## 一、cmd

窗口执行，生成密钥

keytool -genkey -alias symbol -keyalg RSA -keystore d:/keys/symbolkey

## 二、在此目录下生成证书

D:\keys

执行以下语句，生成证书

keytool -export -file d:/keys/symbol.crt -alias symbol -keystore d:/keys/symbolkey

## 三、为客户端的JVM导入证书

keytool -import -keystore "%JAVA\_HOME%\jre\lib\security\cacerts" -file D:/keys/symbol.crt -alias symbol

{删除命令

      keytool -delete -alias symbol -keystore "%JAVA\_HOME%\jre\lib\security\cacerts "

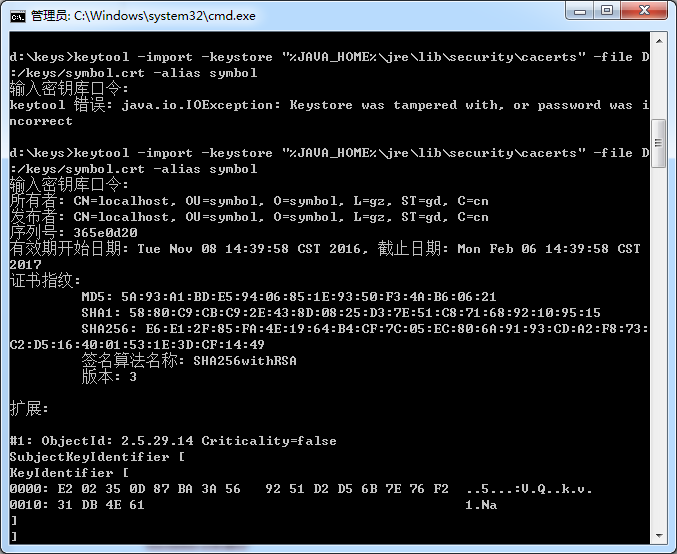
     -storepass changeit

}

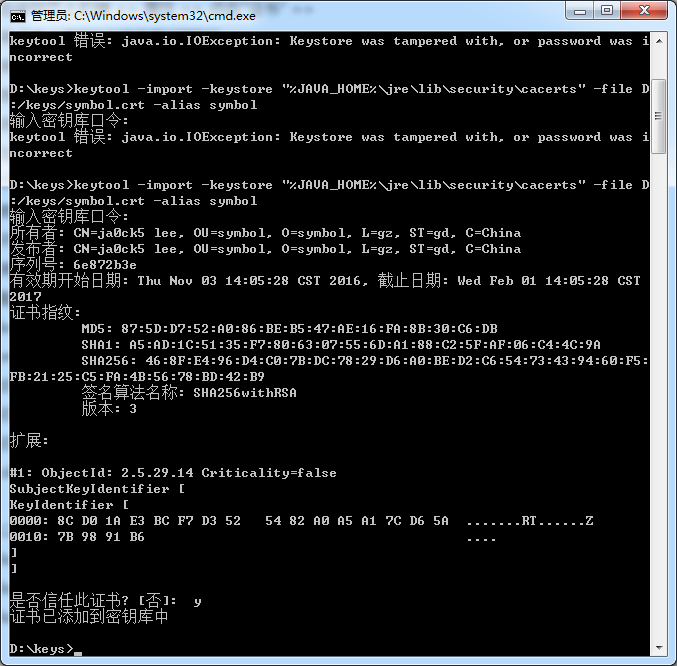
此时命令行会提示你输入cacerts证书库的密码，敲入changeit就行了，这是java中cacerts证书库的默认密码

结果如下

## New

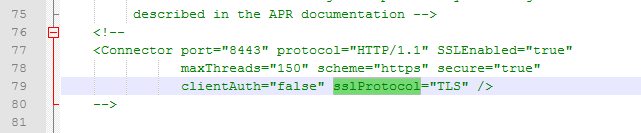


## old

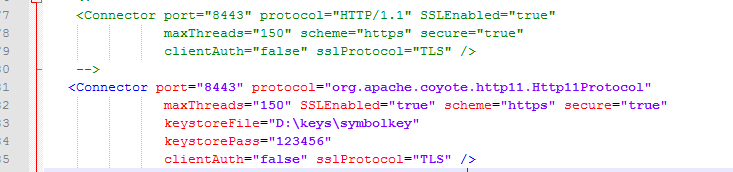


## 四、配置tomcat

　　　　在%TOMCAT\_PATH%/conf下找到server.xml文件，定位到下面的代码上



(建议是复制注释里的内容到下方)打开注释，修改成如下:



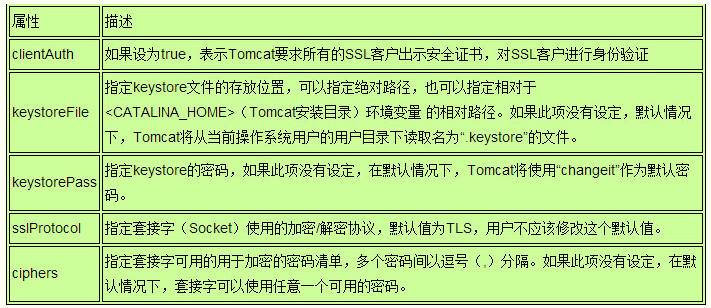
<Connector SSLEnabled="true" acceptCount="100" clientAuth="false"

disableUploadTimeout="true" enableLookups="false" maxThreads="25"

port="8443" keystoreFile="D:\keys\symbolkey" keystorePass="123456"

protocol="org.apache.coyote.http11.Http11NioProtocol" scheme="https"

