**采用CA搭建Tomcat双向SSL认证**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 修订内容 | 修订日期 | 版本号 | 修订人 | 备注 |
|  | 完成初始版本 | 2017.07.31 | 1.0 |  |  |
|  |  |  |  |  |  |

目录

[1. 概述 4](#_Toc489371487)

[1.1 目的 4](#_Toc489371488)

[1.2 架构 4](#_Toc489371489)

[1.3 CA 5](#_Toc489371490)

[1.4 SSL认证 5](#_Toc489371491)

[2. CA签发证书 6](#_Toc489371492)

[2.1 工具 XCA 和 portecle 6](#_Toc489371493)

[2.2 XCA创建数据库 6](#_Toc489371494)

[2.3 自签CA根证书 7](#_Toc489371495)

[2.4 签发服务器证书 9](#_Toc489371496)

[2.5 签发个人证书 13](#_Toc489371497)

[2.6 根证书格式转换 16](#_Toc489371498)

[3. Tomcat配置 21](#_Toc489371499)

[3.1 HTTPS证书登录配置 21](#_Toc489371500)

[3.2 HTTPS自动跳转配置 22](#_Toc489371501)

[4. IE导入证书 23](#_Toc489371502)

[4.1 IE导入根证书： 23](#_Toc489371503)

[4.2 导入用户端证书： 24](#_Toc489371504)

[5. IE证书登录 26](#_Toc489371505)

[5.1 Java从HTTPS请求获取证书代码 26](#_Toc489371506)

[5.2 测试 28](#_Toc489371507)

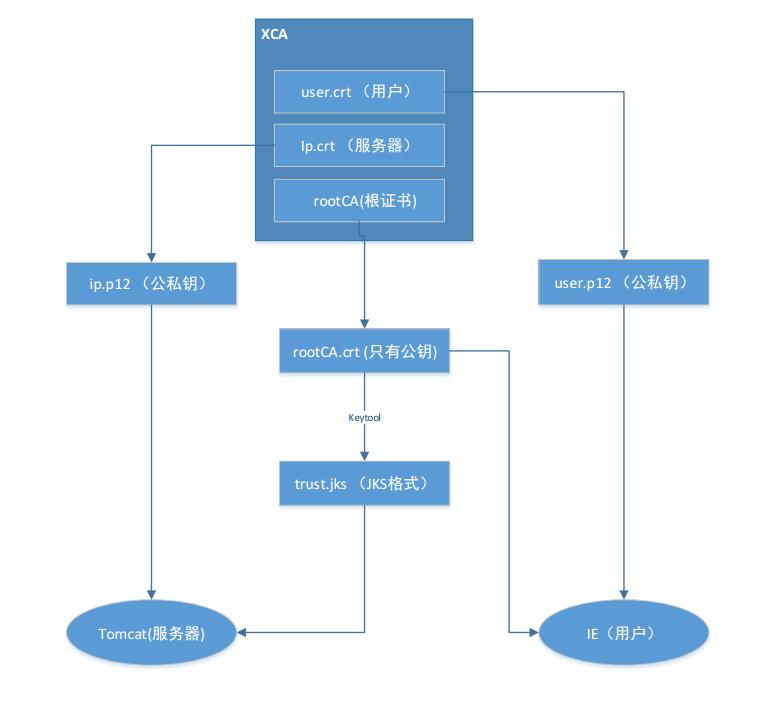
# 概述

## 目的

访问服务器时自动跳转Https加密访问。

登录系统时，根据证书登录，免除输入用户名/密码。

## 架构



## CA

CA是一个机构，是负责签发证书、认证证书、管理已颁发证书的机关。

作用：中间人，证明某用户是某用户，证明某服务器是某服务器。保证用户或者服务器的真实性。

项目中可采用两种方式使用CA：

一是采用国际知名CA，付费；

二是自己搭建CA，免费。

项目中采用第二种。

## SSL认证

SSL认证是在Http的基础上采用ssl（secure Sockets Layer ）协议，进行加密。

采用<https://x.x.x.x>进行访问。

作用：1）认证用户和服务器，确保数据发送到正确的[客户机](https://baike.baidu.com/item/%E5%AE%A2%E6%88%B7%E6%9C%BA" \t "_blank)和[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8)；

2）加密数据以防止数据中途被窃取；

3）维护数据的完整性，确保数据在传输过程中不被改变。

# CA签发证书

## 工具 XCA 和 portecle

XCA：

https://sourceforge.net/projects/xca/files/latest/download

功能：自签CA根证书，管理证书的创建和删除。

portecle：

https://sourceforge.net/projects/portecle/files/portecle/1.10/portecle-1.10.zip/download

