流程

1. **生成CA证书 – 这里是先自己充当一下CA**
2. 创建私钥
   1. openssl genrsa -out root/root-key.pem 1024
3. 创建证书请求，相关参数
   1. openssl req -new -out root/root-req.csr -key root/root-key.pem
   2. Country Name (2 letter code) [AU]:CN
   3. State or Province Name (full name) [Some-State]:Beijing
   4. Locality Name (eg, city) []:Beijing
   5. Organization Name (eg, company) [Internet Widgits Pty Ltd]:ccb
   6. Organizational Unit Name (eg, section) []:it
   7. Common Name (e.g. server FQDN or YOUR name) []:luca
   8. Email Address []:.
   9. Please enter the following 'extra' attributes
   10. to be sent with your certificate request
   11. A challenge password []:testpassword
   12. An optional company name []:.

（”.” 为blank的意思）

1. 自签署证书（无上级CA，自己签发）
   1. openssl x509 -req -in root/root-req.csr -out root/root-cert.pem -signkey root/root-key.pem -days 3650

输出：Signature ok

1. subject=/C=CN/ST=Beijing/L=Beijing/O=ccb/OU=it/CN=luca
2. 将证书导出为浏览器支持的.p12格式
   1. openssl pkcs12 -export -clcerts -in root/root-cert.pem -inkey root/root-key.pem -out root/root.p12
   2. 导出密码testpassword （和上面的A challenge password虽然一样，但是用处不一样，A challenge password用来作为申请证书时候的密码，这个是导出证书用的密码）
3. **生成服务器证书**
4. 创建私钥
   1. openssl genrsa -out server/server-key.pem 1024
5. 创建证书请求
   1. openssl req -new -out server/server-req.csr -key server/server-key.pem
   2. Country Name (2 letter code) [AU]:CN
   3. State or Province Name (full name) [Some-State]:Beijing
   4. Locality Name (eg, city) []:Beijing
   5. Organization Name (eg, company) [Internet Widgits Pty Ltd]:ccb-subserver
   6. Organizational Unit Name (eg, section) []:it-subserver
   7. Common Name (e.g. server FQDN or YOUR name) []:sihongxing
   8. Email Address []:.
   9. Please enter the following 'extra' attributes
   10. to be sent with your certificate request
   11. A challenge password []:subserverpassword
   12. An optional company name []:.
6. 自签署证书（使用自己生成的CA来签发）
   1. openssl x509 -req -in server/server-req.csr -out server/server-cert.pem -signkey server/server-key.pem -CA root/root-cert.pem -CAkey root/root-key.pem -CAcreateserial -days 3650
   2. 注意这里需要输入CA的证书、私钥、以及序列号。（因为这个证书需要CA签发，所以用CA私钥签名肯定是必须的）
7. 将证书导出为.p12格式
   1. openssl pkcs12 -export -clcerts -in server/server-cert.pem -inkey server/server-key.pem -out server/server.p12
   2. 导出密码：subserverpassword
8. **生成客户端证书**
9. 创建私钥：
   1. openssl genrsa -out client/client-key.pem 1024
10. 创建证书请求：
    1. openssl req -new -out client/client-req.csr -key client/client-key.pem
    2. Country Name (2 letter code) [AU]:CN
    3. State or Province Name (full name) [Some-State]:Beijing
    4. Locality Name (eg, city) []:Beijing
    5. Organization Name (eg, company) [Internet Widgits Pty Ltd]:smallclient
    6. Organizational Unit Name (eg, section) []:.
    7. Common Name (e.g. server FQDN or YOUR name) []:pp
    8. Email Address []:pppp@gmail.com
    9. Please enter the following 'extra' attributes
    10. to be sent with your certificate request
    11. A challenge password []:pppassword
11. 自签署证书（使用自己生成的CA来签发）：
    1. openssl x509 -req -in client/client-req.csr -out client/client-cert.pem -signkey client/client-key.pem -CA root/root-cert.pem -CAkey root/root-key.pem -CAcreateserial -days 3650
    2. Signature ok

subject=/C=CN/ST=Beijing/L=Beijing/O=smallclient/CN=pp/emailAddress=pppp@gmail.com

Getting Private key

Getting CA Private Key

1. 将证书导出成浏览器支持的.p12格式：
   1. openssl pkcs12 -export -clcerts -in client/client-cert.pem -inkey client/client-key.pem -out client/client.p12
   2. 导出密码:pppassword
2. **使用root证书生成jks文件以及tomcat的配置**
3. 生成root证书（CA）的jks文件
   1. keytool -import -v -trustcacerts -storepass jksstorepassword -alias root -file root/root-cert.pem -keystore root/root.jks (密码为jksstorepassword)
   2. Owner: CN=luca, OU=it, O=ccb, L=Beijing, ST=Beijing, C=CN
   3. Issuer: CN=luca, OU=it, O=ccb, L=Beijing, ST=Beijing, C=CN
   4. Serial number: e61892b304c29c85
   5. Valid from: Thu May 28 12:26:18 HKT 2015 until: Sun May 25 12:26:18 HKT 2025
   6. Certificate fingerprints:
   7. MD5: 4B:AF:6C:66:0C:02:89:CC:1A:5D:B2:75:98:AA:72:E0
   8. SHA1: 3E:28:DB:87:A2:29:F3:B9:CD:0E:8A:FE:54:B4:E9:43:03:81:43:99
   9. Signature algorithm name: SHA1withRSA
   10. Version: 1
   11. Trust this certificate? [no]: yes
   12. Certificate was added to keystore
   13. [Storing root/root.jks]
4. 配置tomcat
   1. 这一步省略了。目前的工程只在本地文件系统中获取证书文件来进行比较
   2. 如果需要连接服务器进行比较，就要配置tomcat并且连接到server
5. **导入p12证书文件到工程中，通过代码读入，然后输出字段**

* pem为openssl保存证书和密钥的文件格式。里面以64位data的形式存储了证书信息。
* p12也叫PKCS12，为浏览器，java，tomcat之类能识别的证书格式，苹果的证书也是以这样的格式存放。
* csr为certificate signing request，证书签发请求文件，需要用自己的私钥签名来表明自己的身份，并且附上一定的个人信息参数。
* clcerts表示只输出需要签发的客户端证书（申请人的证书），如果为cacerts，则只输出CA证书
* jks文件为java支持的私钥文件格式。相对应的还有微软支持的私钥文件格式pfx