CAS单点登录配置 .

转自：http://blog.csdn.net/zyujie/article/details/6904025

看了咖啡兔的CAS单点登录配置教程，写得非常好，经过了一天的努力，终于配置成功，特将配置步骤记录下来。

1、创建证书
证书是单点登录认证系统中很重要的一把钥匙，客户端于服务器的交互安全靠的就是证书；本教程由于是演示所以就自己用JDK自带的keytool工具生成证书；如果以后真正在产品环境中使用肯定要去证书提供商去购买，证书认证一般都是由VeriSign认证

我们这里采用JDK自带的keytool工具生成证书：keytool -genkey -alias mycas -keyalg RSA -keystore d:/keys/mykey

注意的地方是名字与姓氏就是域名的输入地方，不要用IP地址。

2、导出证书

```
D:\keys>keytool -export -file d:/keys/mycas.crt -alias mycas -keystore d:/keys/mykey
```

3、客户端JVM导入证书

```
keytool -import -keystore D:\jdk\1.6\jdk1.6.0_20\jre\lib\security\cacerts -file D:/keys/mycas.crt -alias mycas
```

值得注意的是，我在配置的过程中报错了，JAVA.IO的一个异常。解决方法是：将jdk目录\jre\lib\security的cacerts删除。

4、将证书应用到web服务器上，这里使用的是tomcat
打开tomcat目录的conf/server.xml文件，8443端处，
并设置keystoreFile、keystorePass修改结果如下：

```
[html] view plaincopyprint?01.<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
02.               maxThreads="150" scheme="https" secure="true"
03.               clientAuth="false" sslProtocol="TLS"
04.           keystoreFile="D:/keys/mykey"
05.           keystorePass="生成KEY的密码"
06./>
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
               maxThreads="150" scheme="https" secure="true"
               clientAuth="false" sslProtocol="TLS"
           keystoreFile="D:/keys/mykey"
           keystorePass="生成KEY的密码"
```

随笔档案

最新评论

/>
参数说明：keystoreFile：在第一步创建的key存放位置 keystorePass：创建证书时的密码

打开https://localhost:8443/，，可以测试

5、配置CAS服务器
cas-server-3.4.3.1.zip解压，解压cas-server-3.4.3/modules/cas-server-webapp-3.4.3.1.war，改名为cas.war，tomcat自动解压开，然后复制cas目录到你的tomcat/webapp目录下

https://localhost:8443/cas/login ，现在打开了CAS服务器的页面输入admin/admin点击登录（CAS默认的验证规则只要用户名和密码相同就通过）系统会提示登录成功。

6、CAS服务器连接数据库的配置
首先打开tomcat/webapp/cas/WEB-INF/deployerConfigContext.xml文件，注释掉SimpleTestUsernamePasswordAuthenticationHandler这个验证Handler

添加如下：JDBC查询接口，数据源，可以加下CAS的MD5加密接口。

```
[html] view plaincopyprint?01.<!--配置JDBC查询接口-->
02.    <bean
class="org.jasig.cas.adaptors.jdbc.QueryDatabaseAuthenticationHandler
">
03.     <property name="dataSource" ref="dataSource" />
04.     <property name="sql" value="select password from t_admin_user
where login_name=?" />
05.     <!--<property name="passwordEncoder"
ref="MD5PasswordEncoder"/>-->
06.    </bean>
07.<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
08.       <property name="driverClassName">
<value>oracle.jdbc.driver.OracleDriver</value></property>
09.       <property name="url">
<value>jdbc:oracle:thin:@127.0.0.1:1521:ORCL</value></property>
10.       <property name="username"><value>zhouyujie</value>
</property>
11.       <property name="password"><value>zhouyujie</value>
</property>
12.    </bean>
13.<!--<bean id="MD5PasswordEncoder"
class="org.jasig.cas.authentication.handler.DefaultPasswordEncoder">
14.     <constructor-arg index="0">
15.      <value>MD5</value>
16.     </constructor-arg>
17.    </bean>-->
<!--配置JDBC查询接口-->
    <bean
class="org.jasig.cas.adaptors.jdbc.QueryDatabaseAuthenticationHandler
">
     <property name="dataSource" ref="dataSource" />
```

--miss you

阅读**排行榜**

1. weblogic线程阻塞性能调优（图解）(7087)
2. struts2标签#、%、$取值(5300)
3. EL表达式取值(3194)
4. 解决(CXF)：SOAPFaultException: Fault occurred while processing(1581)
5. NoGoalSpecifiedException : http://cwiki.apache.org/confluence

