**SSO之CAS单点登录详细搭建教程**

本教程是我个人编写，花费几个小时的时间，给需要学习的人员学习使用，希望能帮助到你们。

【环境说明】：

本文演示过程在同一个机器上的（也可以在三台实体机器或者三个的虚拟机上），环境如下：

* windows7 64位
* jdk1.7.0\_51
* apache-tomcat-7.0.57-windows-x64
* 会使用到的：cas-server-webapp-4.0.0.war、cas-client-core-3.2.1.jar、commons-logging.jar
* 确保本地jdk环境已经搭建好

【软件说明】：

文档中涉及到的一些jar，还有cas服务的war，已经打包放在一起了，可以直接使用。

根据演示需求，用修改hosts 文件的方法添加域名最简单方便（这个非常重要，因为CAS单点登录系统是基于JAVA安全证书的 https 访问， 要使用CAS单点登录必须要配置域名, cas是不能通过ip访问的.），编辑文件 C:\Windows\System32\drivers\etc\hosts 在文件末端添加下面三条信息：

127.0.0.1 server.zhoubang85.com

127.0.0.1 client1.zhoubang85.com

127.0.0.1 client2.zhoubang85.com

* server.zhoubang85.com  =>> 对应部署cas server的tomcat，这个虚拟域名还用于服务端证书生成
* client1.zhoubang85.com  =>>  对应部署client1客户端应用的tomcat
* client2.zhoubang85.com   =>> 对应部署client2客户端应用的tomcat

【注意】：上面3个ip都是127.0.0.1，这是为什么呢？因为我的环境都是在同一台机器，所以ip都是一致的，我们再把不同的服务端和客户端应用，使用不同域名加以区分。一个域名对应一个应用，模拟多端！

【由于CAS是基于HTTPS协议，所以需要配置服务端的tomcat，使之支持SSL安全协议访问】

【说明】：当然也可以设置cas支持http访问，这个之后再补充讨论，下面先配置https协议.

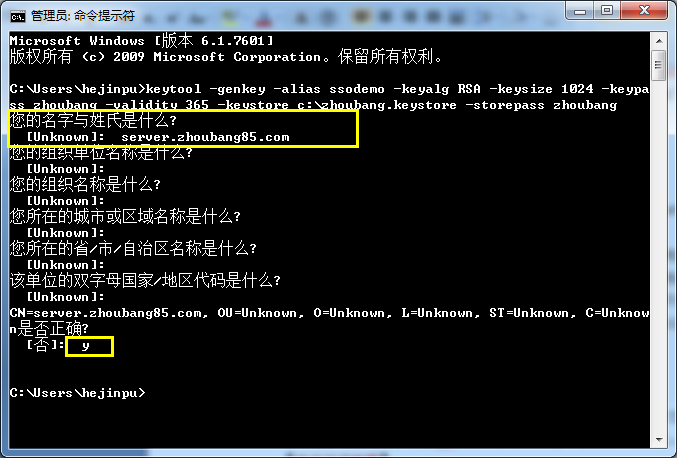
【**安全证书配置**】

1. 打开cmd命令窗口
2. 生成证书，在cmd窗口输入以下命令：

keytool -genkey -alias ssodemo -keyalg RSA -keysize 1024 -keypass zhoubang -validity 365 -keystore c:\zhoubang.keystore -storepass zhoubang

**[说明]：-alias后面的别名可以自定义，-keypass指定证书密钥库的密码, -storepass和前面keypass密码相同,否则下面tomcat 配置https 会访问失败 -keystore指定证书的位置,这里指定放在c盘根目录,密钥库名称可以自定义..这里是zhoubang.keystore**

1. 命令输入完成，回车之后，会提示你输入一些资料，见下图：



【**注意**】：第一个让你输入的“您的名字与姓氏是什么”，请必须输入在C:\Windows\System32\drivers\etc\hosts文件中加入的服务端的域名.我这里也就是server.zhoubang85.com，为何这么做？首先cas只能通过域名来访问，不能通过ip访问，同时上方是生成证书，所以要求比较严格，所以如果不这么做的话，及时最终按照教程配置完成，cas也可以正常访问,访问一个客户端应用虽然能进入cas验证首页，但是，当输入信息正确后，cas在回调转入你想访问的客户端应用的时候，会出现**No subject alternative names present**错误异常信息**，这个错误也就是在上面输入的第一个问题答案不是域名导致、或者与hosts文件配置的不一致导致**。

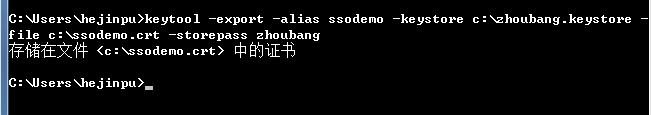
1. 导出证书：

在cmd窗口继续输入以下命令，导出证书

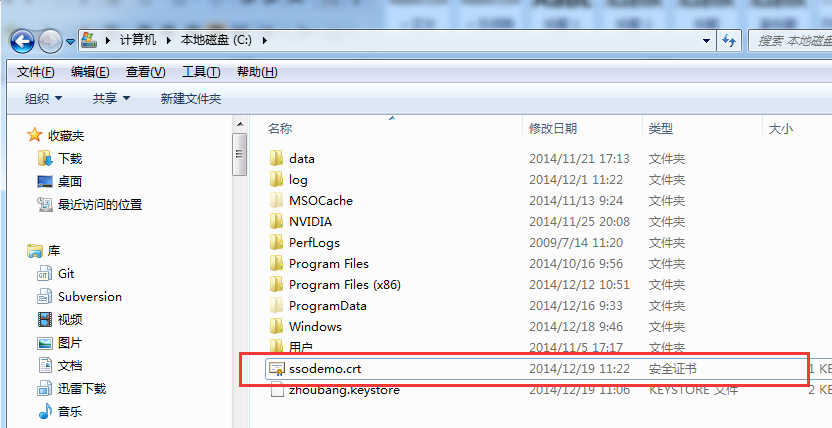
keytool -export -alias ssodemo -keystore c:\zhoubang.keystore -file c:\ssodemo.crt -storepass zhoubang