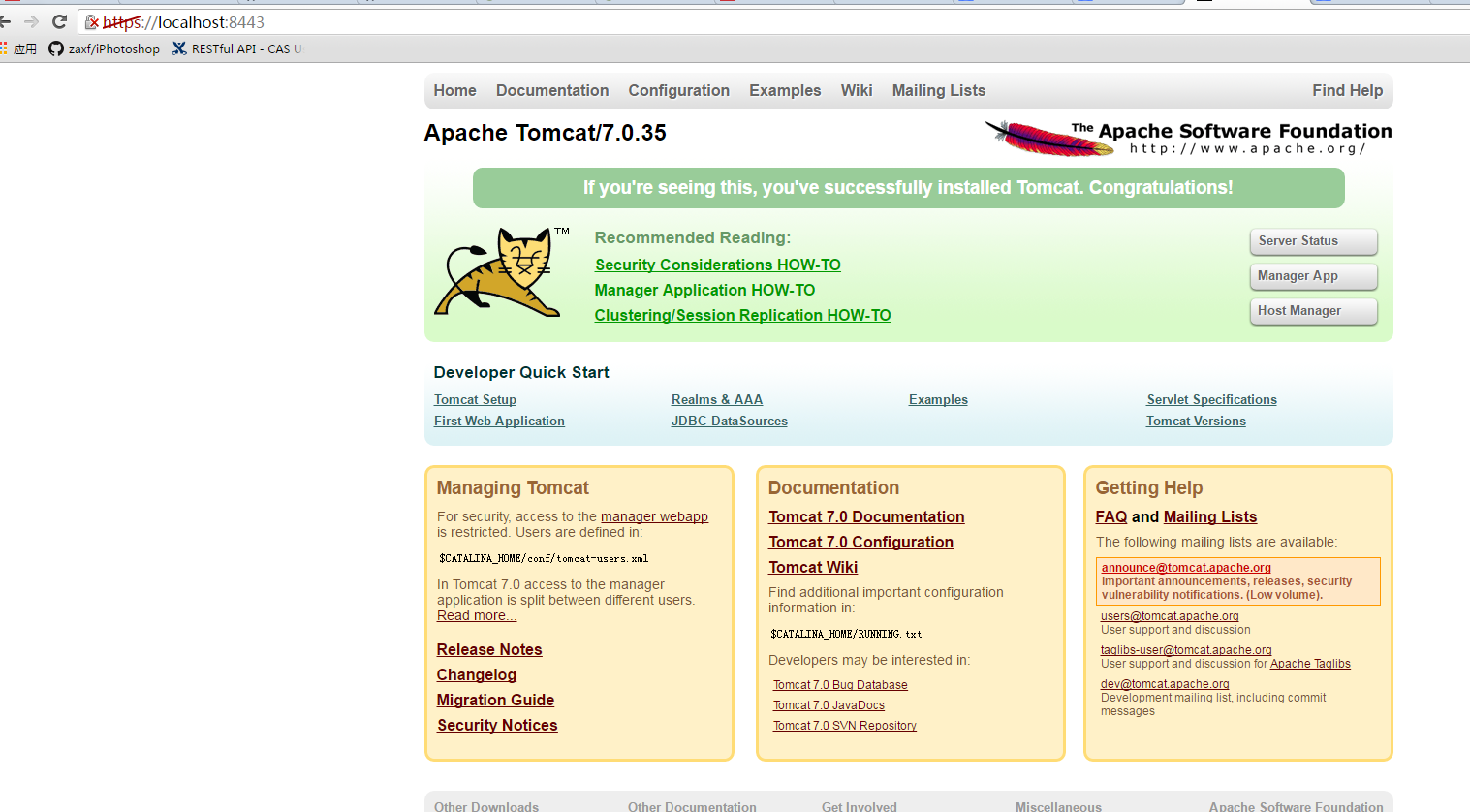
secure="true" sslProtocol="TLS" />



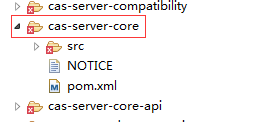
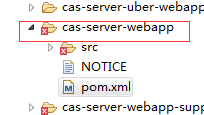
启动tomcat:

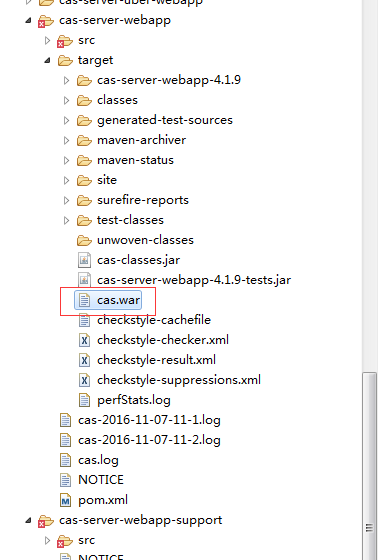


访问：https://localhost:8443/

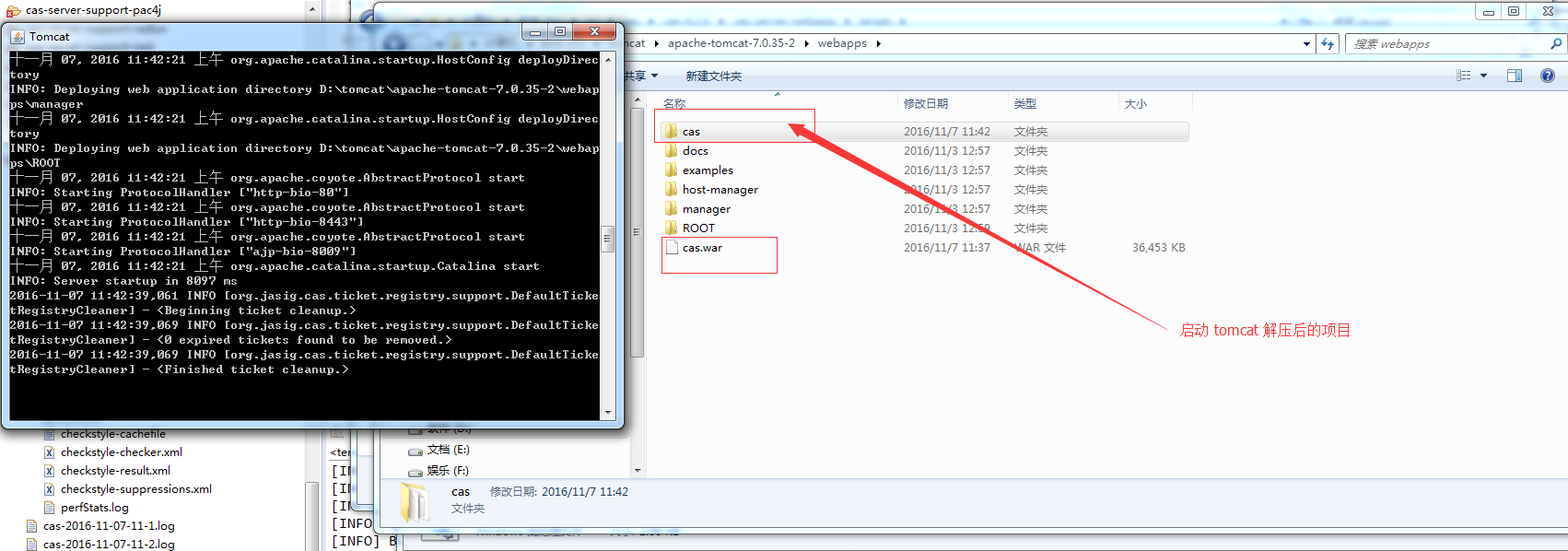


## 整个过程可以在 cas-server 源码上注释掉 pom 文件不需要的模块也可以，为了不影响源码则独立抽出并测试。

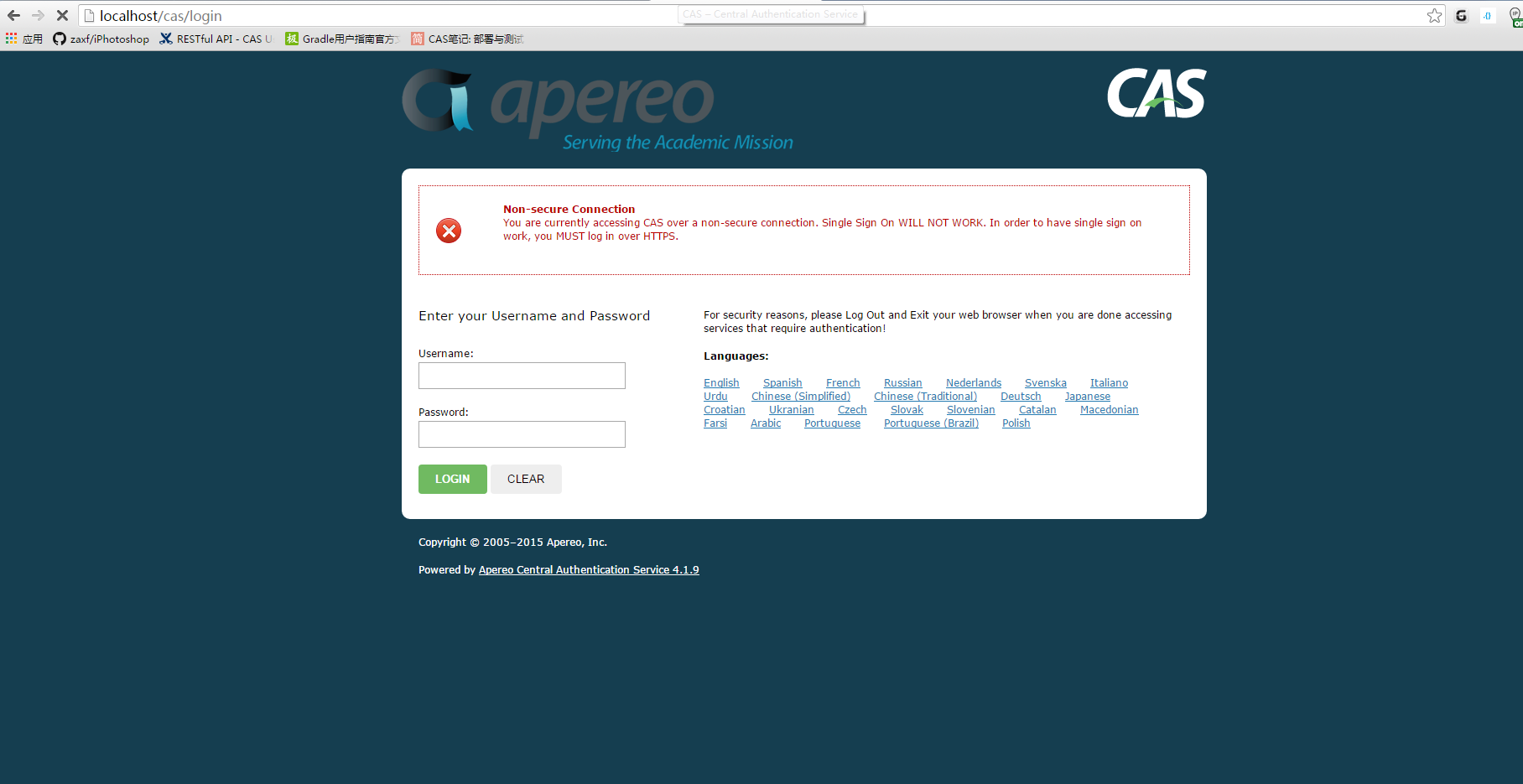
1. 从https://github.com/Jasig/cas/releases 下载cas-server对应的版本；这里使用的是 4.1.9
2. 先编译，因为 test 阶段会报错 所以直接执行 install –Dmaven.test.skip=true
3. 再编译