功能：格式转换。

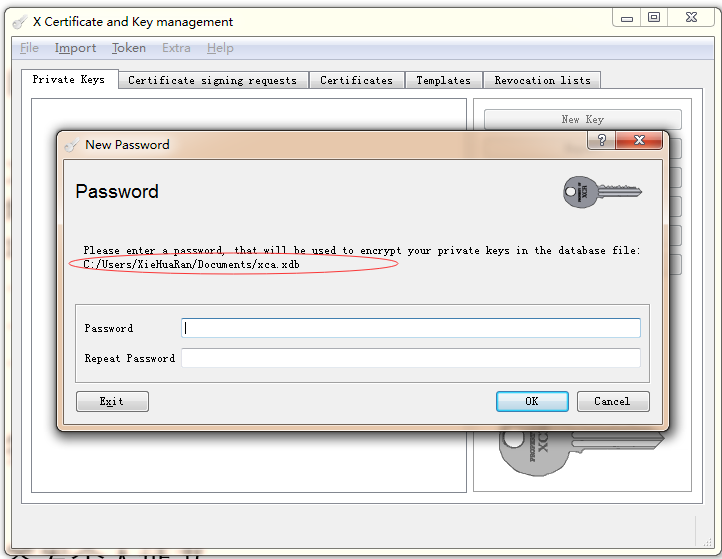
注：直接解压，portecle双击portecle.jar即可运行。

## XCA创建数据库

使用XCA时，首先需要先创建一个数据库。存储相关根证书和根证书相关联的证书，便于多证书的管理。如图所示：

1. 创建xca的数据库。File->New DataBase-> 路径/文件名，保存。

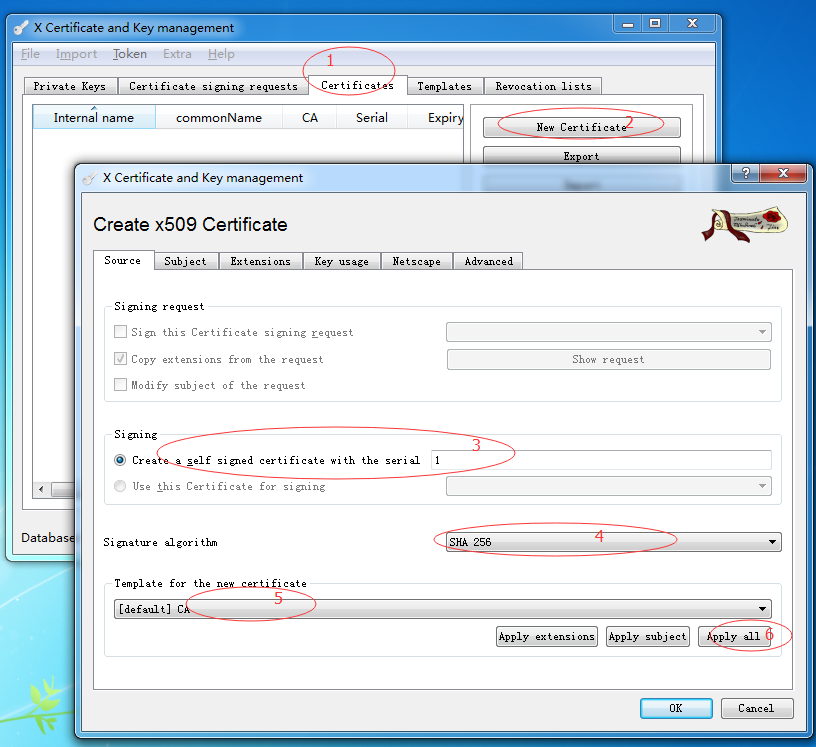
这个密码是打开这个数据库所属的证书时，需要验证的密码。



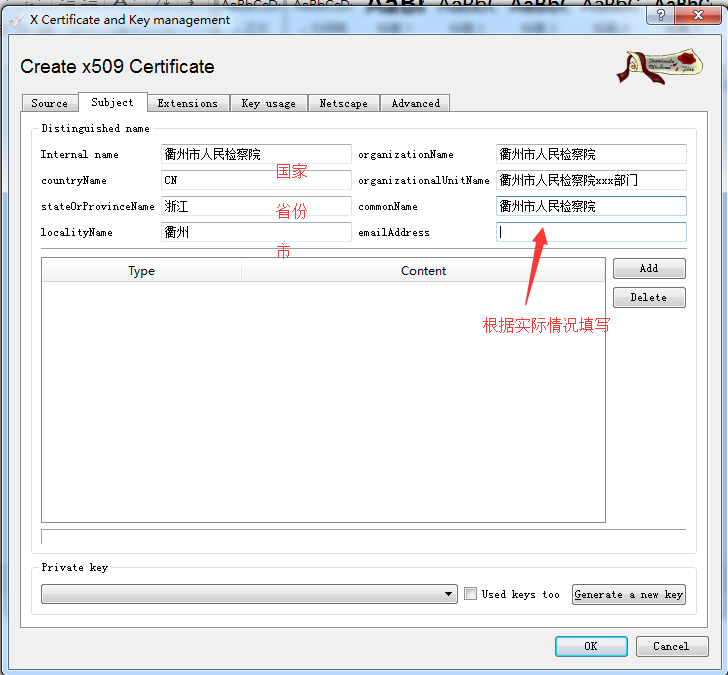
2. 输入密码进行保存，此密码用户下次访问该数据库时要输入的密码。

## 自签CA根证书

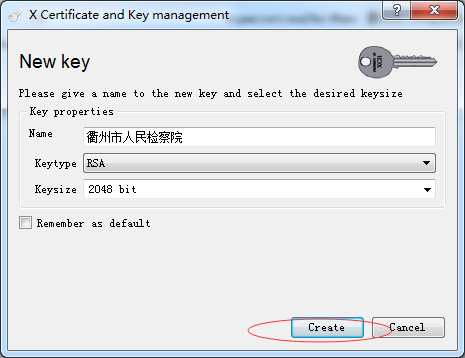
1. 因为要创建根证书，这里选择序号为1的自认证证书，签名算法选择SHA 256，证书模版选择默认CA，再点击Apply all（这个不能漏）



