# 介绍

## 什么是HTTPS

HTTPS其实是有两部分组成：HTTP + SSL / TLS，

也就是在HTTP上又加了一层处理加密信息的模块，并且会进行身份的验证。

问题：

Firebug和postman之类的浏览器调试工具，为什么获取到的是明文？

解答：

SSL是对传输的数据进行加密，针对的是传输过程的安全。

firebug之类的浏览器调试工具，

因为他们得到的是客户端加密之前/解密之后的数据，因此是明文的。

## 什么是自签名证书

就是自己生成的证书，并不是官方生成的证书。

除非是很正式的项目，否则使用自己签发的证书即可，因为官方生成证书是要花钱滴。

## 怎么制作自签名证书

1. 官方生成证书（直接花钱买呗）

2. 使用openssl工具生成

3. 使用jdk自带工具生成

# openssl工具生成

参考：http://www.cnblogs.com/notDog/p/5264666.html

# jdk生成

一般网络上生成自用的不受浏览器信任的证书都是直接通过JDK自带的应用keytool来制作的，而Ngnix不兼容keytool生成的证书来部署HTTPS网站，所以需要使用OpenSSL来生成证书。

## 为服务器生成证书

打开cmd窗口，切换至JDK的bin目录下。

使用keytool命令生成证书：

keytool

-genkey

-alias tomcat(别名)

-keypass 123456(别名密码)

-keyalg RSA(算法)

-keysize 1024(密钥长度)

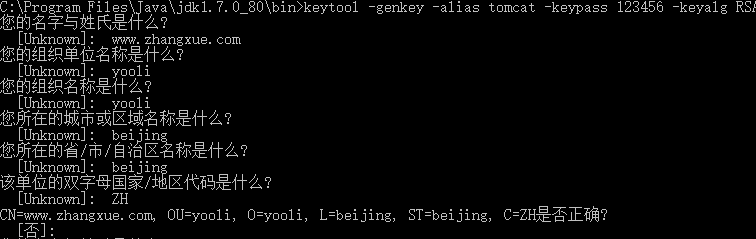
-validity 365(有效期，天单位)

-keystore D:/keys/tomcat.keystore(指定生成证书的位置和证书名称)

-storepass 123456(获取keystore信息的密码)

方便复制版：

keytool -genkey -alias tomcat -keypass 123456 -keyalg RSA -keysize 1024 -validity 365 -keystore D:/tomcat.keystore -storepass 123456



## 为客户端生成证书

为浏览器生成证书，以便让服务器来验证它。

为了能将证书顺利导入至IE和Firefox，证书格式应该是PKCS12，

因此，使用如下命令生成：

keytool

-genkey

-alias client

-keypass 123456

-keyalg RSA

-storetype PKCS12

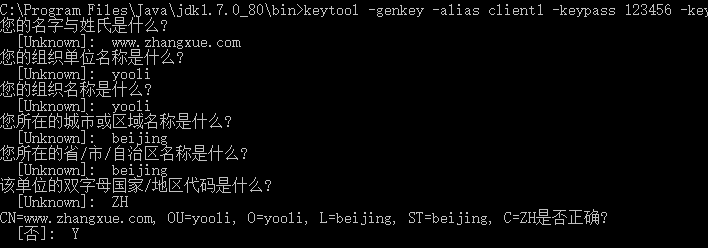
-keypass 123456

-storepass 123456

-keystore D:/client.p12

方便复制版：

keytool -genkey -alias client -keypass 123456 -keyalg RSA -keysize 1024 -validity 365 -storetype PKCS12 -keystore D:/client1.p12 -storepass 123456



## 让服务器信任客户端证书

第一步

由于不能直接将PKCS12格式的证书库导入，必须先把客户端证书导出为一个单独的CER文件，使用如下命令：

keytool -export -alias client -keystore D:/client.p12 -storetype PKCS12 -keypass 123456 -file D:/client.cer