将target 下的war 包扔到tomcat 下



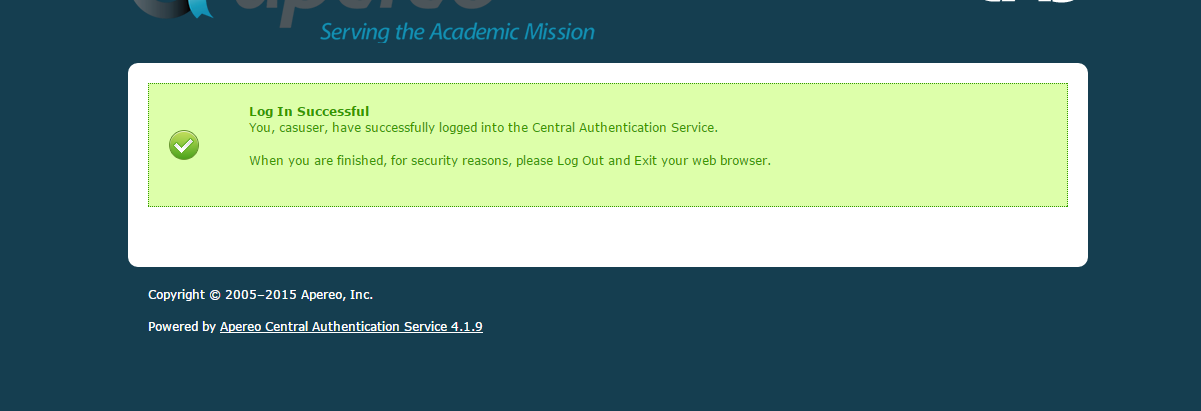
<http://localhost/cas> 因为我使用的是80 端口，所以直接项目名进入页面



默认的登录名 casuser

默认密码 Mellon

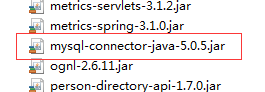
登录成功有以下界面

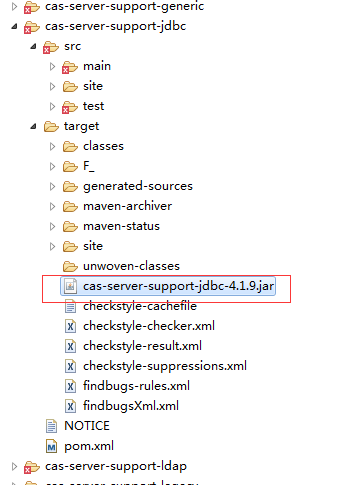


server配置数据库连接认证

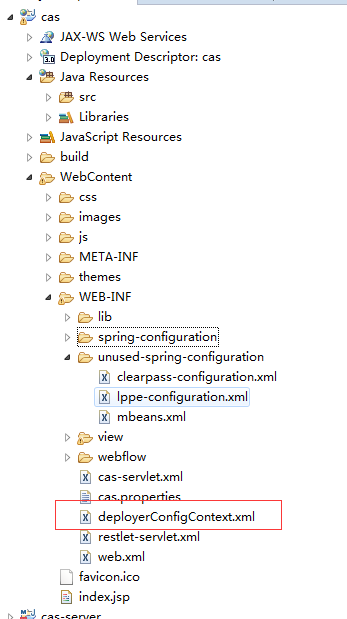
编译，将 jar 包导入cas/WEB-INF/lib

再导入一个 mysql 驱动包





登录账号实际上是在cas/WEB-INF/deployerConfigContext.xml配置写死的，打开配置文件



增加参数p:requireSecure="false"，是否需要安全验证，即HTTPS，false为不采用

找到如下配置

<!-- Required for proxy ticket mechanism. -->

<bean id=*"proxyAuthenticationHandler"*

class=*"org.jasig.cas.authentication.handler.support.HttpBasedServiceCredentialsAuthenticationHandler"*

p:httpClient-ref=*"supportsTrustStoreSslSocketFactoryHttpClient"* />

改成如下配置：

<!-- Required for proxy ticket mechanism. -->

<bean id=*"proxyAuthenticationHandler"*

class=*"org.jasig.cas.authentication.handler.support.HttpBasedServiceCredentialsAuthenticationHandler"*

p:httpClient-ref=*"supportsTrustStoreSslSocketFactoryHttpClient"* p:requireSecure=*"false"*/>



配置在这里，我们修改其配置，以用于自定义验证规则

注释之，增加如下代码(以后可以增加配置文件，这里暂时为了测试)

<!-- 2016-11-07 Ja0ck5 add -->

<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<property name=*"driverClassName"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"url"*

value=*"jdbc:mysql://localhsot:3306/cas?characterEncoding=utf-8&amp;autoReconnect=true"* />

<property name=*"username"* value=*"root"* />

<property name=*"password"* value=*"123456"* />

</bean>

<bean id=*"primaryAuthenticationHandler"* class=*"org.jasig.cas.adaptors.jdbc.QueryDatabaseAuthenticationHandler"*

p:dataSource-ref=*"dataSource"* p:passwordEncoder-ref=*"MD5PasswordEncoder"*

p:sql=*"select password from user\_info where user\_name=?"* />

<bean id=*"MD5PasswordEncoder"*

class=*"org.jasig.cas.authentication.handler.DefaultPasswordEncoder"*>

<constructor-arg index=*"0"*>

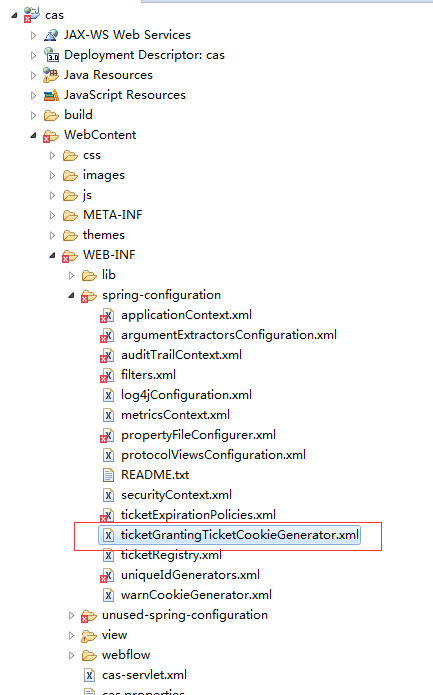
<value>MD5</value>

</constructor-arg>

</bean>

至此，数据库验证配置完成。

cas-server\WEB-INF\spring-configuration\ticketGrantingTicketCookieGenerator.xml



<bean id=*"ticketGrantingTicketCookieGenerator"* class=*"org.jasig.cas.web.support.CookieRetrievingCookieGenerator"*

c:casCookieValueManager-ref=*"cookieValueManager"*

p:cookieSecure=*"true"*

p:cookieMaxAge=*"-1"*

p:cookieName=*"TGC"*

p:cookiePath=*""*/>

改为:

<bean id=*"ticketGrantingTicketCookieGenerator"* class=*"org.jasig.cas.web.support.CookieRetrievingCookieGenerator"*

c:casCookieValueManager-ref=*"cookieValueManager"*

p:cookieSecure=*"false"*

p:cookieMaxAge=*"-1"*

p:cookieName=*"TGC"*

p:cookiePath=*""*/>

即不开启https验证.

