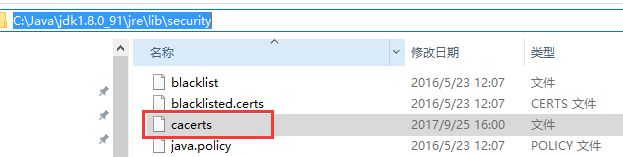
这里主要有两件事要做：

**第一个就是给客户端的jre添加和服务端对应的证书**

之前我们已经生成了一个keystore，我们就直接使用这个keystore生成一个证书，命令如下：

keytool -exportcert -alias cas -keystore D:/liuyx.keystore -file D:/liuyx.keystore.cer -storepass 123456

然后把这个证书导入jre的



注意，这个jre**一定**要是你跑客户端的Tomcat的那个jre，

这里我的目录是：C:/Java/jdk1.8.0\_91/jre/lib/security/cacerts，刚刚导出的证书的目录是D:/liuyx.keystore.cer

那么我的命令就是：

keytool -import -alias cas -keystore C:/Java/jdk1.8.0\_91/jre/lib/security/cacerts -file D:/liuyx.keystore.cer

证书库的密码是：changeit

导入完我们可以看一下导入的结果：

keytool -list -keystore C:/Java/jdk1.8.0\_91/jre/lib/security/cacerts

密码同上，可以看到一堆证书，其中有一个“别名”是cas的，日期是今天，就是我们刚刚导进去的。

当我们做完这一切CAS实验后，也许你会想要删掉这个证书，可以使用这个命令：

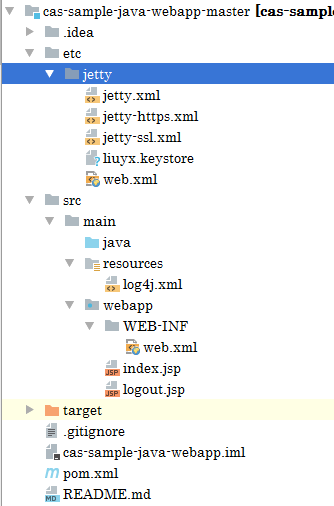
keytool -delete -alias cas -keystore C:/Java/jdk1.8.0\_91/jre/lib/security/cacerts

**接下来就是跑官网的示例了**

官网例子地址：

<https://github.com/cas-projects/cas-sample-java-webapp>

我们先把它下下来，结构如下



可以看到里面内置了jetty，不过我是通过idea中的Tomcat运行的，我把它导入了idea。

然后就是我们的正题了，配置修改：

1、pom.xml

在pom里面加上以下内容：

[复制代码](javascript:void(0);)

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.25</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.25</version>

</dependency>

[复制代码](javascript:void(0);)

2、web.xml

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd">

<!--用来控制cas识别的session的保存，以及判断是否是登出请求-->

<filter>

<filter-name>CAS Single Sign Out Filter</filter-name>

<filter-class>org.jasig.cas.client.session.SingleSignOutFilter</filter-class>

<init-param>

<param-name>casServerUrlPrefix</param-name>

<!--这个地址要和keystore中的CN一致，端口无所谓，域名必须一致，前边我在hosts里面配置了这个域名映射，所以实际上访问的是127.0.0.1-->

<param-value>https://cas.example.org:8443</param-value>

</init-param>

</filter>

<listener>

<listener-class>org.jasig.cas.client.session.SingleSignOutHttpSessionListener</listener-class>

</listener>

<!--用来跳转登录-->

<filter>

<filter-name>CAS Authentication Filter</filter-name>

<filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>

<init-param>

<param-name>casServerLoginUrl</param-name>

<!--这个地址要和keystore中的CN一致，端口无所谓，域名必须一致，前边我在hosts里面配置了这个域名映射，所以实际上访问的是127.0.0.1-->

<param-value>https://cas.example.org:8443/login</param-value>

</init-param>

<init-param>

<param-name>serverName</param-name>

<!--这是你客户端的部署地址，认证时会带着这个地址，认证成功后会跳转到这个地址-->

<param-value>http://localhost:8080</param-value>

</init-param>

</filter>

<!--用来验证ticket-->

<filter>

<filter-name>CAS Validation Filter</filter-name>

<filter-class>org.jasig.cas.client.validation.Cas30ProxyReceivingTicketValidationFilter</filter-class>

<init-param>

<param-name>casServerUrlPrefix</param-name>

<!--这个地址要和keystore中的CN一致，端口无所谓，域名必须一致，前边我在hosts里面配置了这个域名映射，所以实际上访问的是127.0.0.1-->

<param-value>https://cas.example.org:8443</param-value>

</init-param>

<init-param>

<param-name>serverName</param-name>

<!--这是你客户端的部署地址，验证ticket成功后会跳转到这个地址-->

<param-value>http://localhost:8080</param-value>

</init-param>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

<init-param>

<param-name>redirectAfterValidation</param-name>

<param-value>true</param-value>

</init-param>

<init-param>

<param-name>useSession</param-name>

<param-value>true</param-value>

</init-param>

<init-param>

<param-name>authn\_method</param-name>

<param-value>mfa-duo</param-value>

</init-param>

</filter>

<!--用来封装request-->

<filter>

<filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>

<filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>CAS Single Sign Out Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>CAS Validation Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>CAS Authentication Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<welcome-file-list>

<welcome-file>

index.jsp

</welcome-file>

</welcome-file-list>

</web-app>

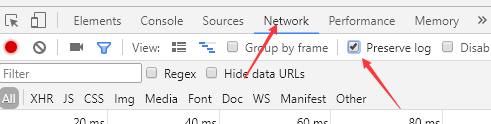
[复制代码](javascript:void(0);)

总共四个过滤器和一个监听器，大部分都在文件内容中注释说明了，自己看一下吧。

客户端应用跑起来，会先跳转到CAS服务器登录页，登录成功后会跳转到首页index.jsp，里面就是获取了所有你可以从CAS服务器获取的属性，如果你前面没有配置多属性获取，那么只有一个默认的id，和一些其他cas自带的属性。

关于如何在代码里获得这些属性，这个index.jsp 的代码里已经写的非常清楚了，自己去看一下即可。

最后建议大家做完这些之后自己打开谷歌浏览器的F12，勾选上Preserve log



通过正常的登陆来看一下页面跳转流程，结合下边这个地址，弄明白TGC、TGT、ST三个概念：

<http://www.cnblogs.com/flying607/p/5626088.html>