7、客户端集成,跳转登录

# 接5或者6中项目

## 解释：客户端代码，模拟客户端官方代码 <https://github.com/apereo/java-cas-client> （我这里直接使用的高人的客户端代码）

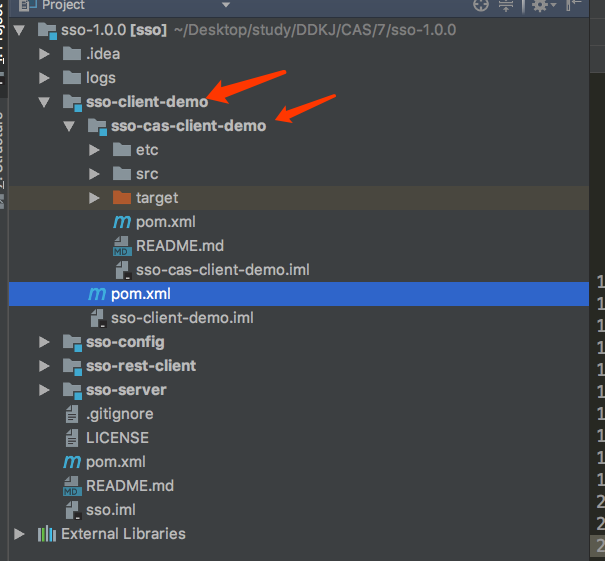
# 1、了解cas单点登录的全过程

## 解释：cas client如何强大方便，都是遵循着cas协议进行认证，否则是不能完成工作的。

### 客户端地址：<http://localhost:8080/sample>  cas服务地址：[https://passport.com](https://passport.com/)

### 协议流程：  1. 若业务系统未登录，302到[https://passport.com?service=http://localhost:8080/sample](https://passport.com/?service=http://localhost:8080/sample)  2. 用户提交用户名密码后，302到<http://localhost:8080/sample?ticket=ABC123>  3. 业务系统验证ticket，并获取用户数据，<https://passport.com/p3/serviceValidate?service=p3/serviceValidate?service&ticket=ABC123>  4. 成功获取用户数据

# 2、创建maven项目（聚合工程）



## 1、父sso-client-demo<亲pom如下

*<?*xml version="1.0" encoding="UTF-8"*?>*<!--  
 ~ 版权所有.(c)2008-2017. 卡尔科技工作室  
 -->  
  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <parent>  
 <artifactId>sso</artifactId>  
 <groupId>com.carl.auth</groupId>  
 <version>1.0.0</version>  
 <relativePath>../pom.xml</relativePath>  
 </parent>  
 <modelVersion>4.0.0</modelVersion>  
 <packaging>pom</packaging>  
 <modules>  
 <module>sso-cas-client-demo</module>  
 </modules>  
 <artifactId>sso-client-demo</artifactId>  
  
</project>

## 2、子工程（这里的工程就是相当于是官方的客户端代码服务器使用的是jetty）sso-cas-client-demo pom如下

*<?*xml version="1.0" encoding="UTF-8"*?>*<!--  
 ~ 版权所有.(c)2008-2017. 卡尔科技工作室  
 -->  
  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <parent>  
 <artifactId>sso-client-demo</artifactId>  
 <groupId>com.carl.auth</groupId>  
 <version>1.0.0</version>  
 <relativePath>../pom.xml</relativePath>  
 </parent>  
 <modelVersion>4.0.0</modelVersion>  
  
 <artifactId>sso-cas-client-demo</artifactId>  
  
 <packaging>war</packaging>  
 <description>A sample web application that exercises the CAS protocol features via the Java CAS Client.</description>  
 <build>  
 <finalName>${project.artifactId}</finalName>  
 <plugins>  
 <plugin>  
 <groupId>org.eclipse.jetty</groupId>  
 <artifactId>jetty-maven-plugin</artifactId>  
 <version>9.3.6.v20151106</version>  
 <configuration>  
 <jettyXml>${basedir}/etc/jetty/jetty.xml</jettyXml>  
 <systemProperties>  
 <systemProperty>  
 <name>org.eclipse.jetty.annotations.maxWait</name>  
 <value>300</value>  
 </systemProperty>  
 </systemProperties>  
 <webApp>  
 <contextPath>/sample</contextPath>  
 <overrideDescriptor>${basedir}/etc/jetty/web.xml</overrideDescriptor>  
 </webApp>  
 <jvmArgs>-Xdebug -Xrunjdwp:transport=dt\_socket,address=5002,server=y,suspend=n</jvmArgs>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
 <dependencies>  
  
 <dependency>  
 <groupId>commons-logging</groupId>  
 <artifactId>commons-logging</artifactId>  
 <version>1.1.1</version>  
 </dependency>  
  
 <dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.17</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.jasig.cas.client</groupId>  
 <artifactId>cas-client-core</artifactId>  
 <version>3.4.1</version>  
 <exclusions>  
 <exclusion>  
 <groupId>javax.servlet</groupId>  
 <artifactId>servlet-api</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.jasig.cas.client</groupId>  
 <artifactId>cas-client-core</artifactId>  
 <version>RELEASE</version>  
 </dependency>  
 </dependencies>  
</project>