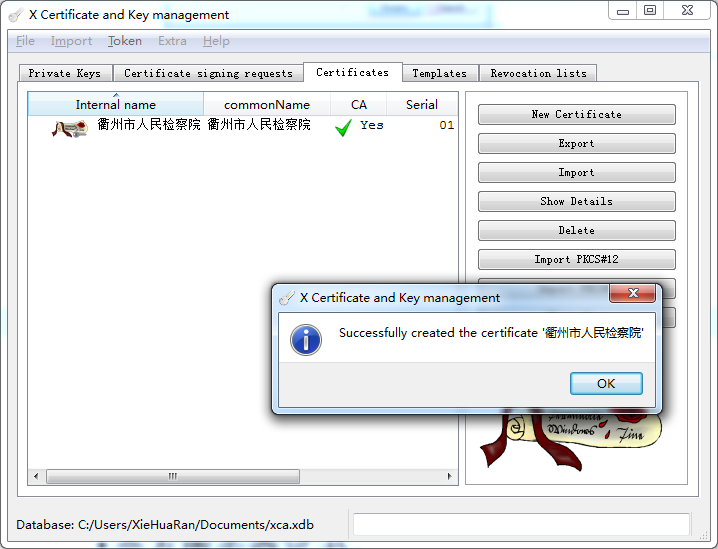
2. 切换到subject页面，填写根证书的内容。

CommonName根据实际填写，可填写至部门。

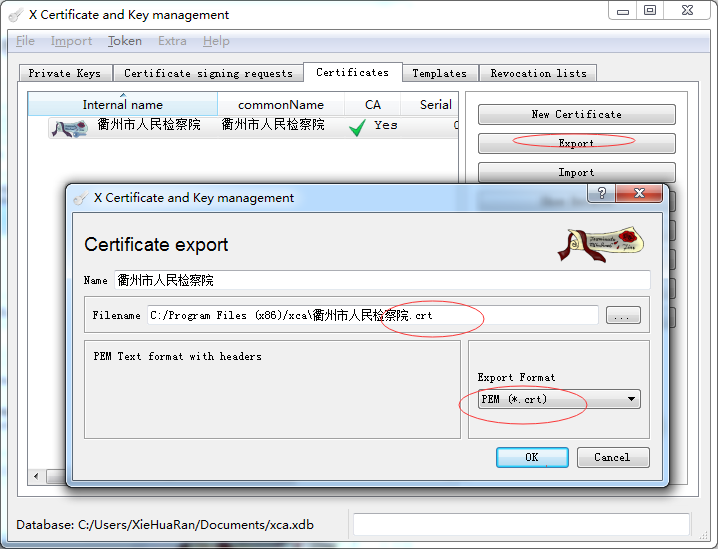
3. 再点击Generate a new key



4. 最后点击OK，CA证书做好了，有效期默认10年

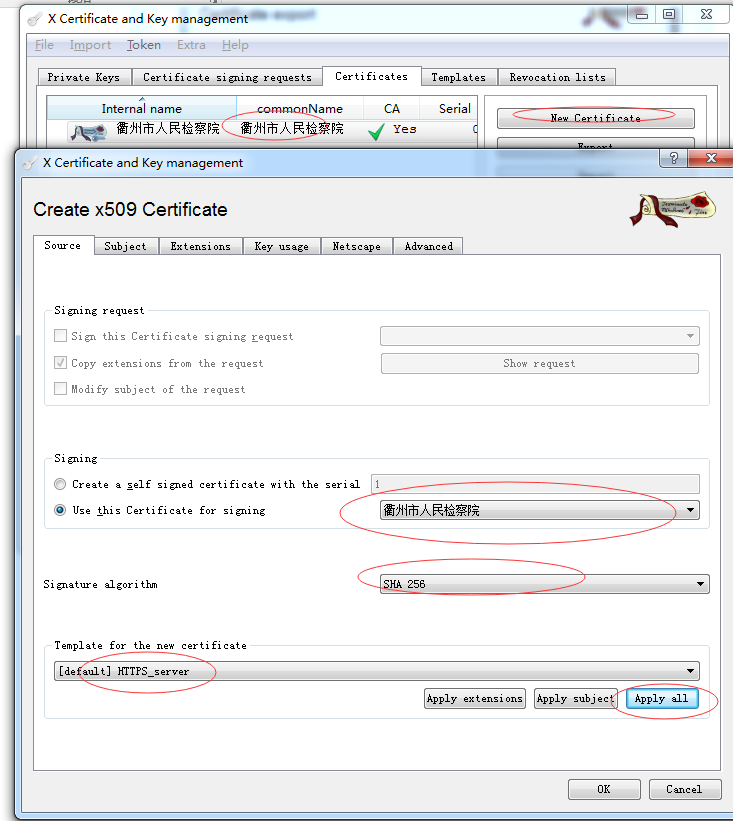


5. 将根证书导出成只包含公钥的证书格式，这本根证书就是放在网站上供用户[下载](http://www.2cto.com/soft" \t "_blank)安装，或主动安装到客户机器中的。