评论**排行榜**

1. spring mvc+freemarker整合(非注解方式)(2)
2. SVN服务端搭建(图解)(1)
3. java生成json总结(1)
4. JAVA反射机制(1)
5. js获取当前日期时间并赋值ext的datefield(1)

推荐排行榜

1. JAVA反射机制(1)
2. struts2标签#、%、$取值(1)

```
     <property name="sql" value="select password from t_admin_user
where login_name=?" />
     <!--<property name="passwordEncoder" ref="MD5PasswordEncoder"/>-
->
    </bean>
<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
       <property name="driverClassName">
<value>oracle.jdbc.driver.OracleDriver</value></property>
       <property name="url">
<value>jdbc:oracle:thin:@127.0.0.1:1521:ORCL</value></property>
       <property name="username"><value>zhouyujie</value></property>
       <property name="password"><value>zhouyujie</value></property>
    </bean>
<!--<bean id="MD5PasswordEncoder"
class="org.jasig.cas.authentication.handler.DefaultPasswordEncoder">
     <constructor-arg index="0">
      <value>MD5</value>
     </constructor-arg>
    </bean>-->
```

复制cas-server-3.4.3.1\modules\cas-server-support-jdbc-3.4.3.1.jar和mysql驱动jar包到tomcat/webapp/cas/WEB-INF/lib目录

7、CAS客户端的配置

在你的客户端工程中添加一个cas-client-core-3.2.1.jar包。

web.xml中添加如下信息：

```
[html] view plaincopyprint?01.<!-- 用于单点退出，该过滤器用于实现单点登出功
能，可选配置-->
02.<listener>
03.
<listener-class>org.jasig.cas.client.session.SingleSignOutHttpSession
Listener</listener-class>
04.</listener>
05.
06.<!-- 该过滤器用于实现单点登出功能，可选配置。 -->
07.<filter>
08.    <filter-name>CAS Single Sign Out Filter</filter-name>
09.
<filter-class>org.jasig.cas.client.session.SingleSignOutFilter</filte
r-class>
10.</filter>
11.<filter-mapping>
12.    <filter-name>CAS Single Sign Out Filter</filter-name>
13.    <url-pattern>/*</url-pattern>
14.</filter-mapping>
15.
16.<!-- 该过滤器负责用户的认证工作，必须启用它 -->
17.<filter>
18.    <filter-name>CASFilter</filter-name>
19.    <filter-
class>org.jasig.cas.client.authentication.AuthenticationFilter</filte
```

```
r-class>
    <init-param>
        <param-name>casServerLoginUrl</param-name>
        <param-value>https://localhost:8443/cas/login</param-value>
        <!--这里的server是服务端的IP-->
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value>http://localhost:8080</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>
        <param-value>https://localhost:8443/cas</param-value>
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value>http://localhost:8080</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>
<!-- ========================= 单点登录结束 ========================= -->
<!-- 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>

<!-- 该过滤器负责用户的认证工作，必须启用它 -->
<filter>
    <filter-name>CASFilter</filter-name>
    <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>
    <init-param>
        <param-name>casServerLoginUrl</param-name>
        <param-value>https://localhost:8443/cas/login</param-value>
        <!--这里的server是服务端的IP-->
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value>http://localhost:8080</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.Cas20ProxyReceivingTicketValidationFi
lter</filter-class>
    <init-param>
        <param-name>casServerUrlPrefix</param-name>
        <param-value>https://localhost:8443/cas</param-value>
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value>http://localhost:8080</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>
 <!-- ======================== 单点登录结束 ======================== --
>
```

8、CAS的登录页面修改

可以对CAS的登录页面进行外观上的修改，实际项目中肯定需要用到。

9、CAS在客户端页面上的获取

在页面上，我们可以通过`String uname = request.getRemoteUser();`获得CAS中的用户名信息。

再次感谢咖啡兔的教程，呵呵，^0^。。。

分类: spring

miss you
关注 - 0
粉丝 - 3
+加关注

« 上一篇：Static attribute must be a String literal
» 下一篇：cas 配置数据源，解决CAS 不支持你提供的凭证.

posted @ 2014-03-12 09:39 miss you 阅读(402) 评论(0) 编辑 收藏

» 更多知识库文章...