## JDK key-tool生成证书

测试服务器IP：192.168.1.126

证书相关文件根目录：C:/Users/ftibw/Desktop/ssl/

key的alias：dmbd4

storepass：123456

nginx的https端口映射的静态资源目录：D: /static

1.生成RSA类型密钥对keystore

|  |
| --- |
| keytool -genkey -alias dmbd4 -keypass 123456 -keyalg RSA -keysize 2048 -validity 3650 -keystore C:/Users/ftibw/Desktop/ssl/server.keystore -storepass 123456 -ext SAN=ip:192.168.1.126 |

2.转换为pkcs12类型keystore

|  |
| --- |
| keytool -importkeystore -srckeystore C:/Users/ftibw/Desktop/ssl/keystore.keystore -destkeystore C:/Users/ftibw/Desktop/ssl/server.keystore -deststoretype pkcs12 |

3.将keystore导出为证书文件(server.cer)

|  |
| --- |
| keytool -export -alias dmbd4 -storepass 123456 -file C:/Users/ftibw/Desktop/ssl/server.cer -keystore C:/Users/ftibw/Desktop/ssl/server.keystore |

## OpenSSL证书转换

使用openssl将jdk生成的证书转化为nginx配置文件中可以使用的证书类型

1. server.cer文件转server.pem文件

|  |
| --- |
| x509 -inform der -in C:/Users/ftibw/Desktop/ssl/server.cer -out C:/Users/ftibw/Desktop/ssl/server.pem |

附:

openssl命令将cer格式转换crt

CER是二进制形式的X.509证书，DER编码，jdk的keytool导出的.cer文件就是这种格式。

CRT是二进制X.509证书，封装在文本（base-64）编码中。

如下两种方式进行转换

1.1.jdk的keytool导出的.cer文件

|  |
| --- |
| x509 -inform DER -in C:/Users/ftibw/Desktop/ssl/server.cer -out C:/Users/ftibw/Desktop/ssl/server\_cer.crt |

1.2.若.cer文件格式已经是.pem文件格式，就直接执行下面命令进行转换，否则先将.cer文件转换为.pem文件格式再转换成.crt文件，显然jdk生成的.cer文件需要先转换为.pem文件

|  |
| --- |
| x509 -inform PEM -in C:/Users/ftibw/Desktop/ssl/server.pem -out C:/Users/ftibw/Desktop/ssl/server\_pem.crt |

可以用上述2种方式生成的.crt文件相互验证，

第一个参数对应的.crt文件可验证第二个参数对应的.crt，反序后则情况相反。

|  |
| --- |
| verify -CAfile C:/Users/ftibw/Desktop/ssl/ server\_cer.crt C:/Users/ftibw/Desktop/ssl/ server\_pem.crt  C:/Users/ftibw/Desktop/ssl/server\_pem.crt: OK |

1. 使用java工具类，将server.keystore转为server.pfx

工具类如下：

|  |
| --- |
| public class ConvertPFXToKeystoreUtil {  public static final String PKCS12 = "PKCS12";  public static final String JKS = "JKS";  public static final String PFX\_KEYSTORE\_FILE = "C:\\Users\\ftibw\\Desktop\\ssl\\server.pfx";  public static final String KEYSTORE\_PASSWORD = "123456";  public static final String JKS\_KEYSTORE\_FILE = "C:\\Users\\ftibw\\Desktop\\ssl\\server.keystore";  /\*\*  \* 将pfx或p12的文件转为keystore  \*/  public static void coverTokeyStore() {  try {  KeyStore inputKeyStore = KeyStore.getInstance("PKCS12");  FileInputStream fis = new FileInputStream(PFX\_KEYSTORE\_FILE);  char[] nPassword = null;  nPassword = KEYSTORE\_PASSWORD.toCharArray();  inputKeyStore.load(fis, nPassword);  fis.close();  KeyStore outputKeyStore = KeyStore.getInstance("JKS");  outputKeyStore.load(null, KEYSTORE\_PASSWORD.toCharArray());  Enumeration enums = inputKeyStore.aliases();  // we are readin just one  // certificate.  while (enums.hasMoreElements()) {  String keyAlias = (String) enums.nextElement();  System.out.println("alias=[" + keyAlias + "]");  if (inputKeyStore.isKeyEntry(keyAlias)) {  Key key = inputKeyStore.getKey(keyAlias, nPassword);  Certificate[] certChain = inputKeyStore.getCertificateChain(keyAlias);  outputKeyStore.setKeyEntry(keyAlias, key, KEYSTORE\_PASSWORD.toCharArray(), certChain);  }  }  FileOutputStream out = new FileOutputStream(JKS\_KEYSTORE\_FILE);  outputKeyStore.store(out, nPassword);  out.close();  } catch (Exception e) {  e.printStackTrace();  }  }  /\*\*  \* 将keystore转为pfx  \*/  public static void coverToPfx() {  try {  KeyStore inputKeyStore = KeyStore.getInstance("JKS");  FileInputStream fis = new FileInputStream(JKS\_KEYSTORE\_FILE);  char[] nPassword = null;  nPassword = KEYSTORE\_PASSWORD.toCharArray();  inputKeyStore.load(fis, nPassword);  fis.close();  KeyStore outputKeyStore = KeyStore.getInstance("PKCS12");  outputKeyStore.load(null, KEYSTORE\_PASSWORD.toCharArray());  Enumeration enums = inputKeyStore.aliases();  // we are readin just one  // certificate.  while (enums.hasMoreElements()) {  String keyAlias = (String) enums.nextElement();  System.out.println("alias=[" + keyAlias + "]");  if (inputKeyStore.isKeyEntry(keyAlias)) {  Key key = inputKeyStore.getKey(keyAlias, nPassword);  Certificate[] certChain = inputKeyStore.getCertificateChain(keyAlias);  outputKeyStore.setKeyEntry(keyAlias, key, KEYSTORE\_PASSWORD.toCharArray(), certChain);  }  }  FileOutputStream out = new FileOutputStream(PFX\_KEYSTORE\_FILE);  outputKeyStore.store(out, nPassword);  out.close();  } catch (Exception e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  coverToPfx();  //coverTokeyStore();  }  } |

1. 使用server.pfx生成server.key文件

|  |
| --- |
| pkcs12 -in C:/Users/ftibw/Desktop/ssl/server.pfx -nocerts -nodes -out C:/Users/ftibw/Desktop/ssl/server.key |

4.配置nginx

|  |
| --- |
| http{  #其他配置省略  server {  listen 443 ssl;  server\_name localhost;  ssl\_certificate C:/Users/ftibw/Desktop/ssl/server.pem;  ssl\_certificate\_key C:/Users/ftibw/Desktop/ssl/server.key;  ssl\_session\_cache shared:SSL:1m;  ssl\_session\_timeout 5m;  ssl\_ciphers HIGH:MD5;  ssl\_prefer\_server\_ciphers on;    location / {  add\_header 'Access-Control-Allow-Origin' '\*';  root D: /static;  }  }  } |

5.将C:/Users/ftibw/Desktop/ssl/server.cer证书文件颁发给客户端

执行addcert.bat脚本，即可将证书导入到受信任的根证书颁发机构中。然后重启浏览器。

addcert.bat脚本：

|  |
| --- |
| @echo off  %1 mshta vbscript:CreateObject("Shell.Application").ShellExecute("cmd.exe","/c %~s0 ::","","runas",1)(window.close)&&exit  cd /d "%~dp0"  certutil -addstore -f root C:/Users/ftibw/Desktop/ssl/server.cer |