搭建HTTPS网站

# 简单介绍

SSL证书申请可以找国内一些网站

腾讯云免费GeoTrust DV SSL证书: <https://console.qcloud.com/ssl>

阿里云证书

七牛云证书

又拍云证书： <https://console.upyun.com/toolbox/createCertificate/> （我用的就是这个）

登录账号：colorfulfrog/lovemyself201314



如图，SSL 证书按大类一般可分为 DV SSL 、OV SSL 、EV SSL 证书。有的也会叫做域名型、企业型、增强型证书，不同的厂商叫法可能有所不同，但差别不大。

**1）**[**域名型SSL证书（DV SSL）**](https://link.zhihu.com/?target=https%3A//www.upyun.com/products/ssl)

即证书颁布机构只对域名的所有者进行在线检查，通常是验证域名下某个指定文件的内容，或者验证与域名相关的某条 TXT 记录；

比如访问 [http|https]://[http://www.domain.com/](https://link.zhihu.com/?target=http%3A//www.domain.com/)…/test.txt，文件内容： 2016082xxxxx39w7b20nelfa；

或在与域名相关的DNS服务器上添加一条 TXT 记录：[http://www.domain.com](https://link.zhihu.com/?target=http%3A//www.domain.com) –> TXT –> 20170xxxxxqmkiby43hpvy8

**2）**[**企业型SSL证书（OV SSL）**](https://link.zhihu.com/?target=https%3A//www.upyun.com/products/ssl)

是要购买者提交组织机构资料和单位授权信等在官方注册的凭证，证书颁发机构在签发 SSL 证书前不仅仅要检验域名所有权，还必须对这些资料的真实合法性进行多方查验，只有通过验证的才能颁发 SSL 证书。

**3）**[**增强型SSL证书（EV SSL）**](https://link.zhihu.com/?target=https%3A//www.upyun.com/products/ssl)

与其他 SSL 证书一样，都是基于 SSL/TLS 安全协议，但是验证流程更加具体详细，验证步骤更多，这样一来证书所绑定的网站就更加的可靠、可信。它跟普通 SSL 证书的区别也是明显的，安全浏览器的地址栏变绿，如果是不受信的 SSL 证书则拒绝显示，如果是钓鱼网站，地址栏则会变成红色，以警示用户。

**常见 CA 厂商对比：**



**国内提供SSL证书服务的云厂商列表：**



[又拍云](https://link.zhihu.com/?target=https%3A//www.upyun.com/products/ssl)免费提供 Let’s Encrypt 和 Symantec 签发的两款 DV SSL 证书，也是业内唯一一家提供两种免费证书的服务商。

# 申请证书

登录<https://console.upyun.com/toolbox/createCertificate/>

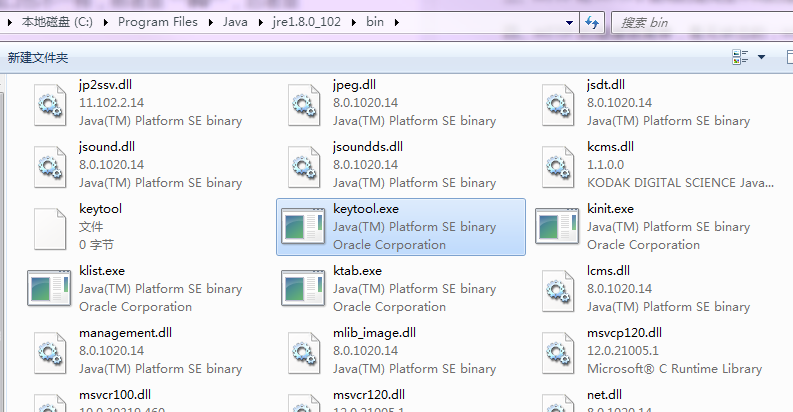
用户/密码：colorfulfrog/lovemyself201314

# 本地模拟

HTTPS 如果生产环境应用在域名上是需要直接或间接的从 CA 申请证书，来取得浏览器的信任的。我们先在本地模拟测试一下这个过程，自己生成证书，后面介绍域名启用 HTTPS。