cas-server\WEB-INF\spring-configuration\warnCookieGenerator.xml

<bean id=*"warnCookieGenerator"* class=*"org.jasig.cas.web.support.CookieRetrievingCookieGenerator"*

p:cookieHttpOnly=*"true"*

p:cookieSecure=*"true"*

p:cookieMaxAge=*"-1"*

p:cookieName=*"CASPRIVACY"*

p:cookiePath=*""*/>

改为：

<bean id=*"warnCookieGenerator"* class=*"org.jasig.cas.web.support.CookieRetrievingCookieGenerator"*

p:cookieHttpOnly=*"true"*

p:cookieSecure=*"false"*

p:cookieMaxAge=*"-1"*

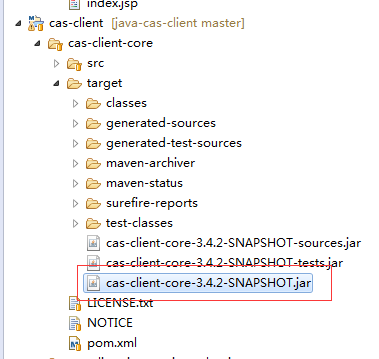
p:cookieName=*"CASPRIVACY"*

p:cookiePath=*""*/>

即不开启https验证

客户端（cas-client）的配置

从 https://github.com/Jasig/java-cas-client 下载java-cas-client，如上步骤解压编译，编译后不再操作，待用。



编译 cas-client,得到 cas-client-core 的jar 包

新建两个 tomcat 服务器,修改配置文件的三个端口，不能冲突，这里不赘述。

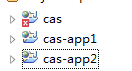
现在：

cas(服务端所在tomcat) 端口 80

新建两个项目

cas-app1(服务端所在tomcat) 端口 8088

cas-app2(服务端所在tomcat) 端口 8089



将上述 cas-client-core 的jar包放置 WEB-INFO/lib 目录下

然后配置 web.xml

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

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"* id=*"WebApp\_ID"* version=*"3.0"*>

<display-name>cas-app1</display-name>

<!-- 用于单点退出，该过滤器用于实现单点登出功能，可选配置 -->

<listener>

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

</listener>

<!-- 该过滤器用于实现单点登出功能，可选配置。 -->

<filter>

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

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

</filter>

<filter-mapping>

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

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

</filter-mapping>

<filter>

<filter-name>CASFilter</filter-name>

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

<init-param>

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

<!-- 服务器登录页面 -->

<!-- <param-value>https://localhost:8443/cas-server-webapp-4.0.0/login</param-value> -->

<param-value>https://localhost:8443/cas/login</param-value>

</init-param>

<init-param>

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

<!-- 当前应用 -->

<param-value>http://cas-app1.symbol.com:8088</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CASFilter</filter-name>

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

</filter-mapping>

<!-- 该过滤器负责对Ticket的校验工作，必须启用它 -->

<filter>

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

<filter-class>

org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter

</filter-class>

<init-param>

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

<!-- 服务器 -->

<!-- <param-value>https://localhost:8443/cas-server-webapp-4.0.0/</param-value> -->

<param-value>https://localhost:8443/cas/</param-value>

</init-param>

<init-param>

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

<!-- 当前应用 -->

<param-value>http://cas-app1.symbol.com:8088</param-value>

</init-param>

</filter>

<filter-mapping>

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

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

</filter-mapping>

<!-- 该过滤器负责实现HttpServletRequest请求的包裹， 比如允许开发者通过HttpServletRequest的getRemoteUser()方法获得SSO登录用户的登录名，可选配置。

--> <filter>

<filter-name>CAS HttpServletRequest WrapperFilter</filter-name>

<filter-class>

org.jasig.cas.client.util.HttpServletRequestWrapperFilter

</filter-class>

</filter>

<filter-mapping>

<filter-name>CAS HttpServletRequest WrapperFilter</filter-name>

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

</filter-mapping>

<!-- 该过滤器使得开发者可以通过org.jasig.cas.client.util.AssertionHolder来获取用户的登录名。 比如AssertionHolder.getAssertion().getPrincipal().getName()。

--> <filter>

<filter-name>CAS Assertion Thread LocalFilter</filter-name>

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

</filter>

<filter-mapping>

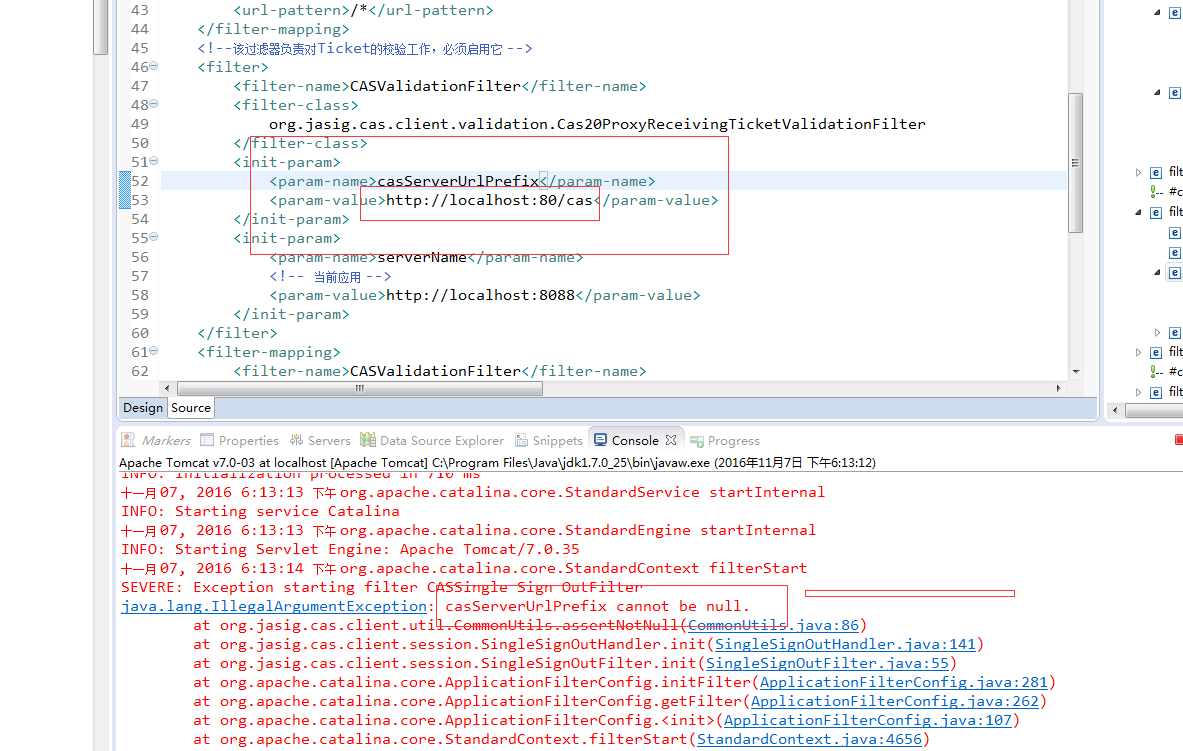
<filter-name>CAS Assertion Thread LocalFilter</filter-name>