【说明】：-alias后面的名称要与生成证书的命令里面的alias的名称一致. –keystore后面指定证书存放的位置，这里我放在C盘根目录，同时证书名称要与【生成证书】对应的命令里的keystore名称一致.这里是zhoubang.keystore，-file后面才crt路径，我也指定在c盘根目录. –storepass的证书密码要与上面输入的密码一致.

如下图所示：



我们再看看c盘下面是否生成crt文件.



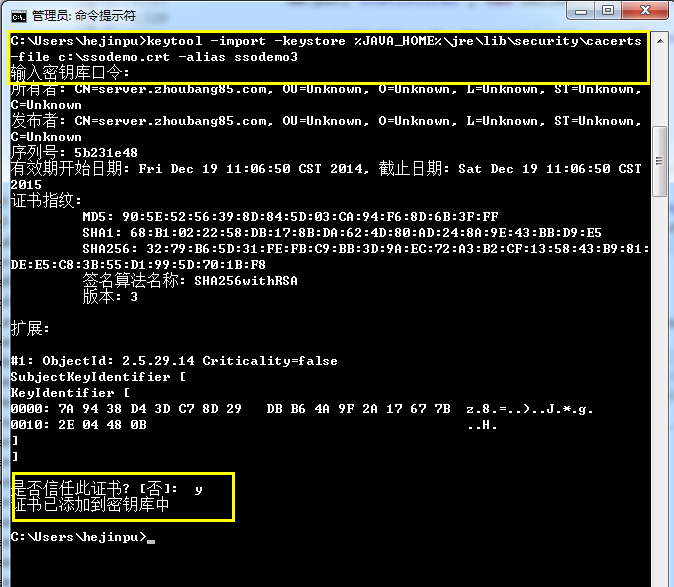
1. 客户端导入证书

在cmd窗口输入命令

keytool -import -keystore %JAVA\_HOME%\jre\lib\security\cacerts -file c:\ ssodemo.crt -alias ssodemo

【说明】：-file指定证书的位置，也就是上一步导出证书的位置，即c:\ ssodemo.crt 命令中指定了JAVA\_HOME，意思是将证书导入到客户端证书库，也就是jdk证书库中.因为客户端应用运行在本地，需要jdk的支持。

回车之后，会让你输入密钥库口令，注意，这里的密码必须要输入changeit，不能输入上面指定的密码zhoubang，切记，否则导入客户端证书会有问题，如果是多台机器演示，需要在每一台客户端导入该证书，步骤都是一样的。当看到提示“是否信任此证书”，输入y回车即可，见下图：(说明，命令中的-alias后面的别名可以自定义,如果出现【证书未导入，别名<\*\*\*>已经存在】的错误，该意思是说客户端的密钥库中已经存在该别名证书了，重新指定其他别名即可.)



**至此，CAS所需的证书环境，已经配置好。**

下面，开始我们的CAS服务、Tomcat、以及多客户端的配置及测试访问。

* **部署CAS-Server相关的Tomcat**

1. 配置HTTPS

解压**apache-tomcat-7.0.57-windows-x64.zip，我本地路径为E:\apache-tomcat-7.0.57，编辑E:\apache-tomcat-7.0.57\conf\server.xml，找到下面片段：**