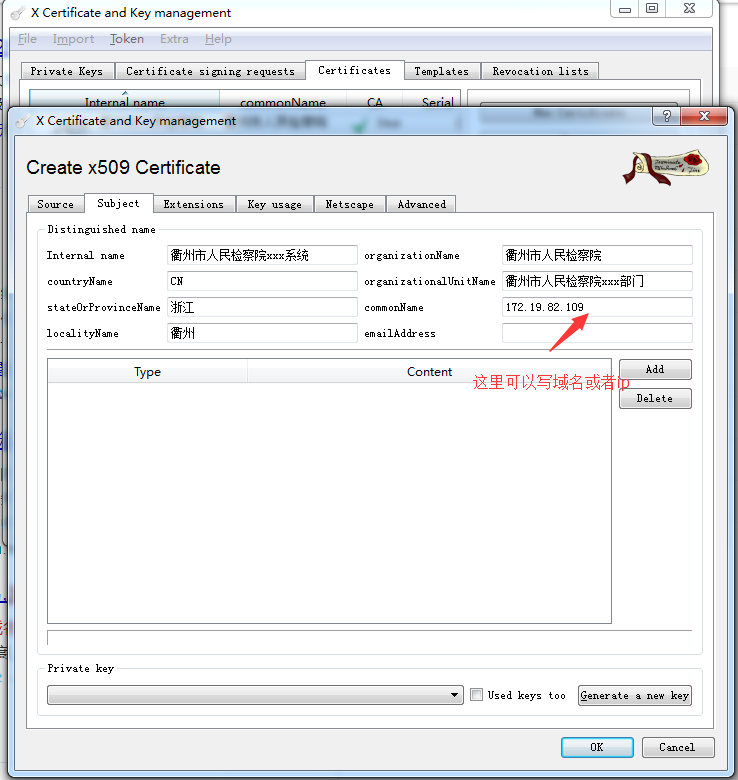


## 签发服务器证书

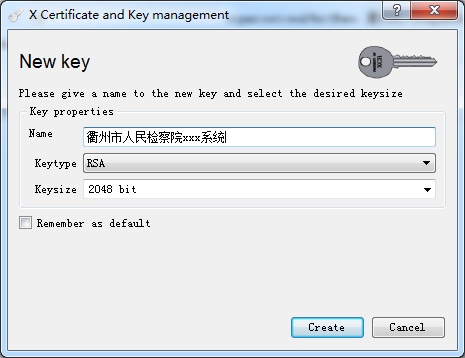
1. 选中“衢州市人民检察院”证书->New Certificate



2. 切换到subject页面；这里InternalName可以写到xx系统，方便证书的管理。根据实际情况，这个页面随便填写。ComomName可以写ip地址或者域名。ip指的是部署项目的服务器地址。本文中写IP。