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

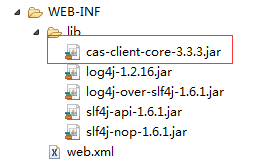
</filter-mapping>

</web-app>

cas-client 版本太高导致的



更换 cas-client 版本

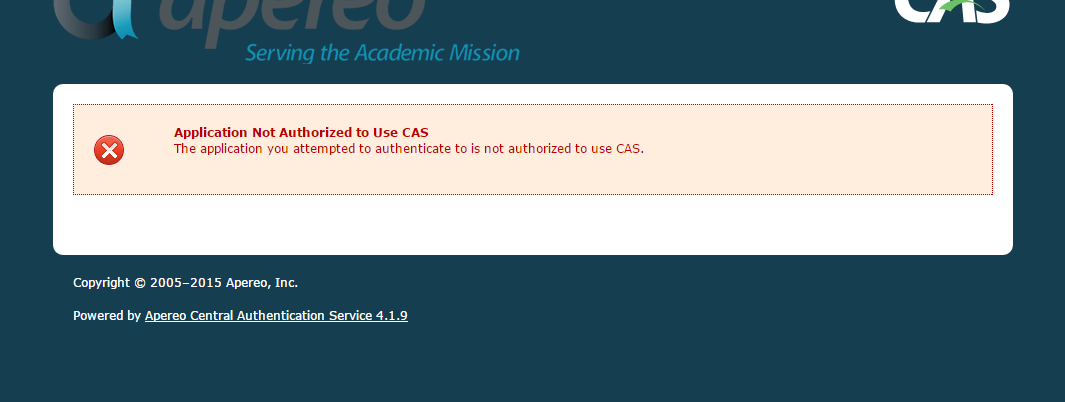


启动之后，不报错

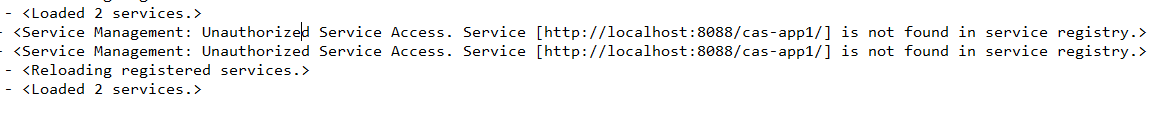
登录

<http://localhost:8088/cas-app1>

报错

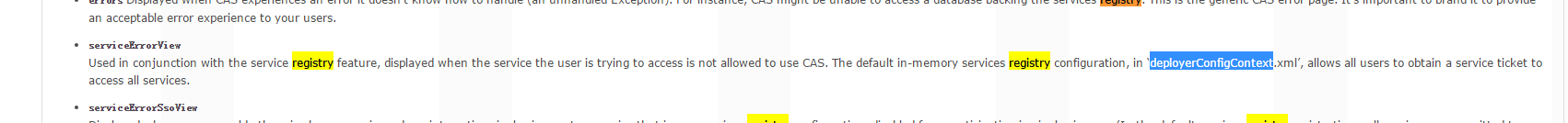


Console

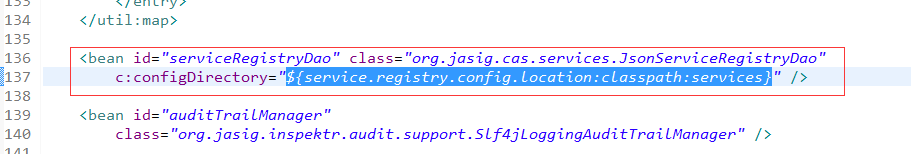


<Service Management: Unauthorized Service Access. Service [http://localhost:8088/cas-app1/] is not found in service registry.>

官方文档解释：

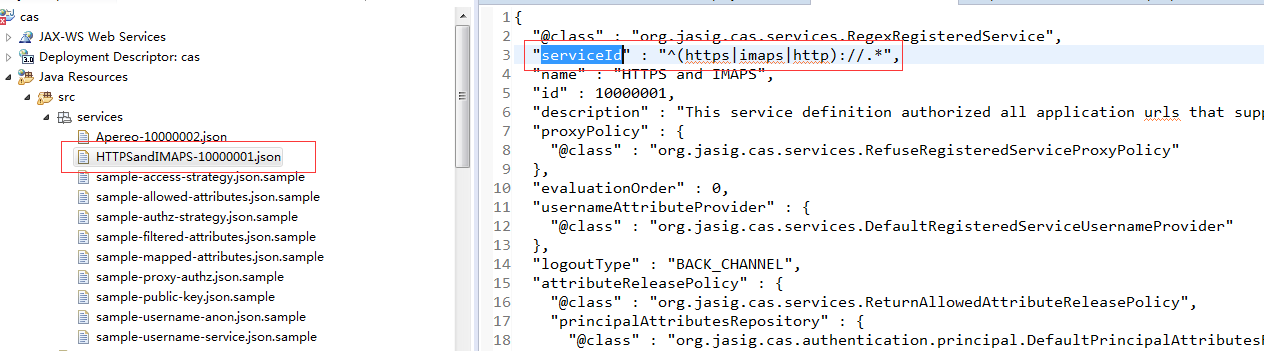


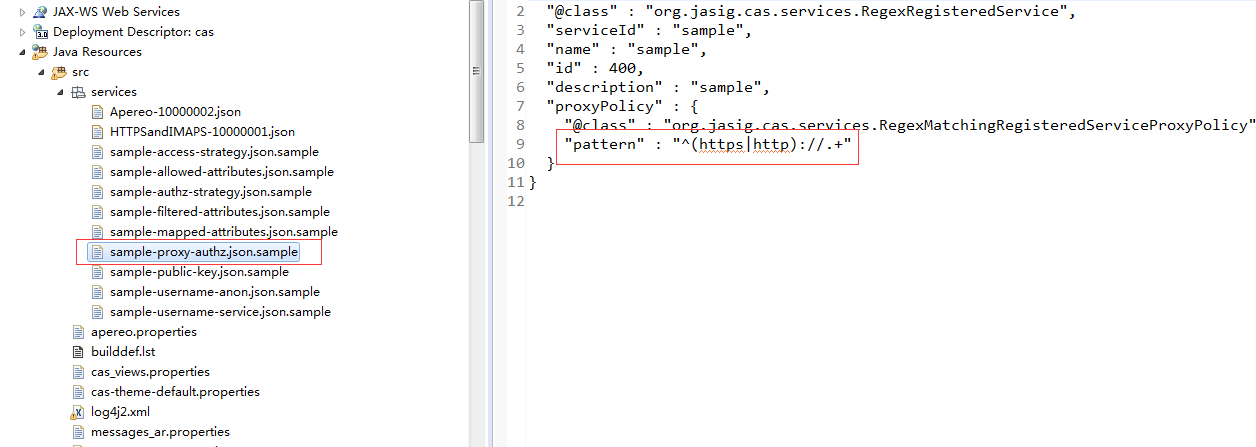
根据报错的信息以及服务端的配置文件 deployerConfigContext.xml：

