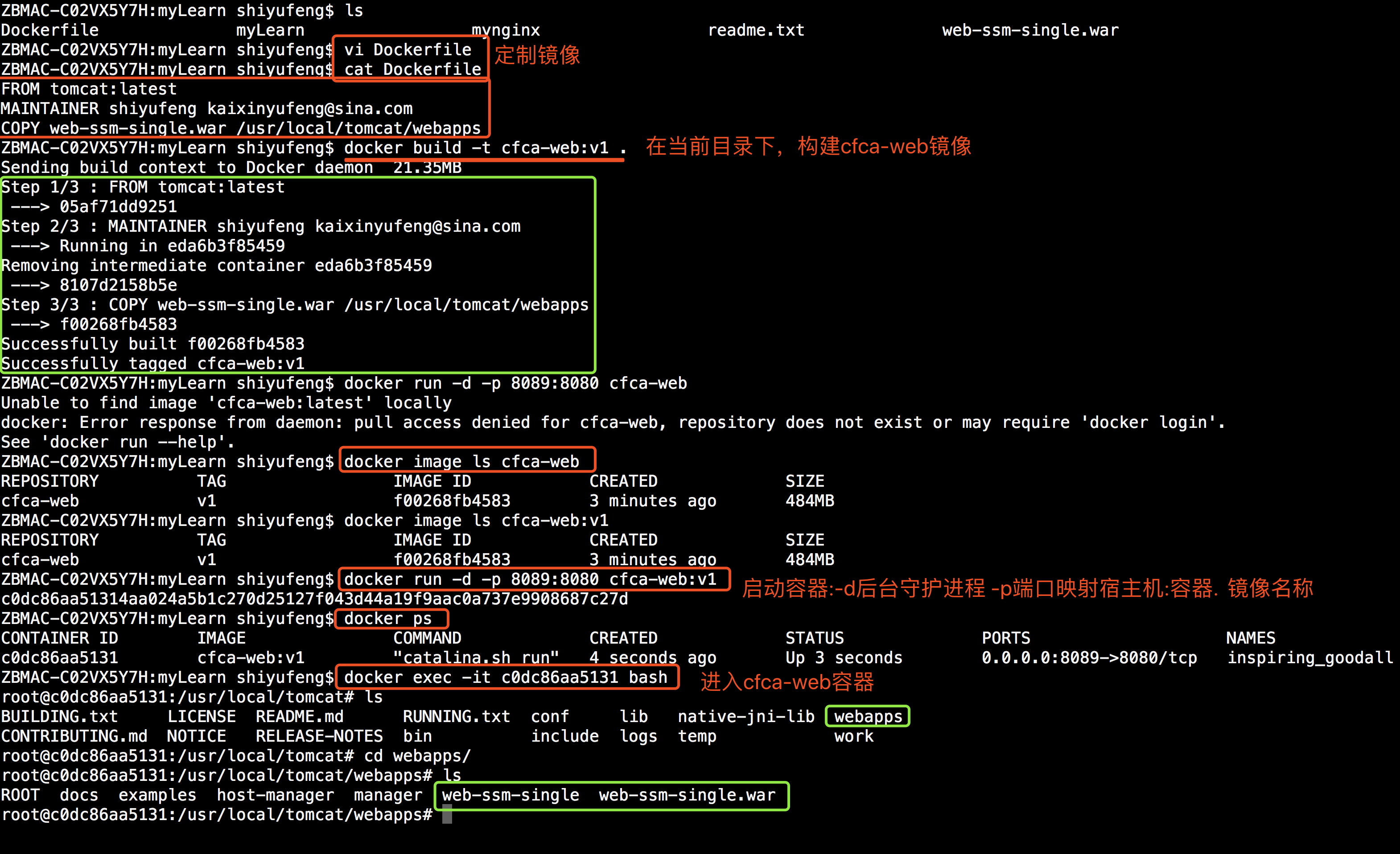
docker build -t cfca-web:v1 . //在当前目录，构建一个cfca-web的镜像，tag为v1的镜像