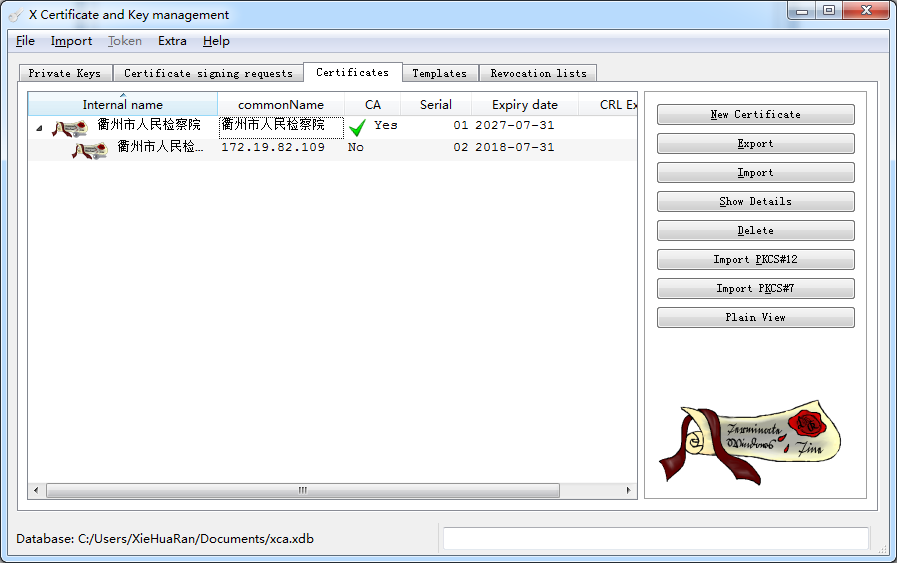
3. 生成私钥->Generate a new key-ok



4. 切换到Extensions,可修改服务器证书有效期

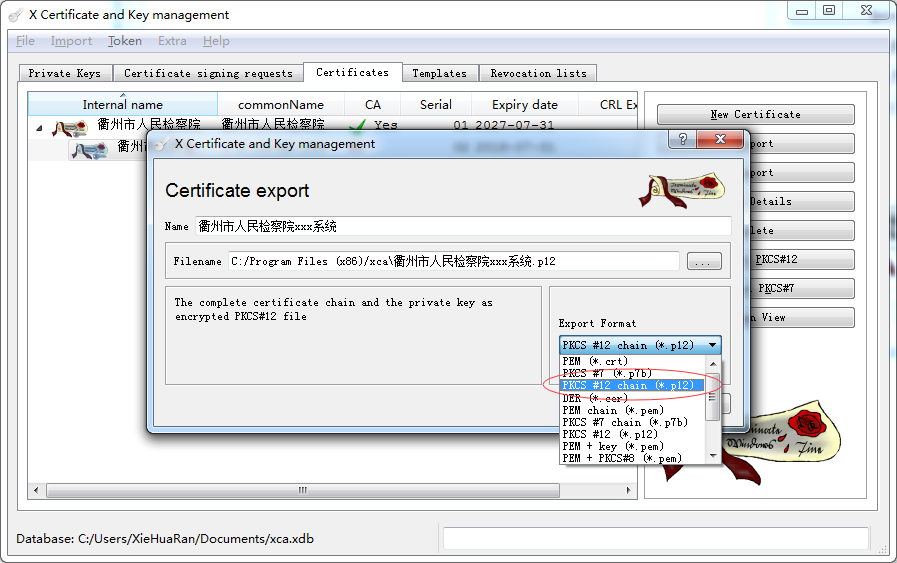


5. 点击OK，创建服务器证书完毕。

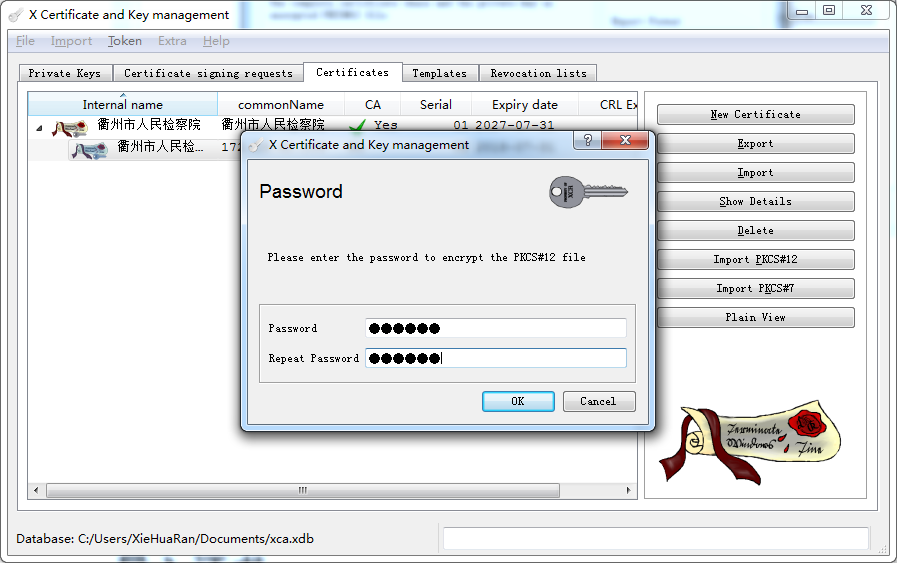


6. 导出

注意格式，要导出为PKCS #12 chain格式

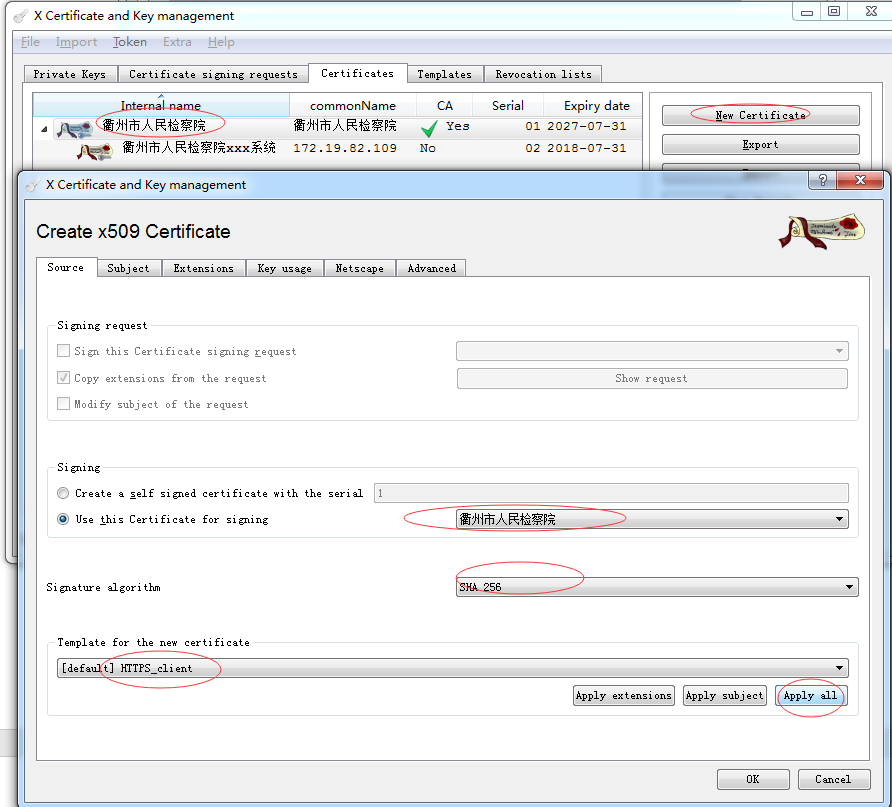


7. 导出时创建密码。这个密码用于导入证书

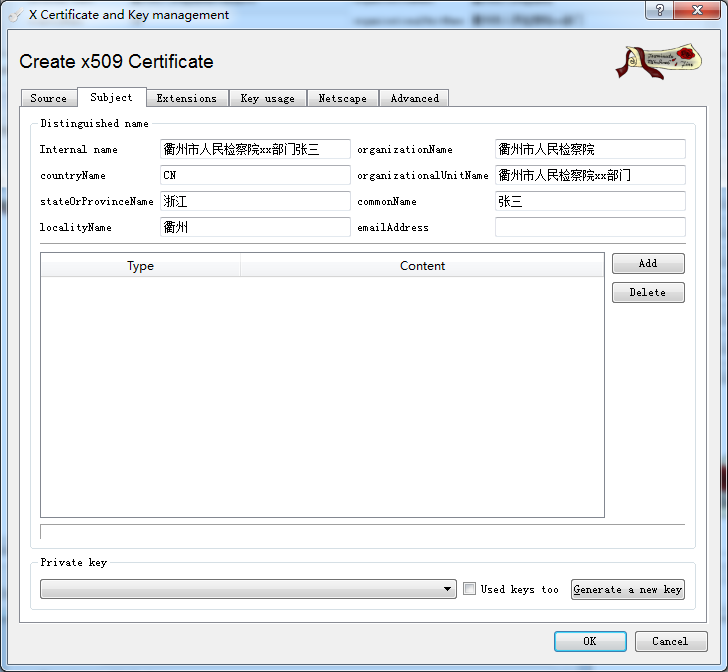


## 签发个人证书

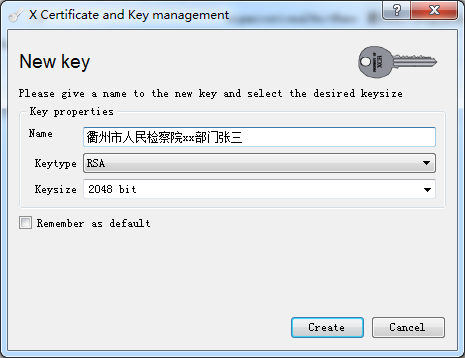
1. 选中“衢州市人民检察院”根证书->New Certificate



2. 切换subject，CommonName填写为实际访问系统的用户。和数据库表tb\_users，username字段的值一一对应。

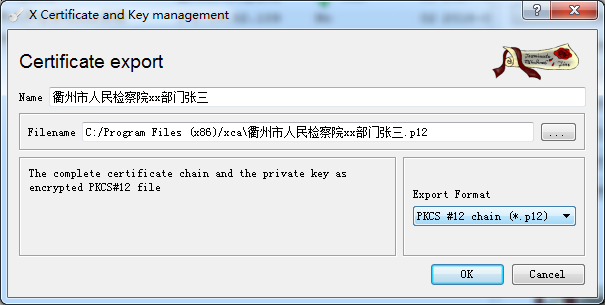


3. 生成私钥 -> Generate a new key



4. 导出：

注意格式，要导出为PKCS #12 chain格式

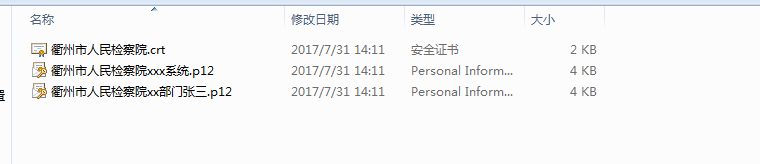


## 根证书格式转换

因为tomcat不支持.crt格式的证书，所以需要转换为java原生的JKS格式。

需要把[衢州市人民检察院.crt]格式转换为JKS格式。

目前为止，我们现在有三张证书。

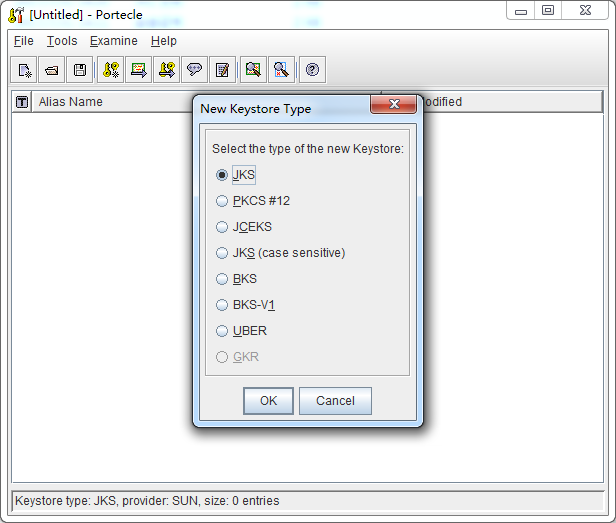


工具：portecle

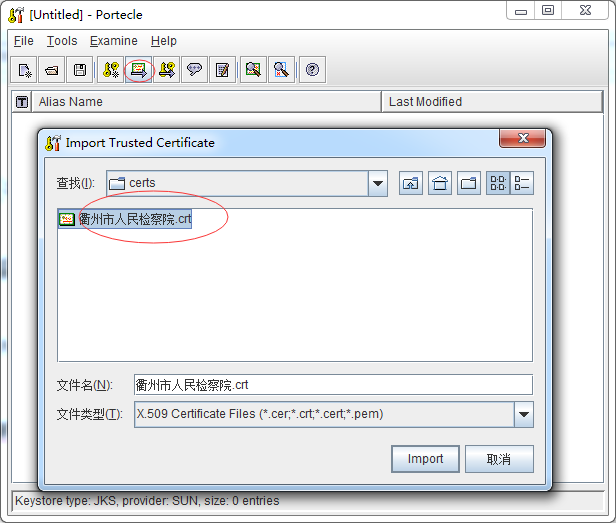