**<!--**

**<Connector executor="tomcatThreadPool"**

**port="8080" protocol="HTTP/1.1"**

**connectionTimeout="20000"**

**redirectPort="8443" />**

**-->**

**去掉注释，修改成：**

**<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"**

**maxThreads="150" SSLEnabled="true" scheme="https" secure="true"**

**keystoreFile="C:/zhoubang.keystore" keystorePass="zhoubang"**

**clientAuth="false" sslProtocol="TLS" />**

**其中，keystoreFile就是**创建证书的路径，**keystorePass就是**创建证书的密码

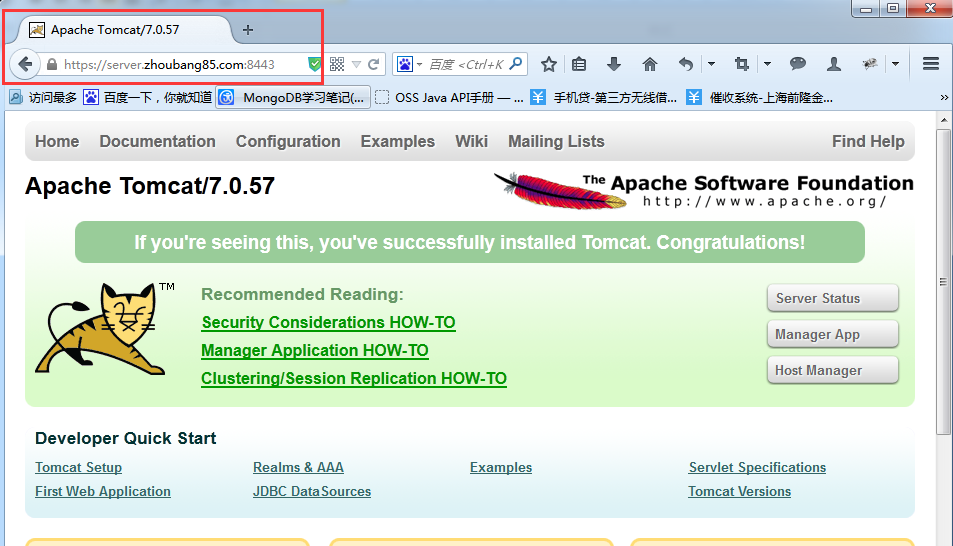
1. 验证HTTPS配置

其他按照默认配置不作修改，双击E:\apache-tomcat-7.0.57\bin \startup.bat 启动tomcat 验证https是否配置成功，我本地使用火狐浏览器访问，在地址栏输入

<https://server.zhoubang85.com:8443/> 出现下面画面，其实这就表明cas服务端tomcat的https配置是没有问题了.

****

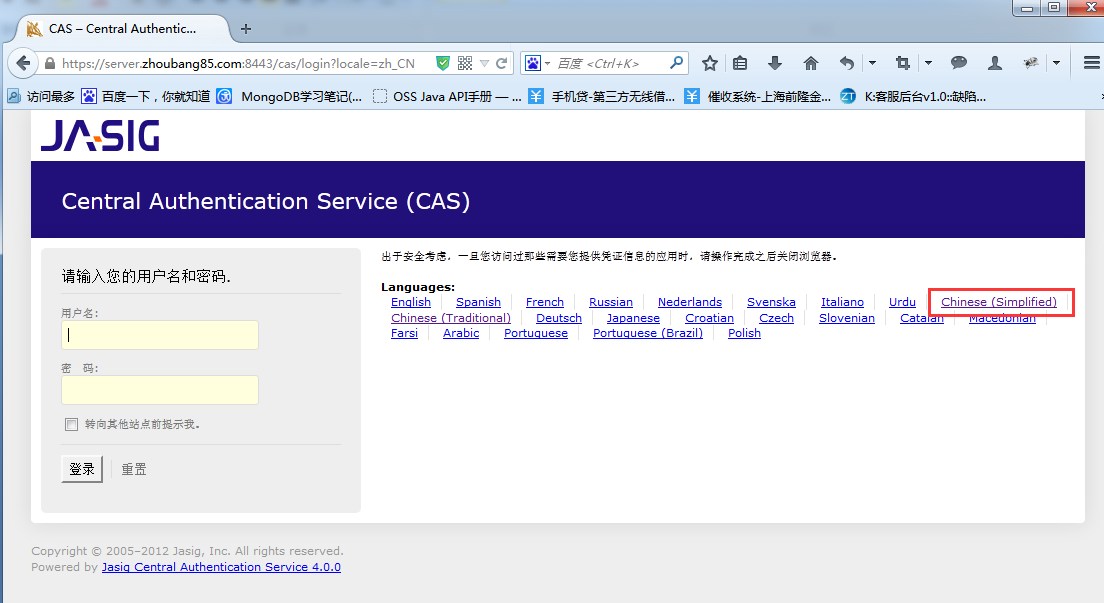
点击【我已充分了解可能的风险】,点击出现的【添加例外】—【确认安全例外】，，将会出现下面tomcat默认首页.tomcat已经支持https协议访问了，一切ok!



**3、部署CAS-Server**

CAS-Server 下载地址：<http://www.jasig.org/cas/download>

本文以cas-server-webapp-4.0.0.rar为例，解压提取cas-server-webapp-4.0.0.war文件，把改文件copy到E:\apache-tomcat-7.0.57\webapps 目下，并重命名为：cas.war.  
启动tomcat，在浏览器地址栏输入：https://server.zhoubang85.com:8443/cas ，回车,出现CAS服务端的登录验证首页：

****

首页默认是英文的，点击红色框的链接，切换成中文显示。当然，并不是所有的地方都会变成中文，这个不作讲解。

【说明】：此时，CAS只是单独运行，至于登录的用户名和密码是什么，请查看：

E:\apache-tomcat-7.0.57\webapps\cas\WEB-INF\deployerConfigContext.xml文件中有这样一段配置：

<bean id="primaryAuthenticationHandler"

class="org.jasig.cas.authentication.AcceptUsersAuthenticationHandler">

<property name="users">

<map>

<entry key="casuser" value="Mellon"/>

</map>

</property>

</bean>

这个配置是默认静态配置用户名和密码，那就在浏览器的CAS服务的登陆框中输入用户名casuser和密码Mellon，登录看看效果，就会出现验证成功的页面，如下图。你也可以在文件中自己配置自己的用户名和密码。