Step4:run镜像为容器

docker run -d -p 8089:8080 cfca-web:v1 //-d后台守护进程 -p映射端口宿主机:容器 镜像名

地址栏访问：http://localhost:8089/web-ssm-single/即可。

[复制代码](javascript:void(0);)



命令行访问：

**ZBMAC-C02VX5Y7H:myLearn shiyufeng$ curl http://localhost:8089/web-ssm-single**

**ZBMAC-C02VX5Y7H:myLearn shiyufeng$ curl http://localhost:8089/web-ssm-single/**

**<html>**

**<body>**

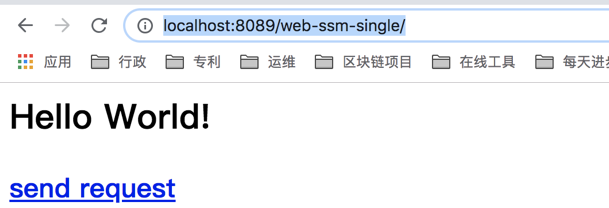
**<h2>Hello World!</h2>**

**<h3><a href="/user/query">send request</a></h3>**

**</body>**

**</html>**

地址栏访问：http://localhost:8089/web-ssm-single/

成功！！！

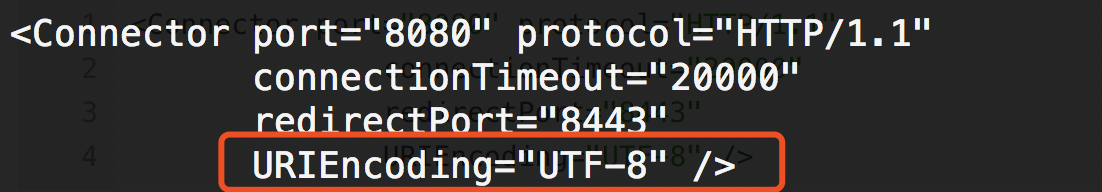
[**二。实战docker，编写Dockerfile定制tomcat镜像，实现web应用在线部署**](https://blog.csdn.net/boling_cavalry/article/details/70184605)

参考博客源码地址：[https://github.com/zq2599/blog\_demos.git](https://github.com/zq2599/blog_demos.git" \t "_blank) 项目名：loadbalancedemo

Step1：创建空文件夹：mkdir image\_tomcat

Step2：拷备apache-tomcat-7.0.86/conf目录下的文件server.xml和tomcat-users.xml到image\_tomcat目录下面

Step3：修改server.xml



<Connector executor="tomcatThreadPool"

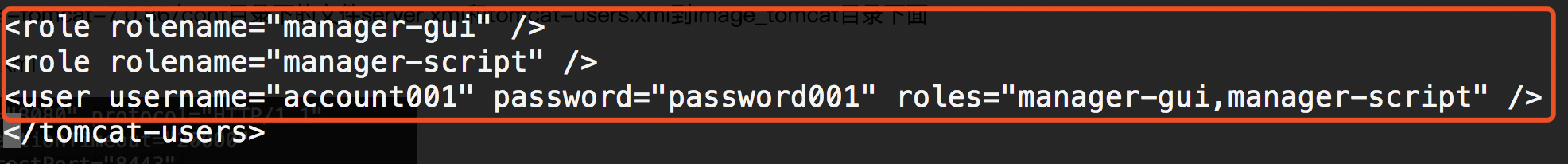
port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

增加：URIEncoding="UTF-8" 目的：处理中文问题

Step4：修改tomcat-users.xml,增加如下内容：



<role rolename="manager-gui" />

<role rolename="manager-script" />

<user username="account001" password="password001" roles="manager-gui,manager-script" />

以上配置完成后，即可用account001账号在对应的tomcat上进行在线部署操作了.