步骤：

1. 创建新的存储秘钥文件

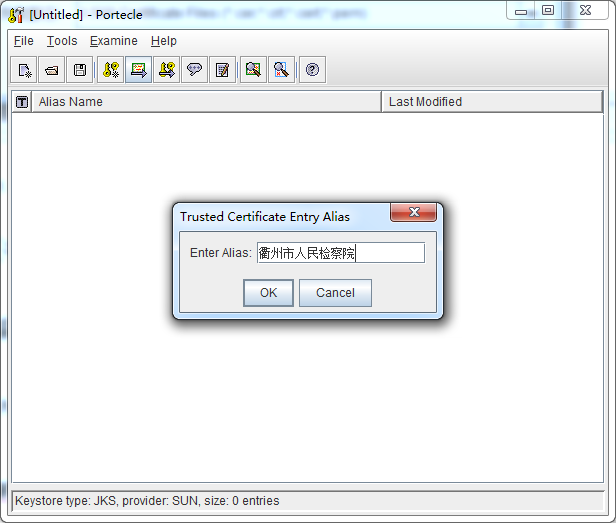
File -> new KeyStore ，选择JKS，点击ok



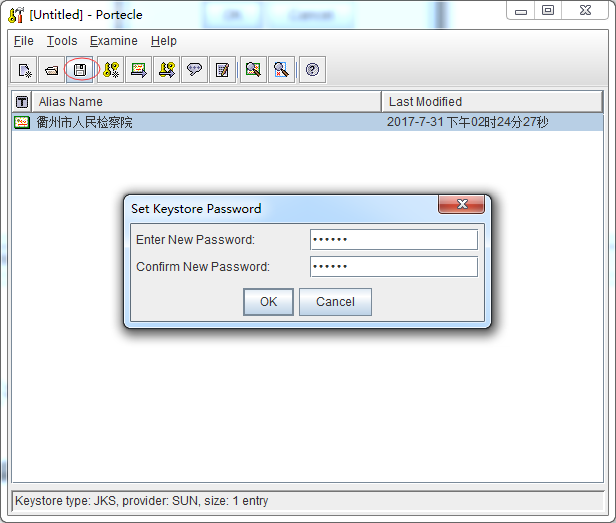
2. 导入：import ->确定 ->OK ->是



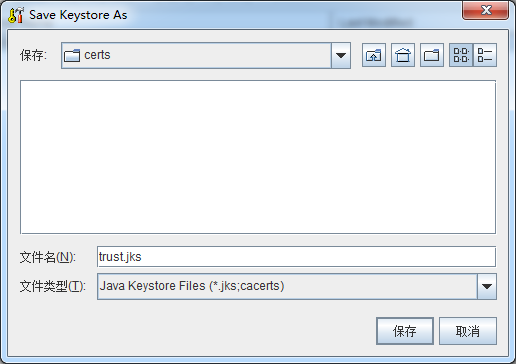
3. 输入名字



4. 保存。密码对应于导入时输入的密码。



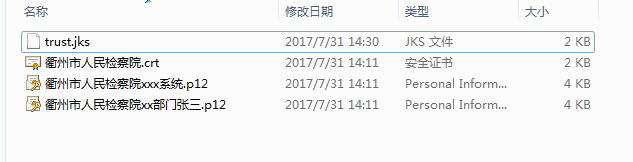
5.保存



注：

现在我们有四个证书。

其中trust.jks和衢州市人民检察院.crt其实是一个证书，格式不同而已。



# Tomcat配置

## HTTPS证书登录配置

服务器端需要两个证书：

①根证书 --> trust.jks

②服务器端证书--> 衢州市人民检察院xxx系统.p12

主要配置：

1. keystoreFile：CA签发的服务器端证书。

2. truststoreFile：CA自签的证书，即受信任的证书。

3. clientAuth：true：双向认证；false：只验证服务器；want：服务器不强制验证客户端，客户端有证书则验证，没有不验证。

具体如下：[tomcat/conf/servlet.xml]

|  |
| --- |
| <Connector  port="8443"  protocol="org.apache.coyote.http11.Http11Nio2Protocol"  keystoreFile="e:\certs\衢州市人民检察院xxx系统.p12" //服务器端的根证书  keystorePass=123456" //保存证书时候输入的那个密码  keystoreType="PKCS12"  truststoreFile="e:\certs\trust.jks" //受信任的根证书  truststorePass="123456" //保存证书时候要输入的密码  truststoreType="jks "  clientAuth="true" ， //ture false want  scheme="https"  sslProtocol="TLS"  secure="true"  SSLEnabled="true"  maxThreads="150"  minSpareThreads="25"  connectionTimeout="20000"  enableLookups="false"  maxHttpHeaderSize="8192"  useBodyEncodingForURI="true"  acceptCount="100"  disableUploadTimeout="true"/> |

## HTTPS自动跳转配置

实现输入<http://172.19.82.109:8080-->>自动转换为[https:// 172.19.82.109:8443](https://xx.x.x.x.x:8443)

在tomcat/conf/web.xml的<web-app>

|  |
| --- |
| <web-app>  ......  <!-- Authorization setting for SSL -->  <login-config>  <auth-method>CLIENT-CERT</auth-method>  <realm-name>Client Cert Users-only Area</realm-name>  </login-config>    <!-- Authorization setting for SSL -->  <security-constraint>  <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>  </web-app> |