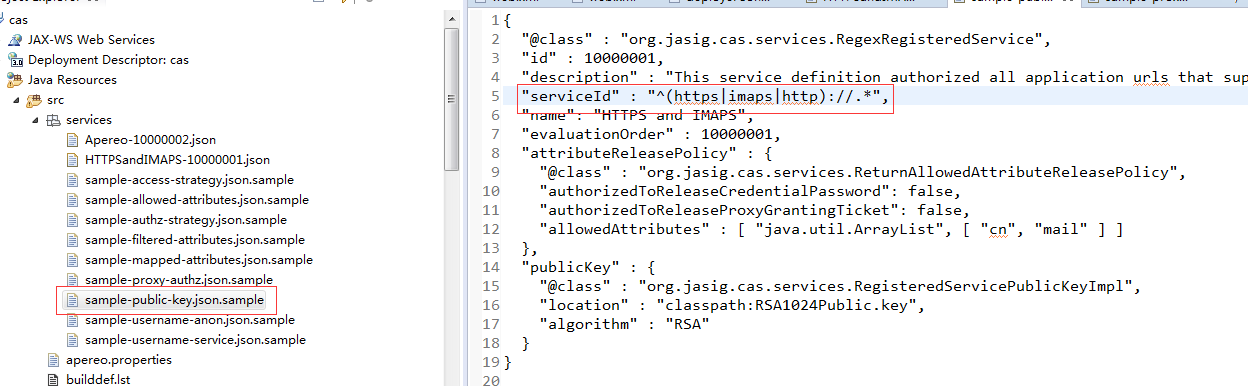


找到 registry 的配置文件

修改如下三处，让其允许 http 协议：

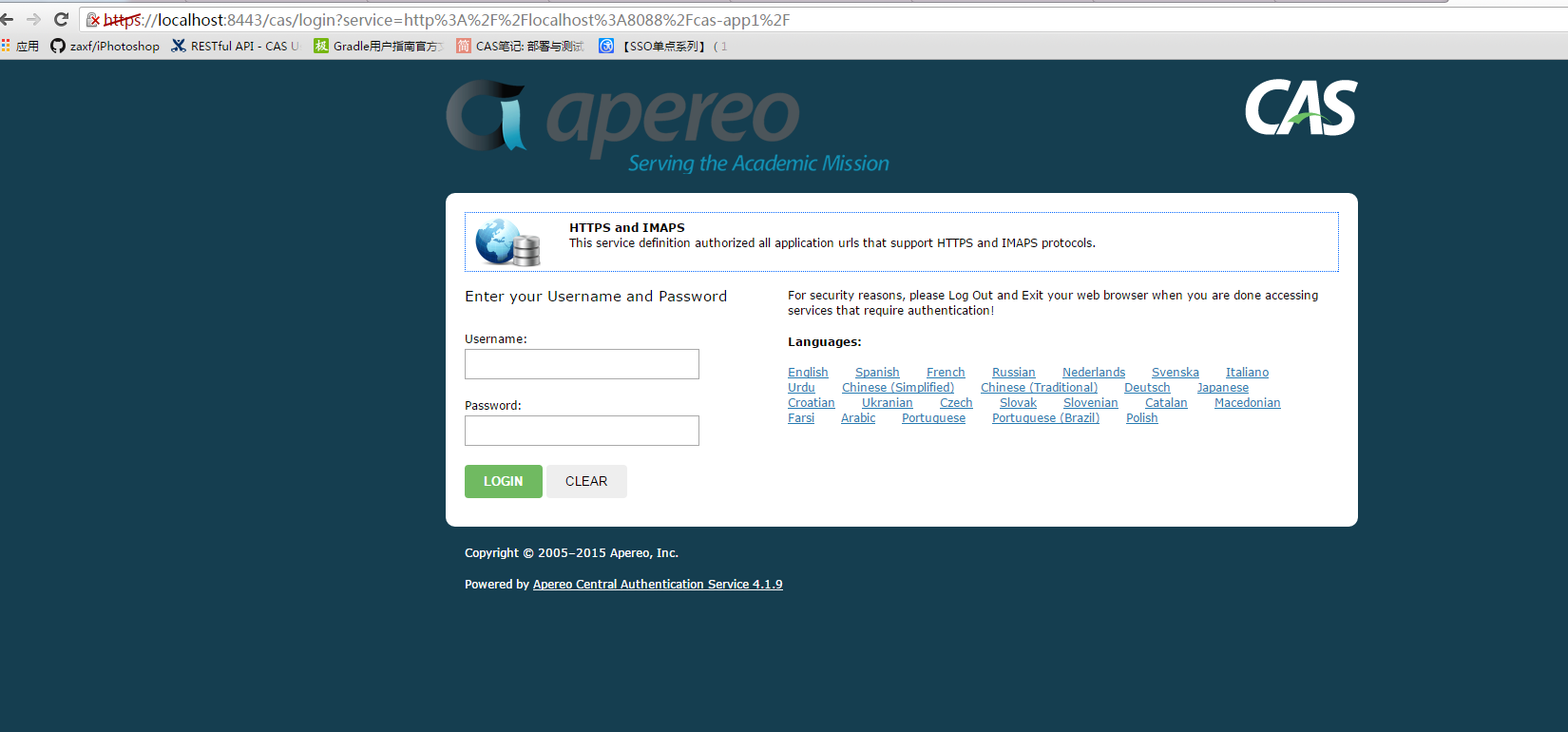
一、

二、

三、

再登录

<http://localhost:8088/cas-app1>



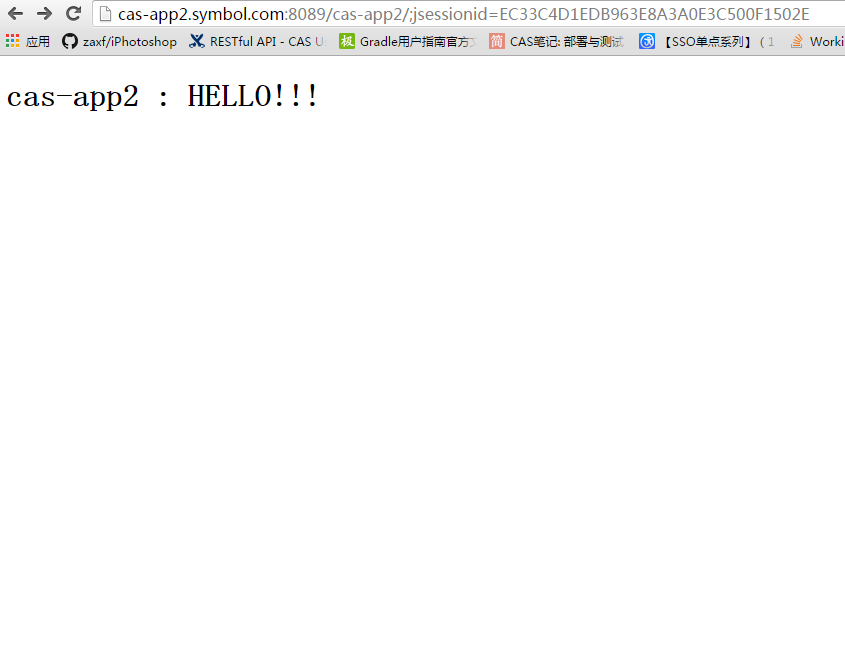
登录测试：报错



TODO 协议问题 使用完整 主机名 不能使用 ip 或者 localhost

登录成功CAS基本搭建成功：





整合 RESTFUL

参考：

版本：

4.0.x

http://stackoverflow.com/questions/22625368/working-java-rest-client-example-to-access-cas-rest-api

cas-server(即cas.war) web.xml下

in your pom.xml add following dependency

**Direct dependency to springframework is not necesarry because exclusions prevent from duplicated packages**

<dependency>

<groupId>org.jasig.cas</groupId>

<artifactId>cas-server-integration-restlet</artifactId>

<version>4.0.0</version>

<scope>runtime</scope>

<exclusions>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

</exclusion>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</exclusion>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

</exclusion>

</exclusions>

</dependency>

**In your web.xml you need to add servlet mapping for restlet (mind package has changed from com.noelios.restlet... to org.restlet...**

<servlet>

<servlet-name>restlet</servlet-name>

<servlet-class>org.restlet.ext.spring.RestletFrameworkServlet</servlet-class>

<load-on-startup>1</load-on-startup>

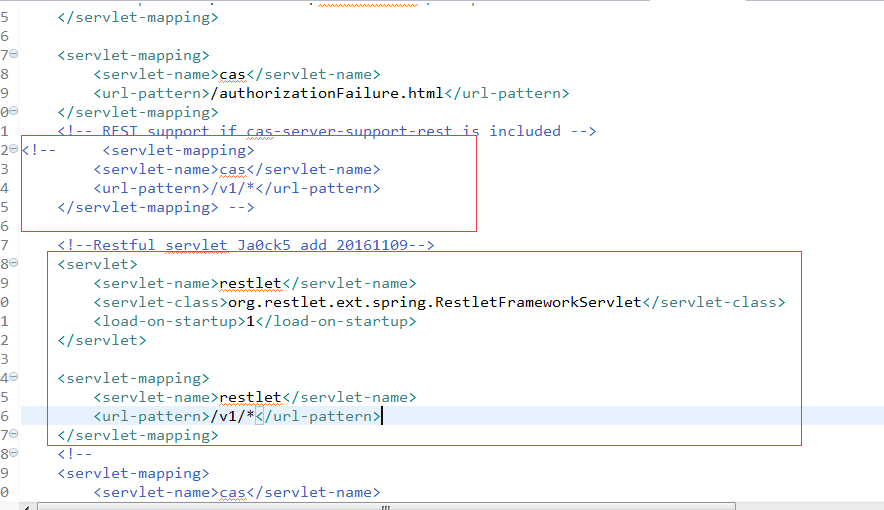
</servlet>

<servlet-mapping>

<servlet-name>restlet</servlet-name>

<url-pattern>/v1/\*</url-pattern>

</servlet-mapping>



依赖 jar 包

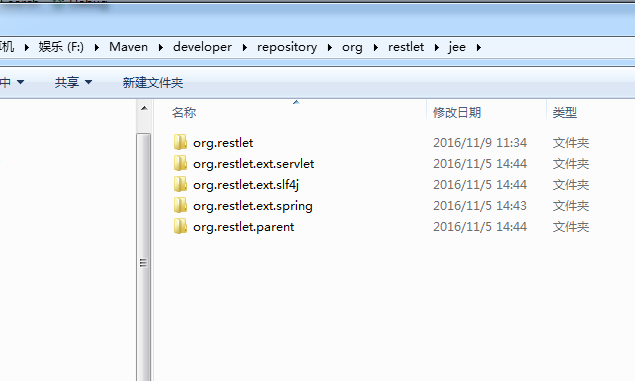
cas-server-integration-restlet-4.1.9.jar