**看到上述页面表示CAS-Server已经部署成功。**

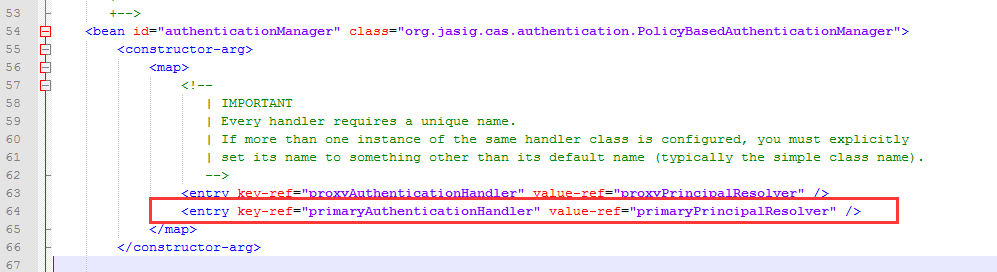
用户名和密码肯定需要和数据库进行交互验证的，，那么，如何配置呢？

【说明】：我本地使用的是mysql数据库。

1、，需要将几个jar文件，放到CAS服务的lib目录下，我本地使用的jar版本分别是c3p0-0.9.1.2.jar、cas-server-support-jdbc-4.0.0.jar、mysql-connector-java-5.1.13-bin.jar，这3个缺一不可。将这3个jar放到E:\apache-tomcat-7.0.57\webapps\cas\WEB-INF\lib目录下。

2、修改配置，支持mysql数据库交互验证

编辑E:\apache-tomcat-7.0.57\webapps\cas\WEB-INF\ deployerConfigContext.xml文件，你会看到有这样一段配置：



注释掉第二个entry配置，最终配置如下：

<constructor-arg>

<map>

<entry key-ref="proxyAuthenticationHandler" value-ref="proxyPrincipalResolver" />

<!-- 注释这个-->

<!--<entry key-ref="primaryAuthenticationHandler" value-ref="primaryPrincipalResolver" />-->

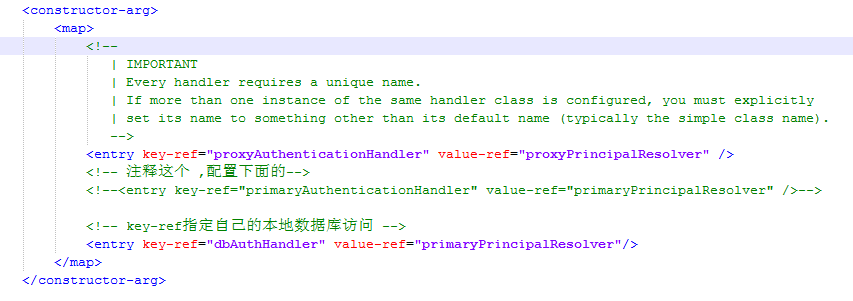
<!-- key-ref指定自己的本地数据库访问 -->

<entry key-ref="dbAuthHandler" value-ref="primaryPrincipalResolver"/>

</map>

</constructor-arg>

上述配置截图：



然后再在这个xml中新加入2个bean配置，如下：

<!-- 指定c3p0数据源 -->

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

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

<property ame="jdbcUrl" value="jdbc:mysql://127.0.0.1:3306/myProject?useUnicode=true&amp;characterEncoding=UTF-8" />

<property name="user" value="root" />

<property name="password" value="root

</bean>

<!-- 访问本地数据库 -->

<bean id="dbAuthHandler"

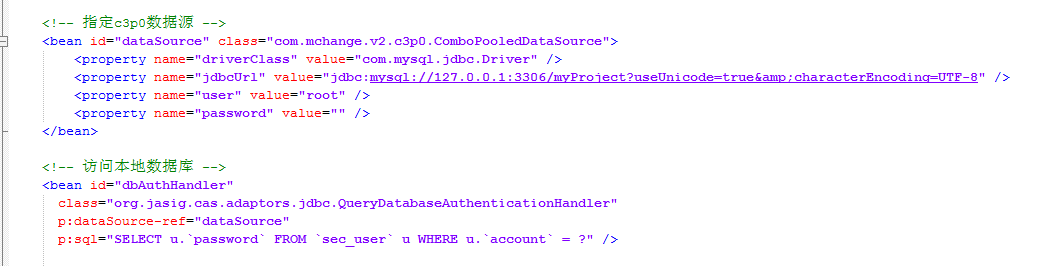
class="org.jasig.cas.adaptors.jdbc.QueryDatabaseAuthenticationHandler"

p:dataSource-ref="dataSource"

p:sql="SELECT u.`password` FROM `sec\_user` u WHERE u.`account` = ?" />

要说明的是，红色标注的内容，我想大家都能看的明白，就是指定数据库驱动和连接信息。其中，sql语句的意思就是，根据用户名获取密码，CAS会根据你页面输入的用户名获取该用户密码，和你输入的密码进行校验，来判断输入是否正确。Sql中的表换成你自己本地的表即可。只要根据用户名查询密码即可。

上述配置截图：



**至此，CAS与数据库交互验证的配置已经配置完成**，你可以重新访问cas，输入数据库中存在的用户名和密码，来看看效果如何~如果登录成功，说明配置无误。否则，请耐心检查配置是否有问题，jar包是否缺少。

现在，CAS已经支持数据库交互验证了，服务端tomcat也支持HTTPS协议访问，现在，我们来搭建客户端，实现多个客户端的单点登录。这里，我本地只使用2个tomcat客户端来测试，其实已经满足单点登录的要求了，至少2个应用。

**4、部署CAS客户端相关的Tomcat**

首先，客户端应用是要和CAS服务端进行交互的，所以这里需要jar文件，放在客户端应用的lib目录下。分别是：cas-client-core-3.2.1.jar、commons-logging.jar

这里呢，我就直接使用tomcat默认自带的 webapps\examples 作为演示的简单web项目。我就不去另写一个web测试的demo了。

既然需要2个客户端应用，则需要2个tomcat做为客户端服务器，所以，我本地解压了2份tomcat，作为客户端服务器，并重新命名，本地路径分别为：E:\apache-tomcat-7.0.57-client1和E:\apache-tomcat-7.0.57-client2