Step5：定制Docker镜像

[复制代码](javascript:void(0);)

cd image\_tomcat

touch Dockerfile //创建Dockerfile文件

vim Dockerfile //编辑Dockerfile文件

//###start 添加如下内容

#First docker file from bolingcavalry  
#version 0.0.1  
#Author:shiyufeng  
#基础镜像  
FROM tomcat:7.0.77-jre8

#作者  
MAINTAINER shiyufeng

#定义工作目录  
ENV WORK\_PATH /usr/local/tomcat/conf

#定义要替换的文件名  
ENV USER\_CONF\_FILE\_NAME tomcat-users.xml

#定义要替换的server.xml文件名  
ENV SERVER\_CONF\_FILE\_NAME server.xml

#删除原文件tomcat-users.xml  
RUN rm $WORK\_PATH/$USER\_CONF\_FILE\_NAME

#复制文件tomcat-users.xml  
COPY ./$USER\_CONF\_FILE\_NAME $WORK\_PATH/

#删除原文件server.xml  
RUN rm $WORK\_PATH/$SERVER\_CONF\_FILE\_NAME

#复制文件server.xml  
COPY ./$SERVER\_CONF\_FILE\_NAME $WORK\_PATH/

//###end

[复制代码](javascript:void(0);)

以上：image\_tomcat目录结构：

ZBMAC-C02VX5Y7H:image\_tomcat shiyufeng$ tree

.

├── Dockerfile

├── server.xml

└── tomcat-users.xml

Step6:构建docker镜像

//在当前目录构建镜像。注意后面的点为当前目录

docker build -t tomcat001:0.0.1 .

//查看镜像

docker image ls tomcat001

Step7:镜像构建完毕，启动tomcat

[复制代码](javascript:void(0);)

//启动tomcat容器

docker run --name=tomcat001 -p 8080:8080 -e TOMCAT\_SERVER\_ID=tomcat\_server\_001 -idt tomcat001:0.0.1

/\*

--name：为容器起的别名

-p:映射端口，宿主机与容器

\*/

[复制代码](javascript:void(0);)

Step8:启动浏览器，访问localhost:8080,显示tomat图标即为启动成功.

Step9:修改mavenl配置文件setting.xml，增加如下配置

<server>

<id>tomcat7</id>

<username>account001</username>

<password>password001</password>

</server>

以上是tomcat的tomcat-users.xml文件中配置的用户和密码，这样执行maven插件的时候就能从此处取得对应的用户名和密码去tomcat上做操作了。

Step10：进入web工程，pom.xml目录执行如下命令

注：刚定制的docker镜像必须启动状态，才可以成功构建，否则会报“构建失败”

mvn clean package -U -Dmaven.test.skip=true tomcat7:redeploy

执行结果如下所示，即代表tomcat7-maven-plugin插件已经帮我们把war部署到docker容器上去了

[复制代码](javascript:void(0);)

[INFO] tomcatManager status code:200, ReasonPhrase:OK

[INFO] OK - Deployed application at context path /loadbalancedemo

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 6.561 s

[INFO] Finished at: 2018-12-15T17:08:27+08:00

[INFO] Final Memory: 21M/216M

[INFO] ------------------------------------------------------------------------

[复制代码](javascript:void(0);)