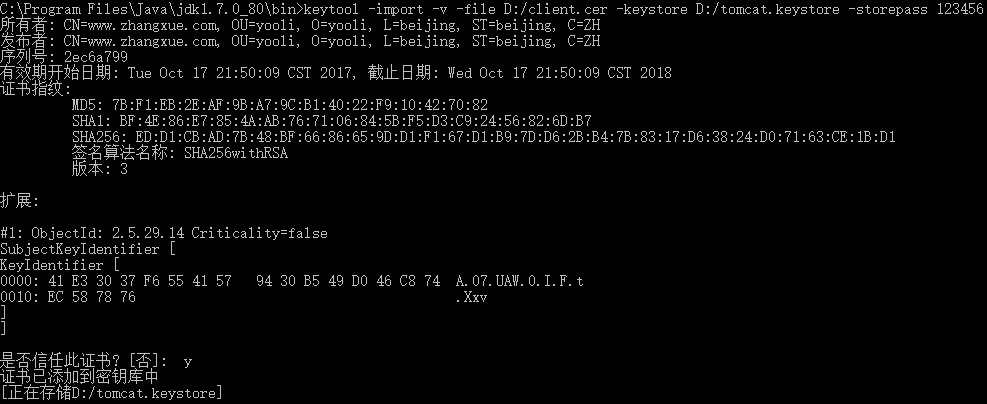
注意：

Keypass：指定CER文件的密码，但会被忽略，而要求重新输入，这里需要指定密钥库的口令，暂时先写为123456

第二步

将该文件导入到服务器的证书库，添加为一个信任证书：

keytool -import -v -file D:/client.cer -keystore D:/tomcat.keystore -storepass 123456



第三步

完成之后通过list命令查看服务器的证书库，可以看到两个证书，一个是服务器证书，一个是受信任的客户端证书（输入密钥库口令）：

keytool -list -v -keystore D:/tomcat.keystore



## 让客户端信任服务器证书

第一步：

由于是双向SSL认证，客户端也要验证服务器证书，因此，必须把服务器证书添加到浏览器的“受信任的根证书颁发机构”。

由于不能直接将keystore格式的证书库导入，必须先把服务器证书导出为一个单独的CER文件，使用如下命令：

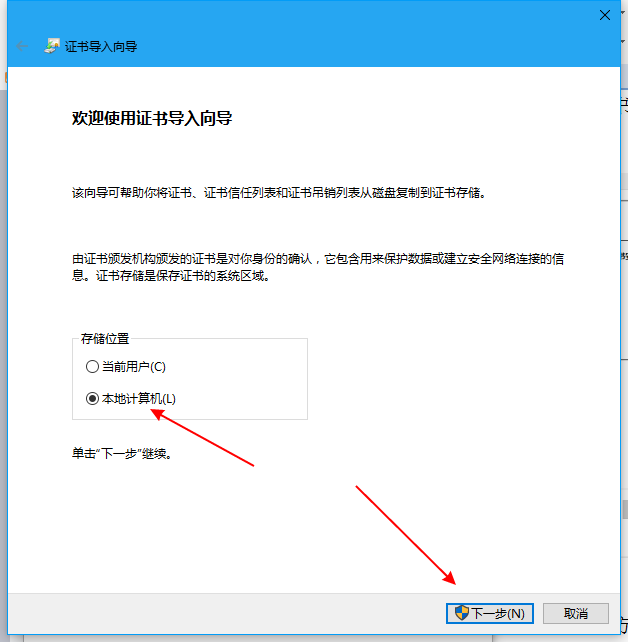
keytool -keystore D:/tomcat.keystore -export -alias tomcat -file D:/server.cer



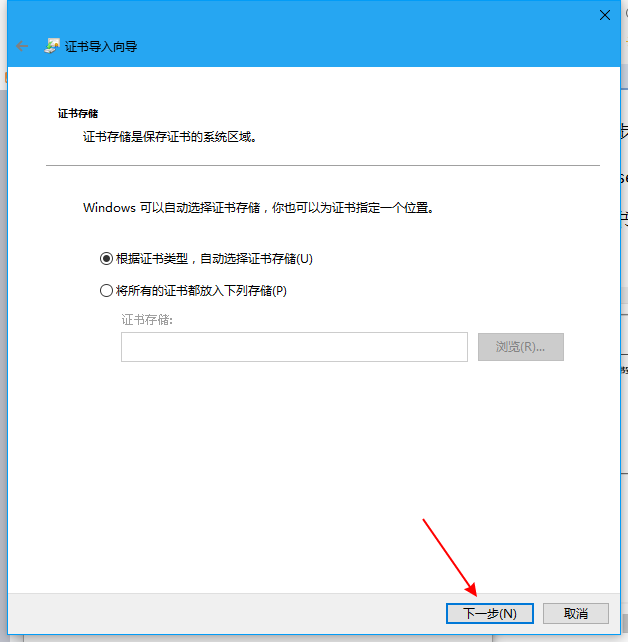
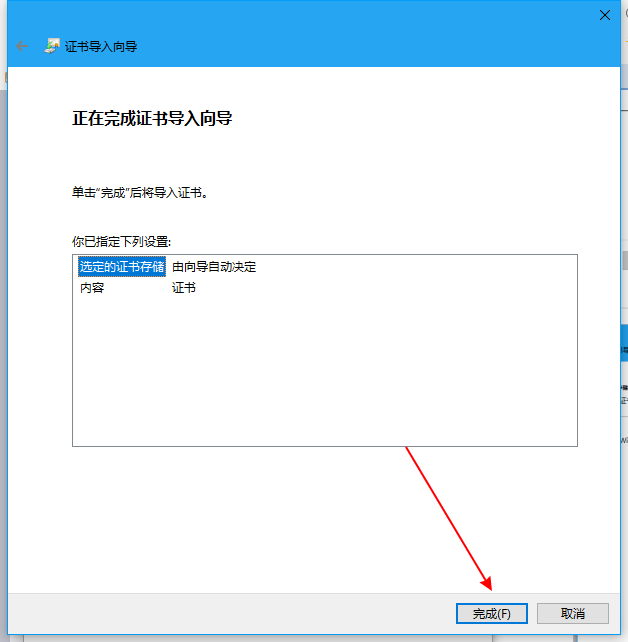
第二步：