首先，将上面2个jar分别放到E:\apache-tomcat-7.0.57-client1\webapps\examples\WEB-INF\lib目录

和E:\apache-tomcat-7.0.57-client2\webapps\examples\WEB-INF\lib目录

1. **配置apache-tomcat-7.0.57-client1客户端1：**

修改tomcat的启动端口：

编辑E:\apache-tomcat-7.0.57-client1\conf\server.xml文件，找到如下2处内容：

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

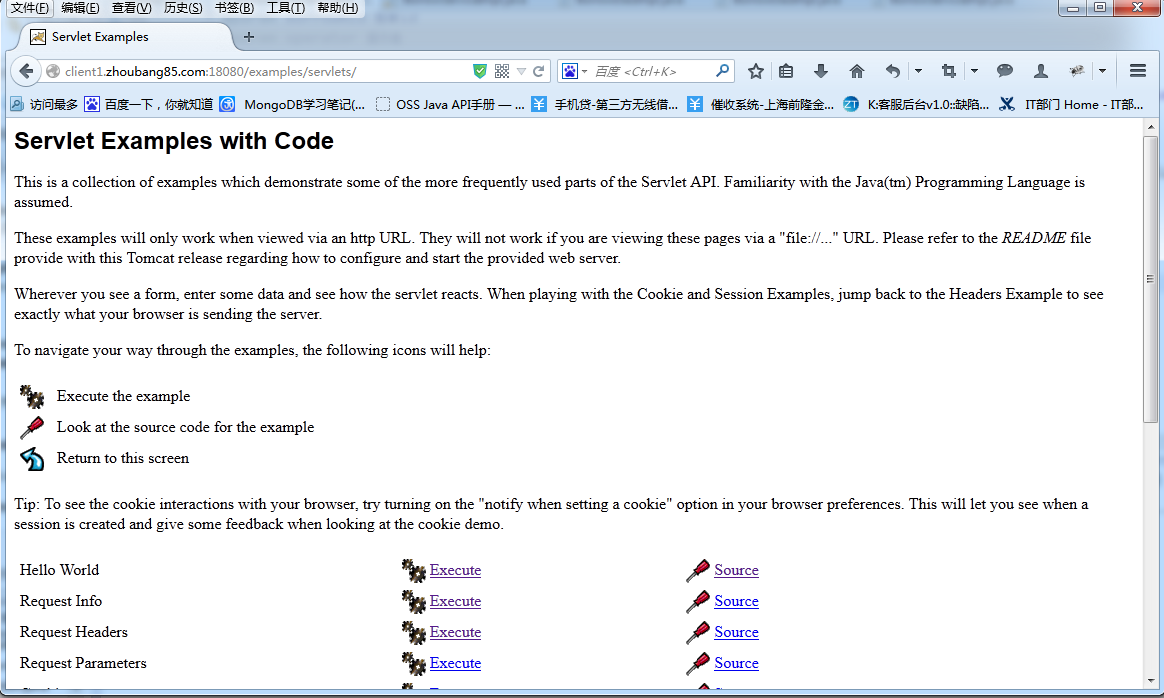
redirectPort="8443" />

这里请将port="8080"修改成其他4位端口，不能和CAS服务端的tomcat相同，否则，在同一台机器上测试会出现端口占用的错误，我修改成18080，如果是在不同的机器上，则不需要配置这些。

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

这里请将port="8009"修改成其他4位端口，不能和CAS服务端相同.我修改成18009

好，启动这个tomcat，即运行E:\apache-tomcat-7.0.57-client1\bin\startup.bat，如果启动窗口中没有出现错误，说明端口配置无误。请记住你配置的端口号。浏览器输入 <http://client1.zhoubang85.com:18080/examples/servlets/>，请注意红色部分域名,就是教程一开始需要配置的C:\Windows\System32\drivers\etc\hosts的域名，用于不同的客户端域名访问，回车：



看到上述界面表示apache-tomcat-7.0.57-client1的基本安装配置已经成功。

接下来需要配置最重要的内容，让客户端应用和CAS服务连接：

编辑E:\apache-tomcat-7.0.57-client1\webapps\examples\WEB-INF\web.xml，在最下面加入如下配置：

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

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

<init-param>

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

<param-value>https://server.zhoubang85.com:8443/cas/login</param-value>

</init-param>

<init-param>

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

<param-value>http://client1.zhoubang85.com:18080</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CAS Filter</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.Cas10TicketValidationFilter</filter-class>

<init-param>

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

<param-value>https://server.zhoubang85.com:8443/cas</param-value>

</init-param>

<init-param>

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

<param-value>http://client1.zhoubang85.com:18080</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 Wrapper Filter</filter-name>