## ****① keytool工具生成证书****

打开 JDK 自带的 keytool 目录。

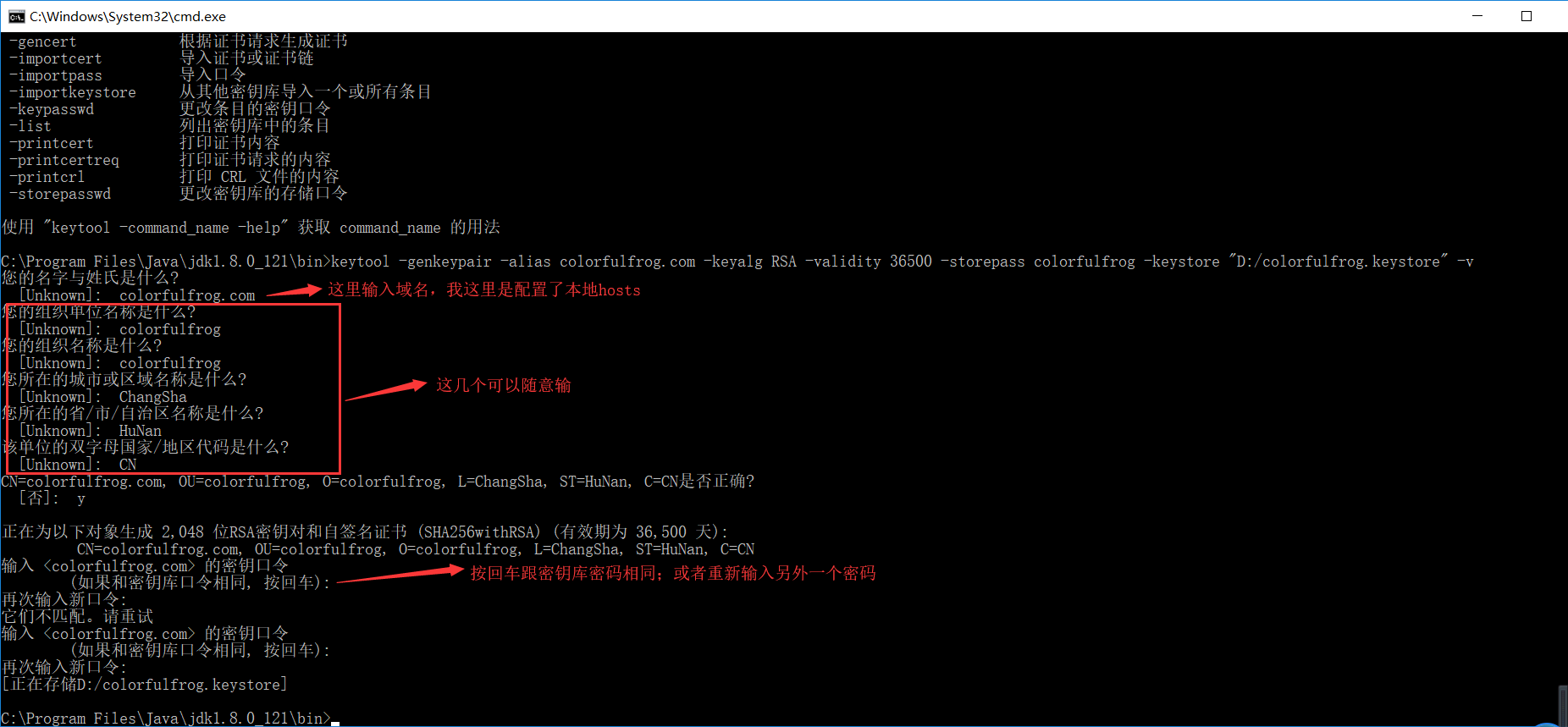


打开cmd控制台

## **1、新生成一个密钥库**

keytool -genkeypair -alias colorfulfrog.com -keyalg RSA -validity 36500 -storepass colorfulfrog -keystore "D:/colorfulfrog.keystore" –v