Step11：浏览器输入http://localhost:8080/loadbalancedemo/hello?name=开心玉凤

显示结果：

您好开心玉凤，这是来自tomcat\_server\_001的问候！2018-12-16 01:51:44

效果如下所示：



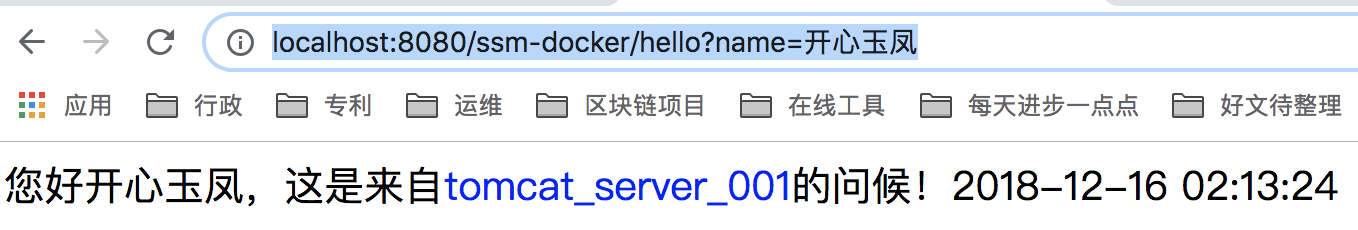
以上：项目名称修改为ssm-docker，修改点：

（1）右击项目名-》Refactor-》重新命名

（2）pom.xml中修改<finalname>

（3）project structure-》相应修改项目名称即可。

地址栏输入：http://localhost:8080/ssm-docker/hello?name=开心玉凤



**项目应用：java-web项目（[spring+springmvc+mybatis+maven架构](https://www.cnblogs.com/kaixinyufeng/p/9143283.html" \t "_blank)）**

项目需求：定制docker镜像，方便jenkins部署。

第一版本：应用docker定制简单的镜像示例，拷备方式。

[复制代码](javascript:void(0);)

touch Dockerfile

vim Dockerfile

##add content

FROM tomcat:latest

COPY ./target/cfca-web.war /usr/local/tomcat/webapps

[复制代码](javascript:void(0);)

第二版本：参考[实战docker,编写Dockerfile定制tomcat镜像,实现web应用在线部署](https://blog.csdn.net/boling_cavalry/article/details/70184605" \t "_blank)

场景：搭建一台虚拟机，定制tomcat镜像。

本机java-web项目只需要修改pom.xml文件配置

[复制代码](javascript:void(0);)

//修改url即可

<plugin>

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

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

<version>2.2</version>

<configuration>

<!--<url>http://localhost:8080/manager/text</url>-->

<url>http://192.168.99.100:8090/manager/text</url>

<server>tomcat7</server>

<path>/${project.artifactId}</path>

<update>true</update>

</configuration>

</plugin>

[复制代码](javascript:void(0);)

maven打包部署：mvn clean package -U -Dmaven.test.skip=true tomcat7:redeploy

浏览器访问：http://192.168.99.100:8090/cfca/ 即可显示index.jsp内容。(任意修改即可成功部署)

**注：**由于jekins部署git地址与指定的url地址不通，会出现问题：不能向指定的url中的tomcat中部署war包。

解决方案：[该url通过maven命令动态传递](https://killeraction.iteye.com/blog/1936019" \t "_blank)

[复制代码](javascript:void(0);)

//step1：属性配置中增加参数<maven.url>或其他如connectionUrl

<properties>

<maven.url></maven.url>

</properties>

//Step2:修改url配置 如${connectionUrl}

<url>${maven.url}</url>

//Step3：maven部署打包命令中增加参数 或DconnectionUrl=xxx

mvn clean package -U -Dmaven.test.skip=true -Dmaven.url=http://192.168.99.100:8090/manager/text tomcat7:redeploy

[复制代码](javascript:void(0);)

**项目过程遇到问题点：**

执行mvn clean package -U -Dmaven.test.skip=true tomcat7:redeploy命令一直报“maven错误，找不到符号”

解决方案：1。[Maven错误找不到符号](https://blog.csdn.net/u012062455/article/details/52347099" \t "_blank)   2。[既使用maven编译，又使用lib下的Jar包](https://blog.csdn.net/catoop/article/details/48489365" \t "_blank)（本项目问题原因）

[方案一：](https://blog.csdn.net/tterminator/article/details/78198801)  
在项目中创建一个目录lib,然后将jar复制到该文件夹下，最后在maven编译插件中配置如下：

[复制代码](javascript:void(0);)

<plugin>

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

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.2</version>

<configuration>

<source>${jdk.version}</source>

<target>${jdk.version}</target>

<encoding>${project.build.sourceEncoding}</encoding>

<!-- 新增的编译配置参数 -->

<compilerArguments>

<!-- 对应jar放置在项目中的位置 -->

<extdirs>${project.basedir}/src/main/webapp/WEB-INF/lib</extdirs>

</compilerArguments>

</configuration>  
</plugin>

[复制代码](javascript:void(0);)

[方案二：](https://blog.csdn.net/tterminator/article/details/78198801)  
  步骤一：模仿maven的本地仓库目录结构，创建类似的目录，把二方包jar包放到创建好的目录中；   
  步骤二：在pom文件中添加本地仓库中的二方包jar包坐标。