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

</filter>

<filter-mapping>

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

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

</filter>

<filter-mapping>

<filter-name>CAS Assertion Thread Local Filter</filter-name>

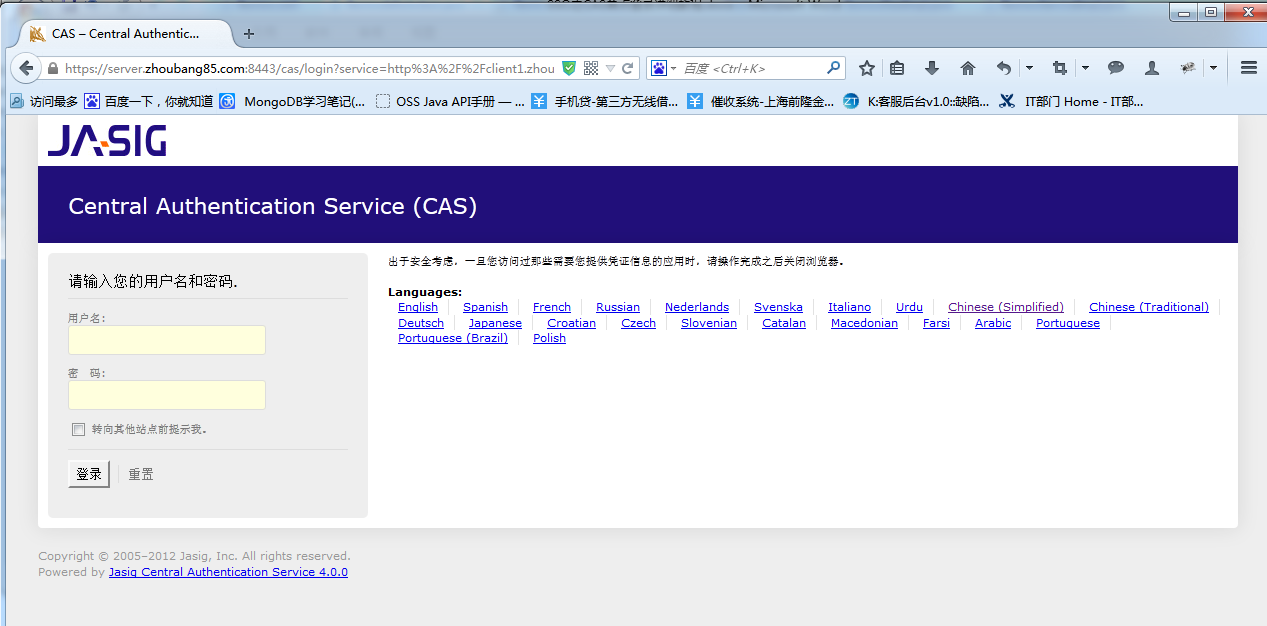
<url-pattern>/\*</url-pattern>

</filter-mapping>

配置完成后，启动CAS服务端tomcat，再启动该客户端1的tomcat，在浏览器访问:

<http://client1.zhoubang85.com:18080/examples/servlets/servlet/HelloWorldExample>

看看是否跳转到了CAS认证界面，回车之后，会出现如下图：

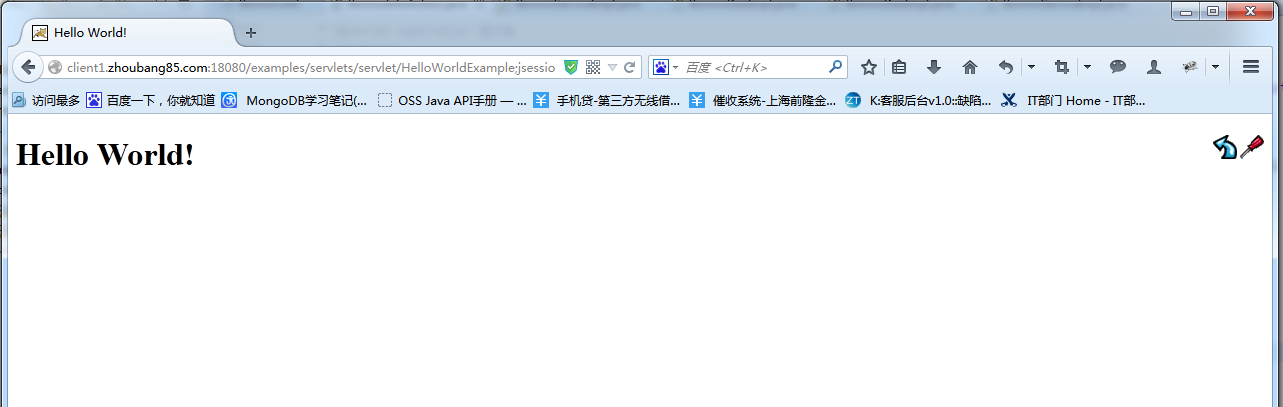


认证观察会发现浏览器的地址栏中（我用的是火狐浏览器），URL信息是这样的：

https://server.zhoubang85.com:8443/cas/login?service=http%3A%2F%2Fclient1.zhoubang85.com%3A18080%2Fexamples%2Fservlets%2Fservlet%2FHelloWorldExample

我想你已经知道什么意思了，由于你没有登录CAS认证系统，CAS认证系统拦截到你访问的客户端应用，首先进入到认证系统登录界面，同时URL后面加上你想访问的地址信息，当你登录成功后，CAS服务会转向到你刚刚访问的地址，也就是<http://client1.zhoubang85.com:18080/examples/servlets/servlet/HelloWorldExample>

转向到这个地址之后，浏览器会显示如下内容：



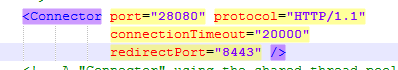
上面这个内容，显示的就是你访问tomcat中的examples项目的一个servlet的返回结果。

这里，如果你够细心的话，你会发现浏览器地址栏又多了一个内容，即多了一个jsessionid参数，这个就是CAS认证的原理所在，使用的是COOKIE机制。这里就不多说了。