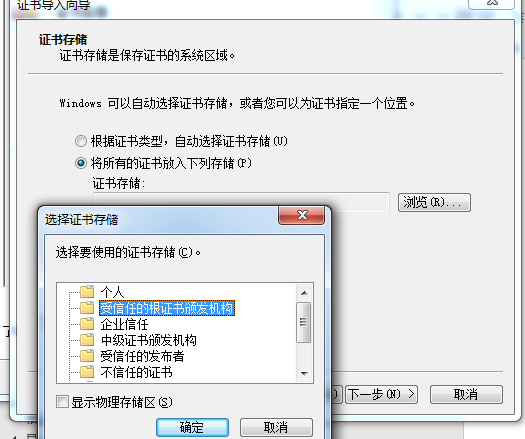
# IE导入证书

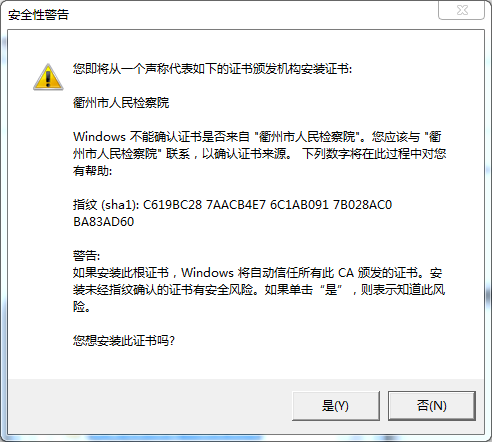
①根证书 --> 衢州市人民检察院.crt

②用户端证书 --> 衢州市人民检察院xx部门张三.p12

## IE导入根证书：

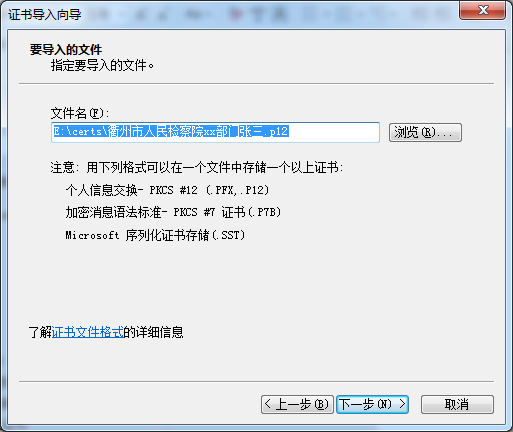
1.双击衢州市人民检察院.crt 🡪安装证书🡪下一步🡪受信任的根证书颁发机构🡪下一步🡪完成->选择“是”



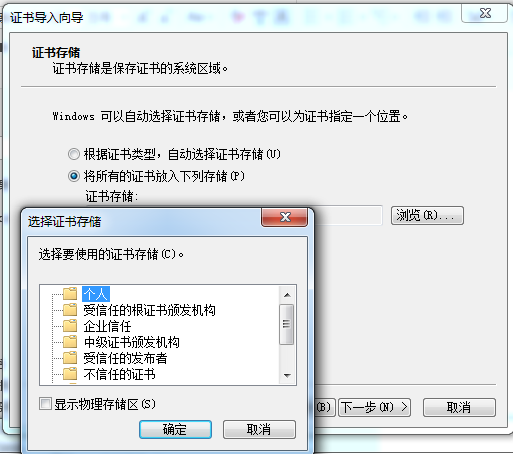


## 导入用户端证书：

1. 双击文件



2. 输入保存证书时的密码->选择“个人”



# IE证书登录

## Java从HTTPS请求获取证书代码

//首先从请求中获取证书，根据证书去查找用户，如果存在这个用户那就采用证书登录。

//如果用户不存在，采用用户名和密码方式登录。

@RequestMapping("login")

public String login(ModelMap map, HttpServletRequest request) {

Map<String, Object> user = getUserByCert(request);

if (user == null) {

logger.info("客户端采用【账号/密码】方式登录！");

return "login";

}

logger.info("客户端采用【数字证书】方式登录！");

String username = (String) user.get("username");

String password = (String) user.get("password");

String remember = "false";

doLogin(username, password, remember);

//加载菜单

Map<String, Object> params = new HashMap<>();

params.put("user\_id", getCurrentUser().get("user\_id"));

List<Map<String, Object>> menus = userService.getUserRoleMenu(params);

map.put("menus", menus);

return "index";

}

//通过证书获取用户

private Map<String, Object> getUserByCert(HttpServletRequest request) {

X509Certificate[] certChain = (X509Certificate[]) request.getAttribute("javax.servlet.request.X509Certificate");

if (certChain == null) {

return null;

}

if (certChain.length < 1) {

return null;

}

X500Principal pSubject = certChain[0].getSubjectX500Principal();

X500Name x500Name = null;

try {

x500Name = new X500Name(pSubject.getName());

} catch (IOException e) {

logger.info("获取证书异常\n");

e.printStackTrace();

return null;

}

String commonName;

try {

commonName = x500Name.getCommonName();

} catch (IOException e) {

logger.info("验证证书失败！");

e.printStackTrace();

return null;

}

if (commonName == null || commonName.isEmpty()) {

return null;

}

Map<String, Object> param = new HashMap<String, Object>();

param.put("username", commonName);

return userService.getUserInfo(param);

}

## 测试

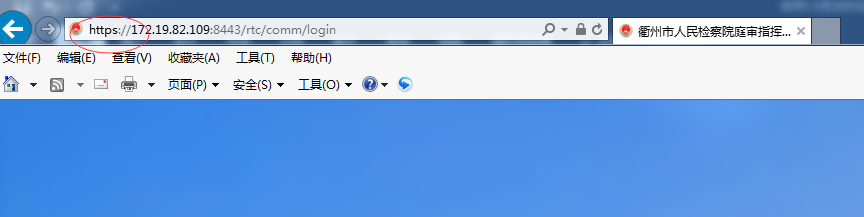
实现的效果：

1. 地址栏输入<http://172.82.19.109:8080/comm/login>，

自动跳转至<https://172.82.19.109:8443/comm/login>

注：如果一个用户有多张证书，会选择用哪一张证书登录。

如图：



2. 如果用户存在证书，直接登录；不存在证书，采用用户名/密码登录。