双击server.cer文件，按照提示安装证书，

将证书填入到“受信任的根证书颁发机构”。



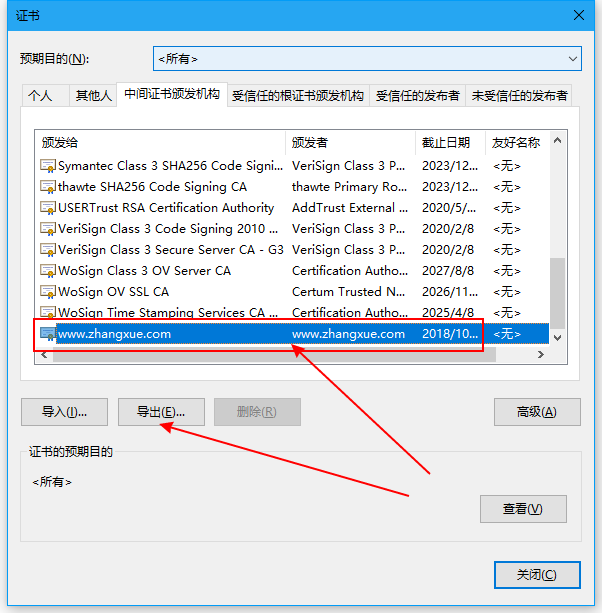
填入方法：



第三步：

打开IE浏览器-->工具-->internet选项-->内容-->证书-->中间证书颁发机构里

找到www.zhangxue.com(该名称即时你前面生成证书时填写的名字与姓氏)证书导出来-再把导出来的证书导入”受信任的根颁发机构” 就OK了。



## 配置Tomcat服务器

修改server.xml

将默认的8080端口的配置修改为下面的配置

<Connector

port="443"

protocol="org.apache.coyote.http11.Http11NioProtocol"

SSLEnabled="true"

maxThreads="150"

scheme="https"

secure="true"

clientAuth="false"

sslProtocol="TLS"

keystoreFile="D:/java/apache-tomcat-8.0.26/tomcat.keystore"

keystorePass="123456"

truststoreFile="D:/java/apache-tomcat-8.0.26/tomcat.keystore"

truststorePass="123456" />

属性说明：

clientAuth:

true强制双向验证，false服务器端单向验证，want客户端有证书就验证，没有就不验证。

keystoreFile:服务器证书文件路径

keystorePass:服务器证书密码

truststoreFile:用来验证客户端证书的根证书，此例中就是服务器证书

truststorePass:根证书密码

注意：

① 设置clientAuth属性为true时，需要手动导入客户端证书才能访问。（小公司就false）

② 要访问https请求默认访问443端口，访问http请求默认则访问80认端口。

## 强制 https 访问（附件）

在 tomcat /conf/web.xml 中的 </welcome- file-list> 后面加上这

<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>

完成以上步骤后，在浏览器中输入http的访问地址也会自动转换为https了。

# Spring boot配置ssl

<https://gitee.com/zx19890628/spring-boot-example/blob/master/lab_000_base_web/src/main/java/com/zx/web/ssl/%E9%85%8D%E7%BD%AESSL%E5%AE%9E%E7%8E%B0https%E8%AF%B7%E6%B1%82.md>