# java生成数字证书(4)

原文链接：<https://blog.csdn.net/happylee6688/article/details/42294037>

## 序

上一篇文章讲到了 Java 生成数字证书，使用的是第三方的组件 BC 。这篇文章也是介绍生成数字证书的，只不过与上一篇不同的是，这篇采用的是 KeyStore 的存储方式，导出的证书文件格式为 pfx ，这种格式的证书不仅包含有公钥，还包含有私钥。从证书中就可以读取到私钥。

## 正文

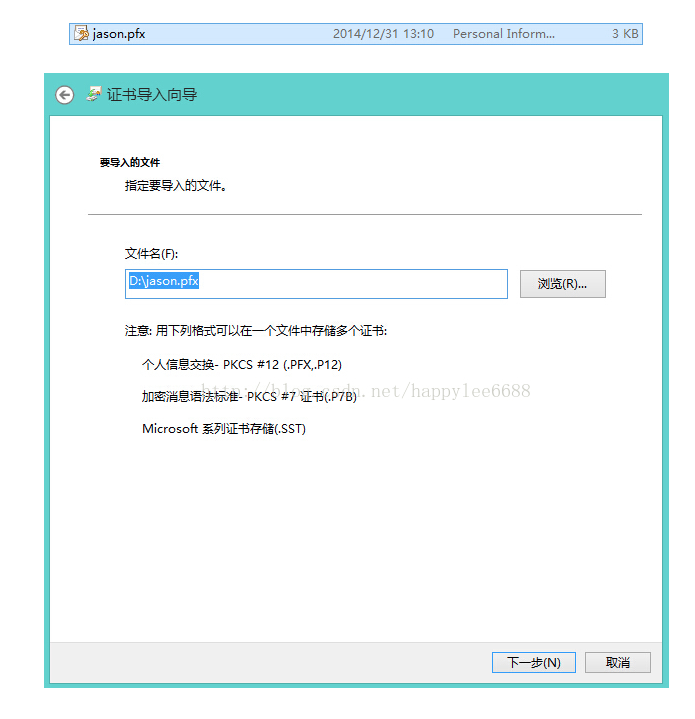
废话不多说，直接上内容。

与上一篇相同，这里也是使用的 Bouncy Castle 提供的组件，不同的是，这里的证书采用的是公钥加密技术12号标准生成的，简写 PKCS12 。具体内容这里就不再详细的介绍了，有需要的童鞋们直接 Google 就行了。下面直接上代码。

## PKCS12Test （测试证书类）

|  |
| --- |
| <span style="font-family:Comic Sans MS;font-size:12px;">package com.cacss.jsceu.core;    import org.bouncycastle.jce.provider.BouncyCastleProvider;  import org.bouncycastle.x509.X509V3CertificateGenerator;    import javax.security.auth.x500.X500Principal;  import java.io.FileOutputStream;  import java.io.IOException;  import java.math.BigInteger;  import java.security.\*;  import java.security.cert.Certificate;  import java.security.cert.CertificateException;  import java.security.cert.X509Certificate;  import java.util.Date;    /\*\*  \* Created With IntelliJ IDEA.  \*  \* @author : lee  \* @group : sic-ca  \* @Date : 2014/12/30  \* @Comments : 测试证书类  \* @Version : 1.0.0  \*/  public class PKCS12Test {    static {  // 系统添加BC加密算法 以后系统中调用的算法都是BC的算法  Security.addProvider(new BouncyCastleProvider());  }    public static void main(String args[]) throws NoSuchAlgorithmException,  InvalidKeyException, SecurityException, SignatureException,  KeyStoreException, CertificateException, IOException {    String certPath = "d:/jason.pfx";    // 创建KeyStore  KeyStore store = KeyStore.getInstance("PKCS12");  store.load(null, null);    /\* RSA算法产生公钥和私钥 \*/  KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");  kpg.initialize(2048);  KeyPair keyPair = kpg.generateKeyPair();  // 组装证书  String issuer = "C=CN,ST=BJ,L=BJ,O=SICCA,OU=SC,CN=SICCA";  String subject = issuer;    X509V3CertificateGenerator certGen = new X509V3CertificateGenerator();    certGen.setSerialNumber(BigInteger.valueOf(System.currentTimeMillis()));  certGen.setIssuerDN(new X500Principal(issuer));  certGen.setNotBefore(new Date(System.currentTimeMillis() - 50000));  certGen.setNotAfter(new Date(System.currentTimeMillis() + 50000));  certGen.setSubjectDN(new X500Principal(subject));  certGen.setPublicKey(keyPair.getPublic());  certGen.setSignatureAlgorithm("SHA256WithRSAEncryption");    X509Certificate cert = certGen.generateX509Certificate(keyPair.getPrivate());  System.out.println(cert.toString());  //System.out.println(keyPair.getPrivate());  //store.setCertificateEntry(alias, cert);    store.setKeyEntry("atlas", keyPair.getPrivate(),  "atlas".toCharArray(), new Certificate[] { cert });    FileOutputStream fout =new FileOutputStream(certPath);  store.store(fout, "atlas".toCharArray());  fout.close();  }    /\*\*  \* 得到key store  \*  \* @return  \* @throws Exception  \*/  private KeyStore getKeyStore() throws Exception {    KeyStore store = KeyStore.getInstance("PKCS12");  store.load(null, null);  return store;  }  }</span> |

## 效果图



## 结束语

如果只看证书文件的话，是没有什么区别的，只不过嘛，在 windows 下这两种证书的打开方式是不一样的，大家可以自己去体会一下。

值得指出的一点是，使用密钥库的方式存储证书，是一个很不错的方案，当然，如果涉及到安全的问题，可以对密钥库进行加密，在写入证书的时候，设置一种比较安全的加密方式，生成的证书就会有访问限制，需要提供访问密码。没有密码的人是访问不了的。当然，我提供的这个例子中是没有经过加密存储的。有这方面需求的童鞋，可以看一下 Java api 中的相关介绍。