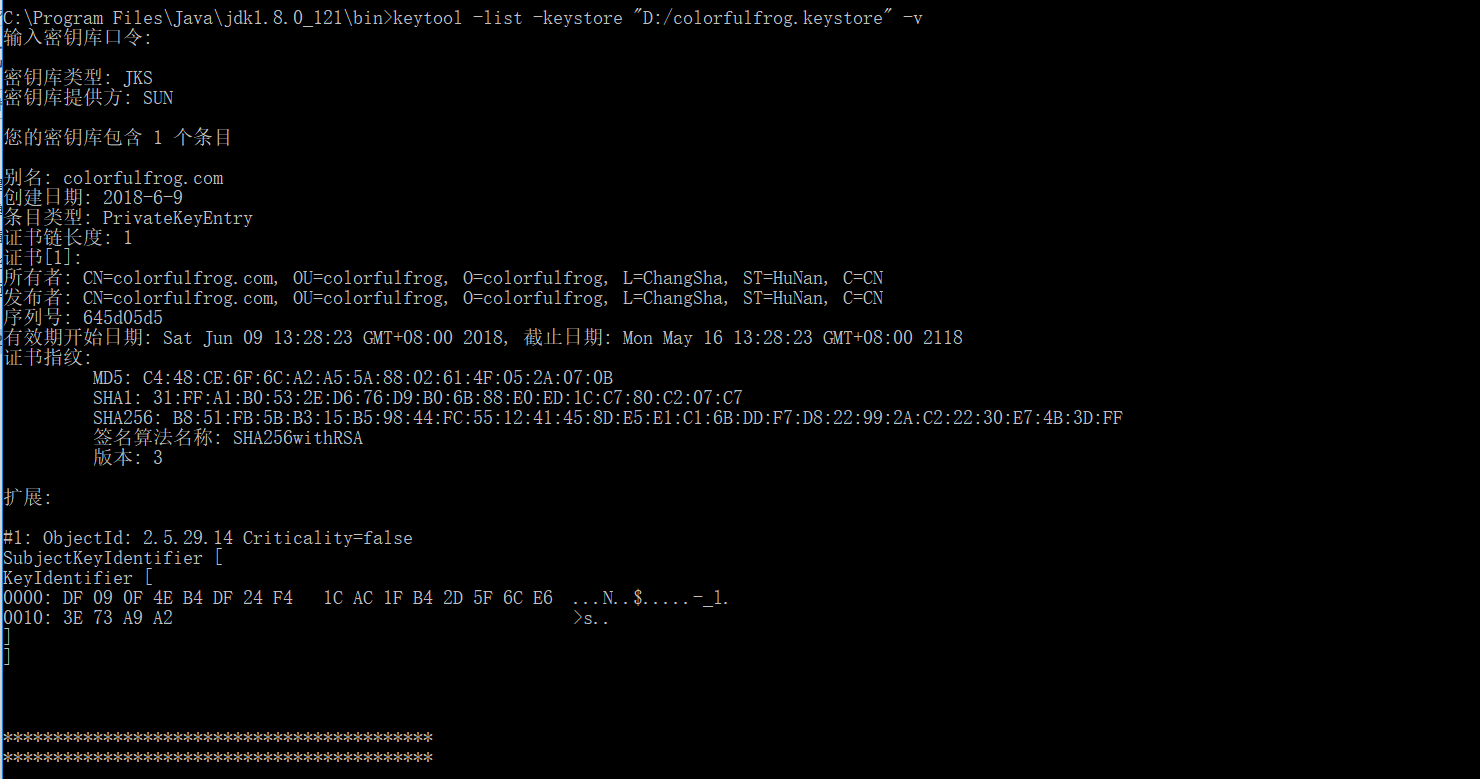
* -alias colorfulfrog.com：证书项的名字，必填项
* -keyalg RSA：证书签名算法，tomcat建议RSA
* -validity 36500：证书有效期，36500天，即100年
* -storepass colorfulfrog：密钥库密码，也就是等下要生成的colorfulfrog.keystore的访问密码，妥善保管
* -keystore "D:/colorfulfrog.keystore"：要生成的文件的位置
* -v：该选项会显示比较详细的信息



如上图所示第一项必须写你测试用的网址，这里我写的是colorfulfrog.com。至于图中提示的密钥口令，没什么用，就是将来访问那条密钥时要输入的口令，直接回车就行。

## **2、查看密钥库中的项**

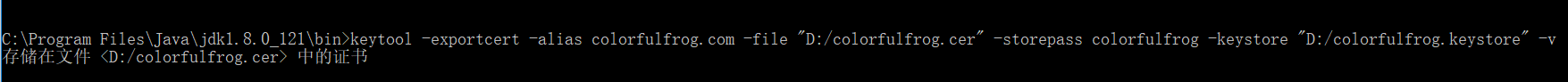
keytool -list -keystore "D:/colorfulfrog.keystore" –v