**2、配置apache-tomcat-7.0.57-client2客户端2**

配置与上面的客户端配置步骤相同，其中需要注意的就是，这第二个客户端的tomcat端口要与上面的客户端和CAS服务端的端口要不一样，否则出现端口占用的错误。

主要配置如下：



Port修改成28080



Port修改成28009

修改编辑E:\apache-tomcat-7.0.57-client2\webapps\examples\WEB-INF\web.xml文件，加入的内容就是上面在第一个客户端的web.xml内容一致，只不过需要修改2处内容，就是几个url地址， 最终的配置如下:

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

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

<init-param>

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

<param-value>https://server.zhoubang85.com:8443/cas/login</param-value>

</init-param>

<init-param>

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

<param-value>http://client2.zhoubang85.com:28080</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CAS Filter</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.Cas10TicketValidationFilter</filter-class>

<init-param>

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

<param-value>https://server.zhoubang85.com:8443/cas</param-value>

</init-param>

<init-param>

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

<param-value>http://client2.zhoubang85.com:28080</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 Wrapper Filter</filter-name>

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

</filter>

<filter-mapping>

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

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

</filter>

<filter-mapping>

<filter-name>CAS Assertion Thread Local Filter</filter-name>

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

</filter-mapping>

**3、测试验证SSO**

至此，2个客户端也已经配置完毕，启动配置好的三个tomcat分别为：CAS服务端tomcat、2个客户端tomcat

我本地路径分别为：

服务端tomcat：E:\apache-tomcat-7.0.57

客户端1：E:\apache-tomcat-7.0.57-client1

客户端2：E:\apache-tomcat-7.0.57-client2

**4、基本的测试**

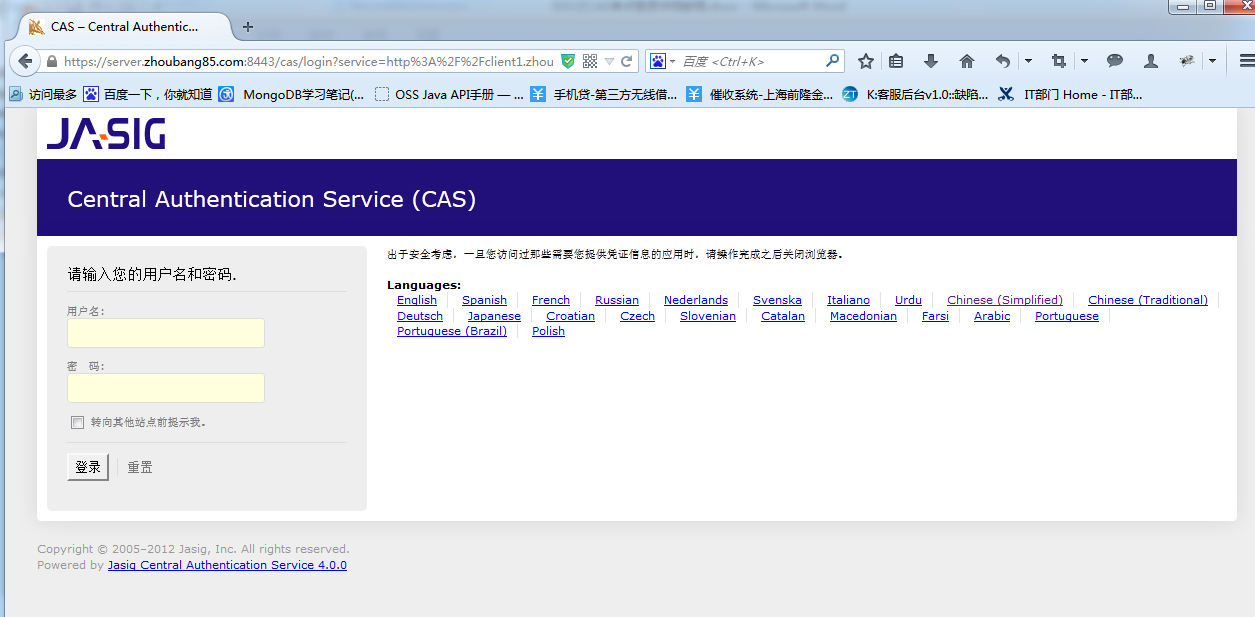
访问客户端1—-> 跳转到cas server 验证 —-> 显示客户端1的应用 —->新开选项卡访问客户端2 —-> 显示客户端2应用 —-> 注销cas server —-> 打开客户端1/客户端2 —-> 重新跳转到cas server 验证.

下面截图，根据上面所示进行测试，看看是否与上面说的流程一致：

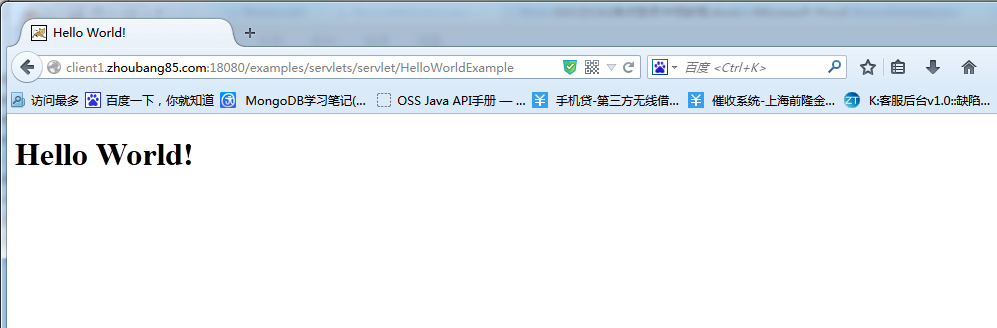
首先，打开火狐浏览器，地址栏输入：

<http://client1.zhoubang85.com:18080/examples/servlets/servlet/HelloWorldExample>