# 3、查看这个工程的web.xml，其实可以知道这里的工程其实是有一个jsp的工程 （和1中的客户端其实是一样的）

*<?*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">  
  
 <!--  
 <context-param>  
 <param-name>renew</param-name>  
 <param-value>true</param-value>  
 </context-param>  
 -->

<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>

<param-value>https://passport.sso.com:8443/cas</param-value>

</init-param>

</filter>

<listener>  
 <listener-class>org.jasig.cas.client.session.SingleSignOutHttpSessionListener</listener-class>  
 </listener>  
  
 <filter> **<!-- 该过滤器负责用户的认证工作，必须启用它 -->**   
 <filter-name>CAS Authentication Filter</filter-name>  
 <!--<filter-class>org.jasig.cas.client.authentication.Saml11AuthenticationFilter</filter-class>-->  
 <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>  
 <init-param>  
 <param-name>casServerLoginUrl</param-name>  
 <param-value>https://passport.sso.com:8443/cas/login</param-value>

</init-param>  
 <init-param>  
 <param-name>serverName</param-name>  
 <param-value>http://localhost:8080</param-value>

</init-param>  
 </filter>  
 **<!-- 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.Saml11TicketValidationFilter</filter-class>-->  
 <filter-class>org.jasig.cas.client.validation.Cas30ProxyReceivingTicketValidationFilter</filter-class>  
 <init-param>  
 <param-name>casServerUrlPrefix</param-name>  
 <param-value>https://passport.sso.com:8443/cas</param-value>

</init-param>  
 <init-param>  
 <param-name>serverName</param-name>  
 <param-value>http://localhost:8080</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>acceptAnyProxy</param-name>  
 <param-value>true</param-value>  
 </init-param>  
 <init-param>  
 <param-name>proxyReceptorUrl</param-name>  
 <param-value>/sample/proxyUrl</param-value>  
 </init-param>  
 <init-param>  
 <param-name>proxyCallbackUrl</param-name>  
 <param-value>https://mmoayyed.unicon.net:9443/sample/proxyUrl</param-value>  
 </init-param>  
 -->  
 <init-param>  
 <param-name>authn\_method</param-name>  
 <param-value>mfa-duo</param-value>  
 </init-param>  
 </filter>  
  
 <filter>  
 <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>  
 <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>  
 </filter>

<!-- 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>

<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>

## 解释：以上的核心点有两个：

* **serverName 客户端访问路径**
* **casServerUrl cas服务路径**

**但上面的配置非常的简单，除了被排除的所有路径都会跳转进行登录**

**再加一个需求，判断路径是否需要跳转到登录页再跳转，那么不得不介绍一下AuthenticationFilter的一些简单配置：**

| **属性名** | **类型** | **备注** | **默认值** | **是否必须** |
| --- | --- | --- | --- | --- |
| **casServerUrlPrefix** | **string** | **cas服务路径** |  | **必须** | |
| **serverName** | **string** | **客户端访问路径** |  | **必须** | |
| **renew** | **boolean** | **验证成功是否新创建会话** | **true** | **非必须** | |
| **service** | **string** | **服务路径** |  | **非必须** | |
| **ignorePattern** | **string** | **忽略登录路径正则表达式** |  | **非必须** | |
| **ignoreUrlPatternType** | **string** | **忽略路径表达式类型（可以配置实现UrlPatternMatcherStrategy类路径）** |  | **非必须** | |

# 4、现在我想添加两个页面，一个页面zhangsan.jsp不需要登录，wangwu一个页面必须登录，

## 1、过滤器中添加自己配置的过滤对象 ，认证过滤器中添加

<filter>  
 <filter-name>CAS Authentication Filter</filter-name>  
 <!--<filter-class>org.jasig.cas.client.authentication.Saml11AuthenticationFilter</filter-class>-->  
 <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>  
 <init-param>  
 <param-name>casServerLoginUrl</param-name>  
 <param-value>https://passport.sso.com:8443/cas/login</param-value>  
 </init-param>  
 <init-param>  
 <param-name>serverName</param-name>  
 <param-value>http://localhost:8080</param-value>  
 </init-param>  
 <init-param>  
 <param-name>ignorePattern</param-name>  
 <param-value>.\*</param-value>  
 </init-param>  
 <init-param>  
 <param-name>ignoreUrlPatternType</param-name>  
 <param-value>com.carl.auth.sso.client.demo.SimpleUrlPatternMatcherStrategy</param-value>  
 </init-param>  
</filter>