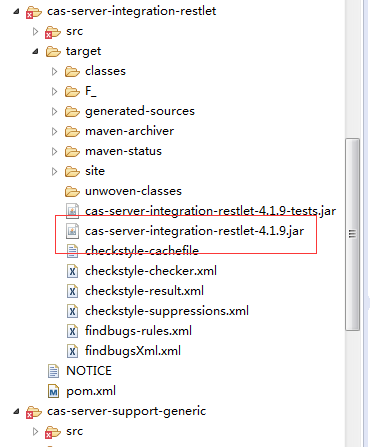
org.restlet-2.1.0.jar

org.restlet.ext.servlet-2.1.0.jar

org.restlet.ext.slf4j-2.1.0.jar

org.restlet.ext.spring-2.1.0.jar





自己使用的版本 4.1.9

官方 REST-Protocol

参考：

<https://apereo.github.io/cas/4.1.x/protocol/REST-Protocol.html>

REST support is currently provided internally by the Spring framework.

REST 支持 目前已经由Spring framework 内部提供了

4.1.x 的版本 在 cas-server 端的 web.xml 文件中已经说明了如下配置

<!-- REST support if cas-server-support-rest is included -->

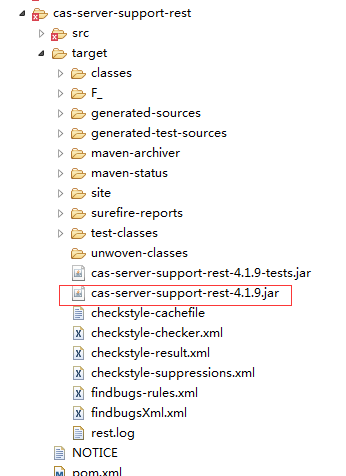
<servlet-mapping>

<servlet-name>cas</servlet-name>

<url-pattern>/v1/\*</url-pattern>

</servlet-mapping>

所以还需要加上个这个 cas-server-support-rest jar 包（可以直接在源码中编译获得）

如下：

官方说明的文档：配置如下参数

Please note that if there are local customizations in overlay’s web.xml, the following contextConfigLocation <context-param> must also be added in order to enable the new REST module: classpath\*:/META-INF/spring/\*.xml. So the entire context-param block would look like this:

**<context-param>**

**<param-name>**contextConfigLocation**</param-name>**

**<param-value>**

/WEB-INF/spring-configuration/\*.xml

/WEB-INF/deployerConfigContext.xml

classpath\*:/META-INF/spring/\*.xml

**</param-value>**

**</context-param>**

当然我发现其实原本就是这样

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

/WEB-INF/spring-configuration/\*.xml

/WEB-INF/deployerConfigContext.xml

<!-- this enables extensions and addons to contribute to overall CAS' application context

by loading spring context files from classpath i.e. found in classpath jars, etc. -->