回车，将会出现CAS认证界面：



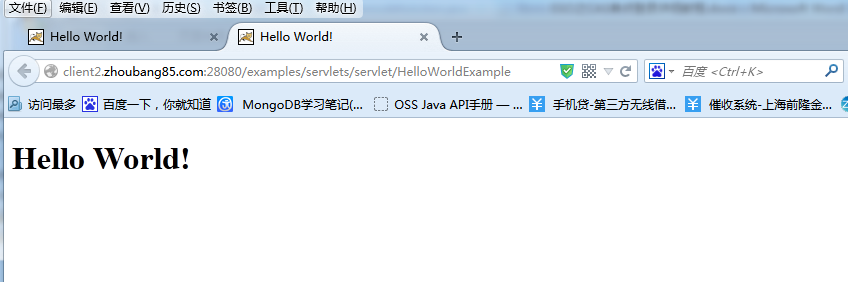
输入用户名和密码，登录成功后，显示的界面：



重新打开一个选项卡，地址栏输入

<http://client2.zhoubang85.com:28080/examples/servlets/servlet/HelloWorldExample>

回车，将直接显示客户端2界面，没有重新登录，浏览器显示界面：



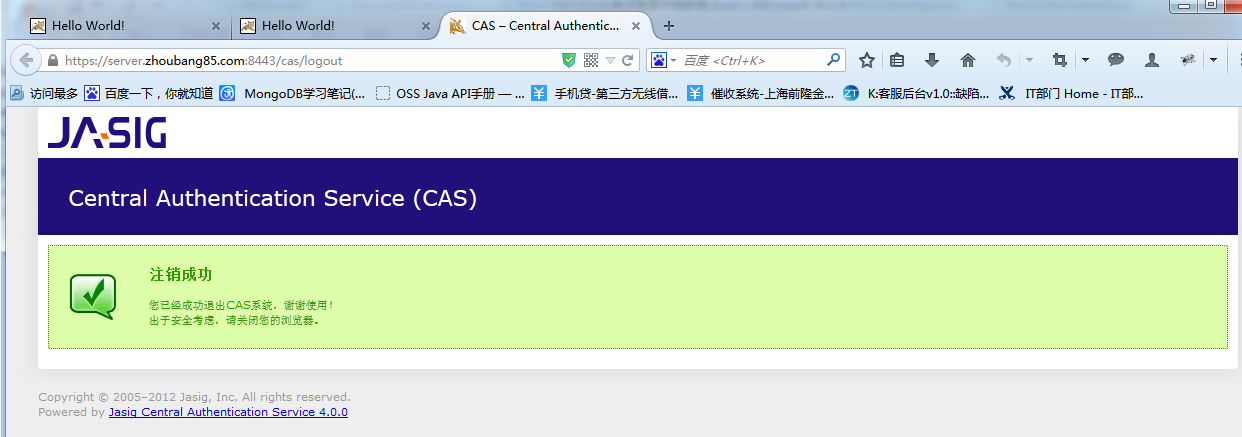
根据上面显示的界面，按照普通想逻辑，当访问客户端2的时候，应该会被CAS拦截，转到CAS认证服务的界面，但是结果却直接显示客户端2的界面，原因就是因为你在访问客户端1的时候已经登录认证过了，CAS会在你浏览器中注入COOKIE，记录你的认证凭证，如果你的浏览器没有关闭或者退出的话，当你访问客户端2应用的时候，CAS检测到认证的凭证，所以，就直接显示了客户端2的界面，大概的认证处理流程就是这样了，我这里说的比较浅显，其实CAS还是有很多更加复杂的操作在里面的，有兴趣的朋友自己慢慢研究吧。

**这里，其实就已经说明，CAS单点登录系统已经搭建完毕！运行一切正常！**

**下面，我们新打开一个选项卡(也可在当前页面的地址栏输入)，在浏览器地址栏中输入**

<https://server.zhoubang85.com:8443/cas/logout>

回车显示：



上述表示 认证注销成功，此时如果再访问 :

<http://client1.zhoubang85.com:18080/examples/servlets/servlet/HelloWorldExample>

或

<http://client2.zhoubang85.com:28080/examples/servlets/servlet/HelloWorldExample>

都将会跳转到CAS服务重新进行认证。

到此，SSO之CAS单点登录系统已经搭建完毕！

这里我借鉴一篇文章，提供大家学习：

<http://www.micmiu.com/enterprise-app/sso/sso-cas-sample/>

当然，CAS认证之后，至于在后台如何获取到登录的用户信息，这里我就不作描述了。该文章最后有相应的讲解。其他的知识，自己慢慢学习和研究吧。

我的邮箱 [zhoubang85@163.com](mailto:zhoubang85@163.com)

QQ: 842324724 请备注添加缘由

我自己的SVN项目，集成许多主流技术，如：SpringBatch批处理、SpringQuartz任务调度、SpringSecurity权限认证、AOP日志拦截、Webservice等。

SVN地址 <http://svn1.okbase.net:8080/svn/2014/zb_myProject>

如果搭建的过程中，发现文档中有写错的地方，请发送邮件说明清楚，我第一时间进行修正。Thanks。