## 2、过滤url的匹配 UrlPatternMatcherStrategy

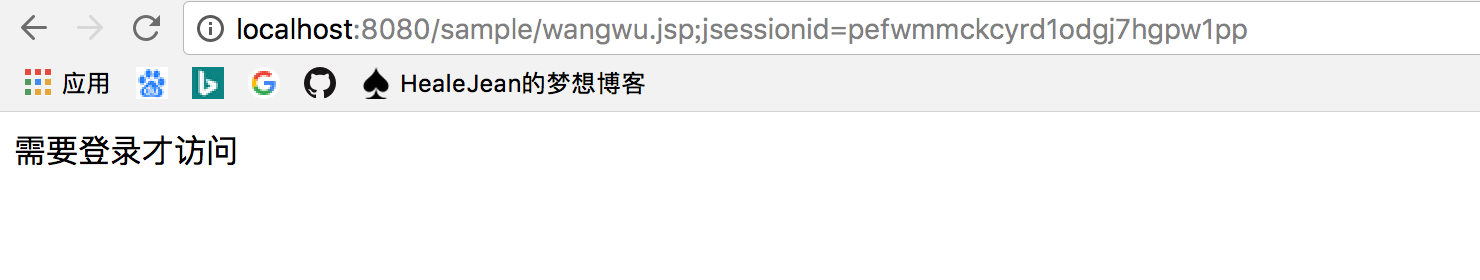
public class SimpleUrlPatternMatcherStrategy implements UrlPatternMatcherStrategy {  
 protected final Logger logger = LoggerFactory.getLogger(getClass());  
  
 @Override  
 public boolean matches(String url) {  
 logger.debug("访问路径：" + url);  
 return url.contains("zhangsan.jsp");  
 }  
  
 @Override  
 public void setPattern(String pattern) {  
  
 }  
}

# 5、启动 sso-config sso-rest-client sso-server 最后启动这个项目sso-cas-client-demo （使用命令 mvn jetty:run 启动）

## 1、浏览器中访问 <http://localhost:8080/sample/wangwu.jsp>

### 1、地址变化为 <https://passport.sso.com:8443/cas/login?service=http%3A%2F%2Flocalhost%3A8080%2Fsample%2Fwangwu.jsp> （进入登录页面）

### 2、输入用户名和密码，然后点击登录，登录成功 地址变化为 <http://localhost:8080/sample/wangwu.jsp;jsessionid=pefwmmckcyrd1odgj7hgpw1pp>

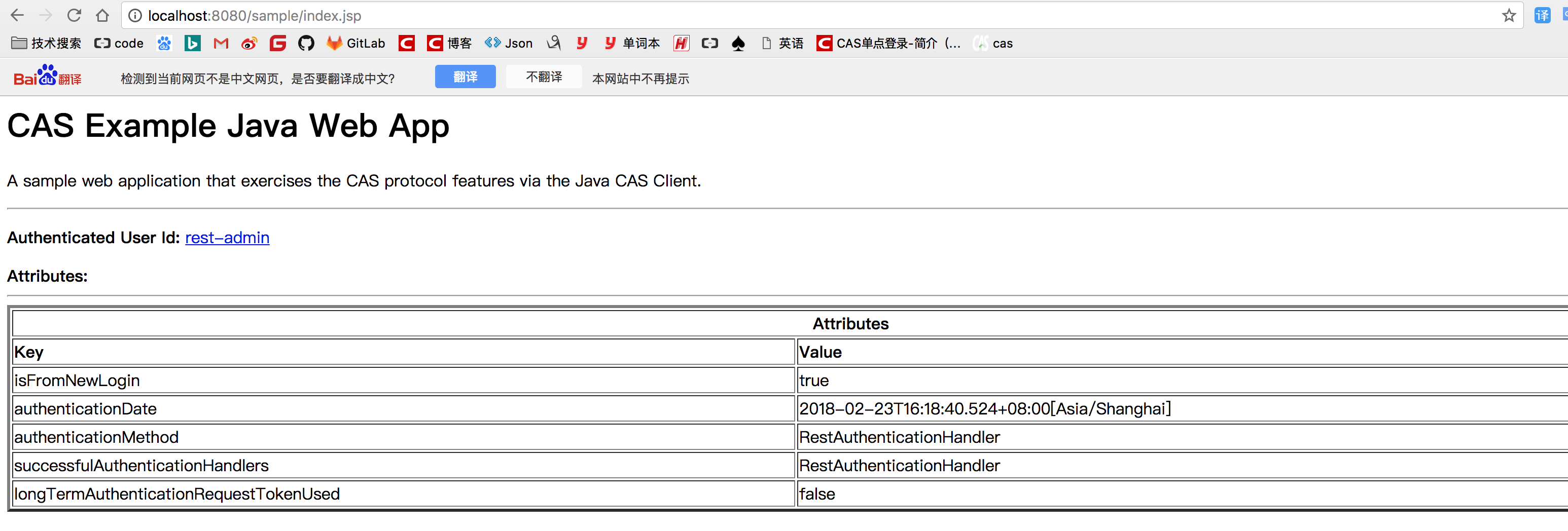


## 2、浏览器中访问 <http://localhost:8080/sample/zhangsan.jsp> 直接就可以访问了



# 6、获取登录用户数据

## 1、index.jsp



## 下面二者都是登录的用户名

request.getRemoteUser()  
AssertionHolder.getAssertion().getPrincipal().getName())