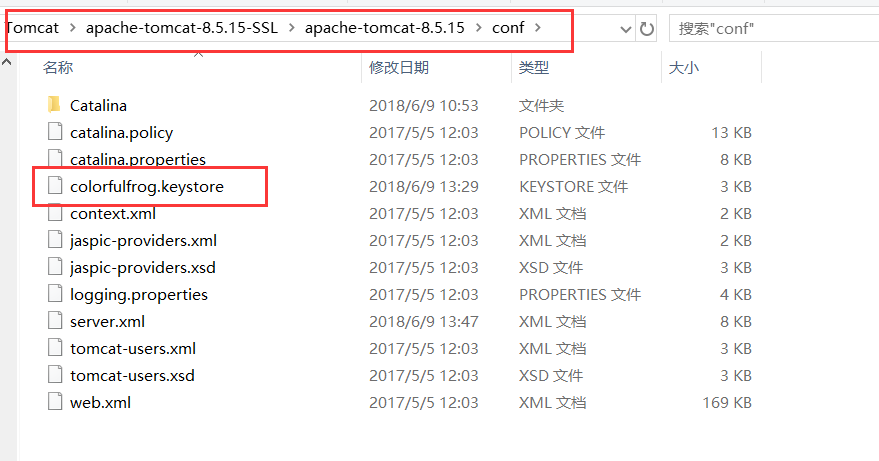
## 3、导出证书（供浏览器导入）

keytool -exportcert -alias colorfulfrog.com -file "D:/colorfulfrog.cer" -storepass colorfulfrog -keystore "D:/colorfulfrog.keystore" –v

* -alias colorfulfrog.com：要导出的证书的名字，即刚才创建的密钥的名字，即keytool -list时显示的名字
* -file D:/colorfulfrog.cer：要导出的证书的存储位置，这里我放在D盘下
* -keystore D:/colorfulfrog.keystore：刚才创建的密钥库的位置



## 4、将密钥库拷贝至tomcat目录下，我这里放在/conf目录下



## 5、配置SSL访问

<Connector connectionTimeout="20000" port="80" protocol="HTTP/1.1" redirectPort="443"/>

<Connector port="443" protocol="org.apache.coyote.http11.Http11NioProtocol"

maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

clientAuth="false" sslProtocol="TLS" keystoreFile="/conf/ colorfulfrog.keystore"

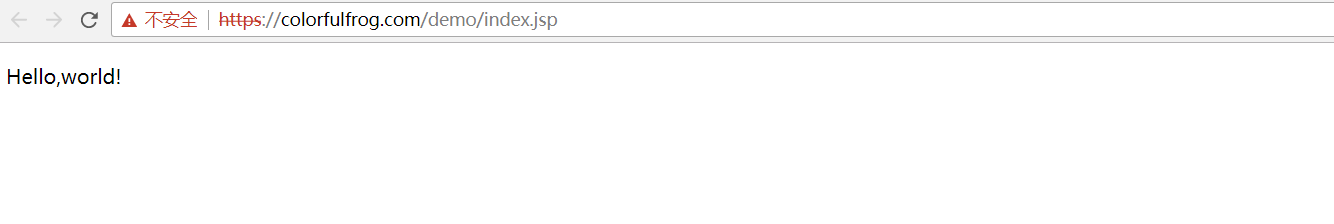
keystorePass="colorfulfrog"/>

<Connector port="8009" protocol="AJP/1.3" redirectPort="443" />

要修改的配置文件是tomcat的server.xml文件。仔细看上面的配置，第一个Connector是默认就有的，这里只是把8080端口改成了公认的80端口，把8443端口改成了公认的443端口。第二个Connector默认是注释掉的，搜索8443就能找到，直接把上述第二个Connector粘贴到server.xml中第一个Connector的下面，方便管理。keystoreFile=”/conf/ colorfulfrog.keystore”就是刚才copy的文件的位置，可以自己改到其他位置。keystorePass=” colorfulfrog”就是刚才创建密钥库时使用的口令。

## 6、启动tomcat，

访问<https://colorfulfrog.com/demo/index.jsp>



这时浏览器上显示不安全，可以将colorfulfrog.cer证书导入浏览器，设置为可信任机构，就不会报不安全啦。

## 7、强制http转https

在tomcat\conf\web.xml中的</welcome-file-list>后面加上这样一段：

<!-- 配置强制使用https协议 -->

<login-config>

<!-- Authorization setting for SSL -->

<auth-method>CLIENT-CERT</auth-method>

<realm-name>Client Cert Users-only Area</realm-name>

</login-config>

<security-constraint>

<!-- Authorization setting for SSL -->

<web-resource-collection >

<web-resource-name >SSL</web-resource-name>

<url-pattern>/\*</url-pattern>

</web-resource-collection>

<user-data-constraint>

<transport-guarantee>CONFIDENTIAL</transport-guarantee>

</user-data-constraint>

</security-constraint>