classpath\*:/META-INF/spring/\*.xml

</param-value>

</context-param>

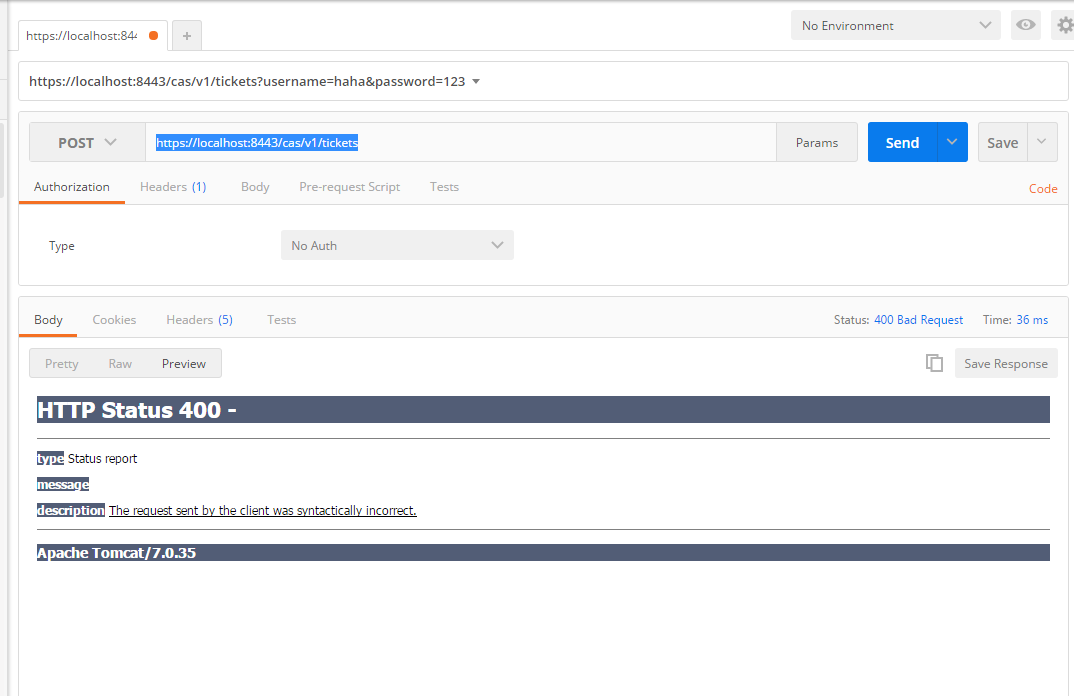
使用 POSTMAN 测试工具测试

## Request a Ticket Granting Ticket

### Sample Request

1. 直接访问 <https://localhost:8443/cas/v1/tickets>

会报 400 错误，

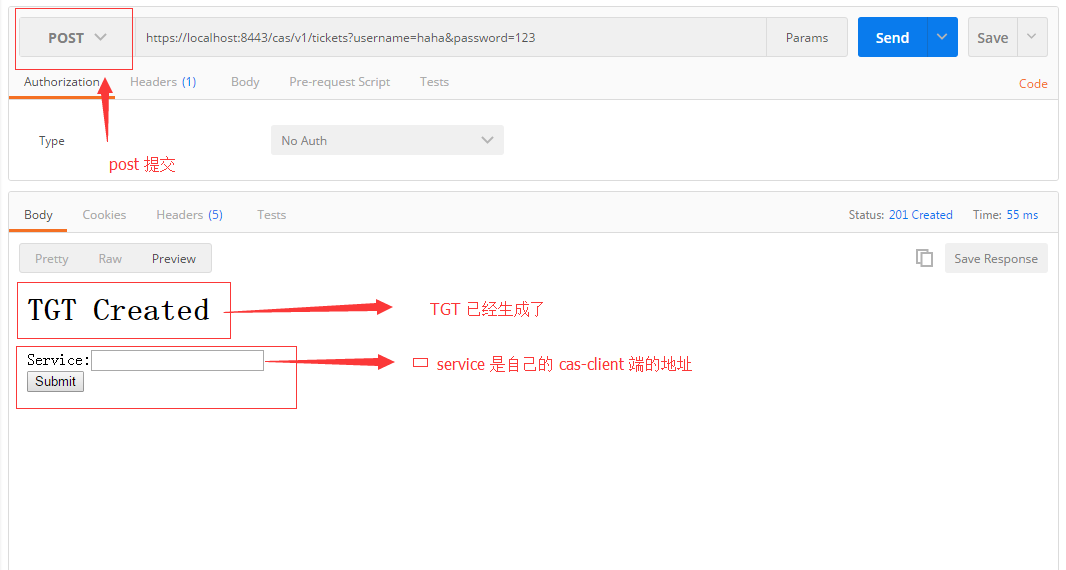
需要携带参数

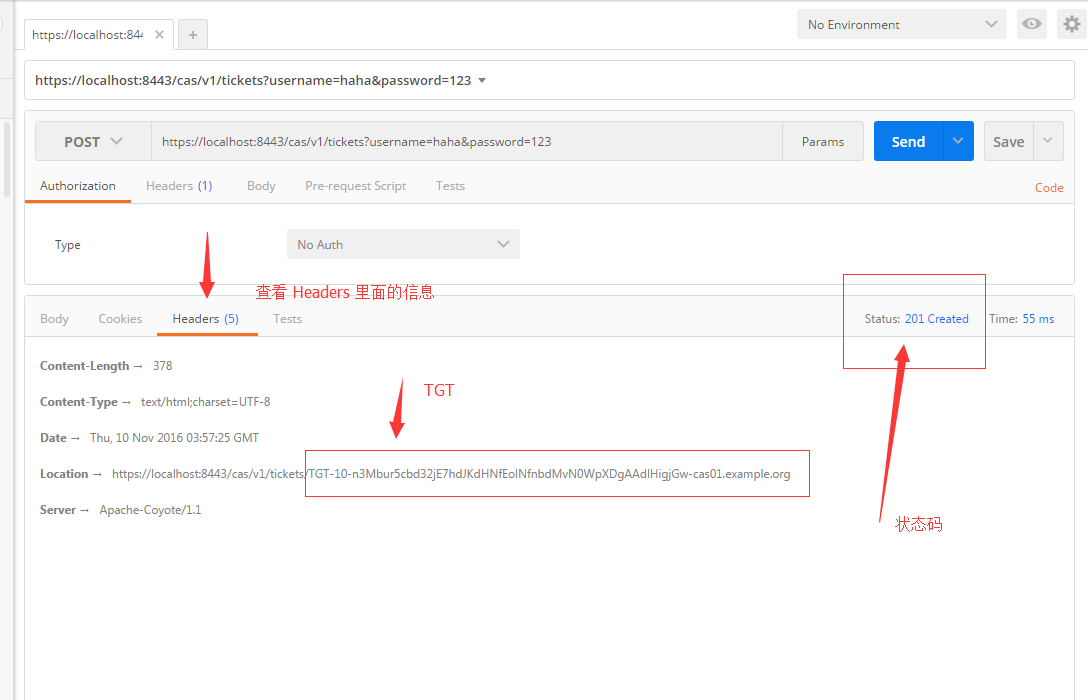
additionalParam1=paramvalue 可加 可不加(其实就是告诉你还可以加参数而已)

username=haha&password=123&additionalParam1=paramvalue

访问： <https://localhost:8443/cas/v1/tickets?username=haha&password=123>

### TGT Created





#### 后台控制台输出



#### Unsuccessful Response

If incorrect credentials are sent, CAS will respond with a 400 Bad Request error (will also respond for missing parameters, etc.). If you send a media type it does not understand, it will send the 415 Unsupported Media Type.

如果不正确的凭据被发送 CAS 会返回 400 Bad Request 的状态，也会说遗漏参数 等等，如果你发送一个 它不能识别的多媒体类型，它会发送 415 Unsupported Media Type

## Request a Service Ticket

### Sample Request

{TGT id} 就是

**Location →**https://localhost:8443/cas/v1/tickets/TGT-10-n3Mbur5cbd32jE7hdJKdHNfEolNfnbdMvN0WpXDgAAdlHigjGw-cas01.example.org

里面的

TGT-10-n3Mbur5cbd32jE7hdJKdHNfEolNfnbdMvN0WpXDgAAdlHigjGw-cas01.example.org

POST /cas/v1/tickets/{TGT id} HTTP/1.0

service={form encoded parameter **for** the service url}

测试：url

<https://localhost:8443/cas/v1/tickets/TGT-4-kIahGcfkdMPjTmXIqbmCgvZATm7LfG2uy3igjvnoQ0Scv3Sljr-cas01.example.org?service=http%3A%2F%2Fcas-app1.symbol.com%3A8088%2Fcas-app1%2F>

Successful Response

返回信息

200 OK

ST-1-o9rR9fOJ7yrbuAUsTyKd-cas01.example.org

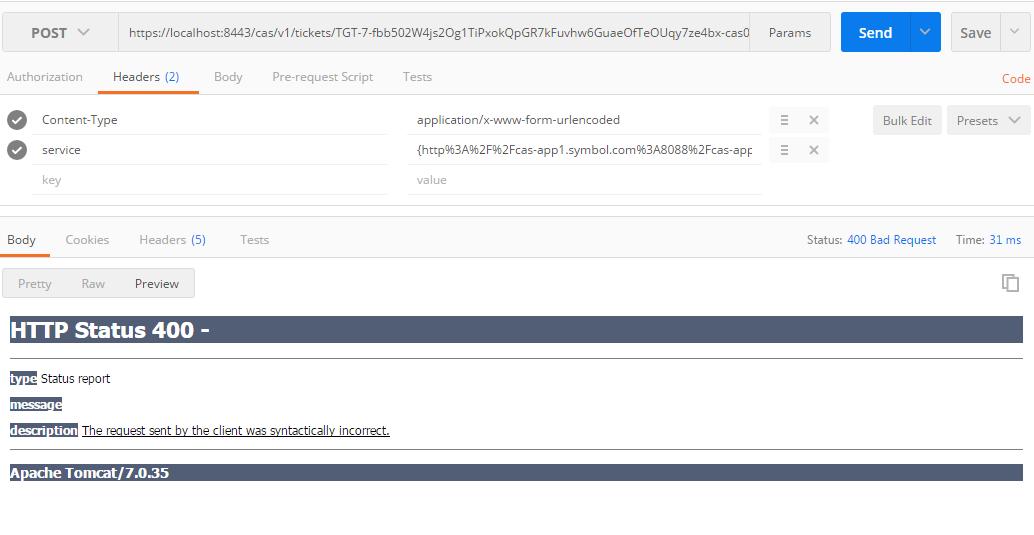
#### 返回 ST截图

#### 

#### 后台控制台输出

#### Unsuccessful Response

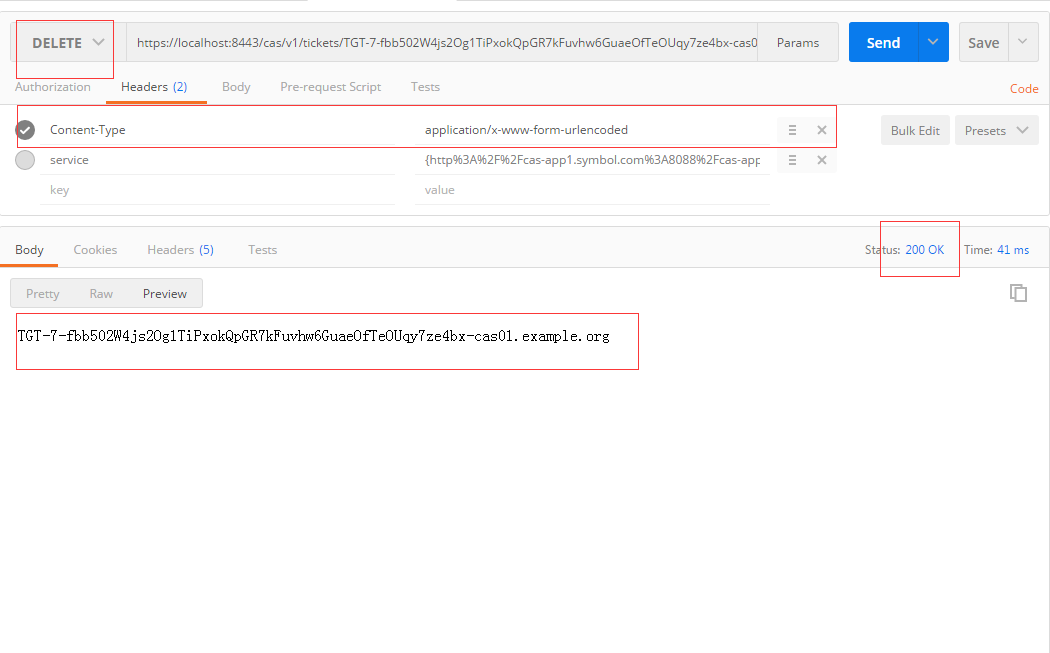
CAS will send a 400 Bad Request. If an incorrect media type is sent, it will send the 415 Unsupported Media Type.



## Logout

DELETE /cas/v1/tickets/TGT\*\*\*OUqy7ze4bxcas01.example.org HTTP/1.0

## DELETE 截图



## 控制台输出

