西北师范大学CAS统一认证系统与消息接口对接文档（Version 1.0.0）

# 消息接口

### **1.1 消息接口功能**

门户系统中集成了多个域名的业务系统，由于浏览器同源策略(Same-Origin Policy)的访问约束。所谓同源是指，域名，协议，端口相同。不同源的客户端脚本(javascript、ActionScript)在没明确授权的情况下，不能读写对方的资源。所以选择JSONP同步多个业务系统的数据访问。

JSONP（JSON with Padding）是JSON的一种”使用模式”，可用于解决主流浏览器的跨域数据访问的问题。由于同源策略，一般来说位于 server1.example.com 的网页无法与不是 server1.example.com的服务器沟通，而 HTML 的script 元素是一个例外。利用 <script> 元素的这个开放策略，网页可以得到从其他来源动态产生的 JSON 资料，而这种使用模式就是所谓的 JSONP。用 JSONP 抓到的资料并不是 JSON，而是任意的JavaScript，用 JavaScript 直译器执行而不是用 JSON 解析器解析。

### **1.2业务系统返回待处理消息条数接口**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口名** | **请求URL** | **功能说明** | **参数说明** | **结果说明** |
| 待处理消息条数接口 | 由业务系统自定义URL | 调用该接口时指定返回待处理消息条数 | userId=？参数userId为CAS的登录名。 | 结果以票据形式返回,获取请参照下述客户端配置 |

示例：

1. 访问该URL返回待办数量：

如OA的访问地址为[http://192.168.1.99](Http://192.168.1.99),那么获取待办数量的URL为

<Http://192.168.1.99>/seeyon/byGetWaitHandle.do?method=getWaitHandle&userCode=002604

其中参数为：userId=CAS的登录名

1. 使用http请求访问该地址，返回后的格式为：callback({"totalCounts":"6"});
2. 业务系统具体实现代码：

/\*\*  
 \* 获取当前用户没有读的代办事项数据总数  
 \*/  
public void achivementAuditCount(HttpServletRequest request,

HttpServletResponse response)

{

Request.getParameter(“userId”);  
long count = 0;

/\*业务系统用来获取未读待办事项总数;给count赋值\*/  
response.setContentType("text/xml");  
response.setCharacterEncoding("utf-8");  
PrintWriter out = null;  
try{

out = response.getWriter();  
  }

catch (IOException e) {  
      e.printStackTrace();  
    }

out.print("callback" + "({" + "\"totalCounts\":" + "\""+count+"\"});");  
out.flush();  
out.close();

}

### **1.3业务系统已处理消息更新接口**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口名** | **请求URL** | **功能说明** | **参数说明** | **结果说明** |
| 更新待处理消息条数接口 | **http://p.nwnu.edu.cn/changecount** | 调用该接口时触发门户系统刷新待处理消息条数 | userId=？参数userId为CAS的登录名。 | 请求url中域名后期会有调整。 |

# CAS接口介绍

### 2.1业务功能

业务系统调用CAS接口，页面跳转到CAS系统。用户完成登录或注册后，CAS返回Ticket并跳转到业务系统页面,业务系统导入的CAS客户端获取Ticket并验证合法性后,获取CAS服务端返回的用户信息。

### 2.2交互模式

请求：页面跳转交互模式

返回结果：页面跳转交互模式

### 2.3页面接口介绍

[**cas.castest.cc]需替换为实际发布的域名**

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名** | **请求URL** | **功能说明** | **结果说明** |
| CAS登录接口 | **http://cas.castest.cc/login** | 调用该接口时指定返回URL，完成登录后，采用页面跳转交互模式返回 | 结果以票据形式返回,获取请参照下述客户端配置 |
| CAS修改密码接口 | **http://cas.castest.cc/changePwd** | 调用该接口时指定返回URL，完成修改密码后，采用页面跳转交互模式返回 | 结果以票据形式返回,获取请参照下述客户端配置 |
| CAS注销接口 | **http、://cas.castest.cc/logout** | 调用该接口时指定返回URL，完成注销后，采用页面跳转交互模式返回 | 结果以票据形式返回,获取请参照下述客户端配置 |

# CAS登录接口

### 2.1 CAS登录接口请求参数列表

请求url：**http://www.castest.cc/cas/login**

通过get请求

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **变量名** | **必填** | **类型** | **说明** |
| **业务参数** | | | | |
| 返回URL | service | 是 | String | 不能为空,设置为空时无法返回 |

### 2.2返回结果

页面返回通过请求中的返回URL进行

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **变量名** | **必填** | **类型** | **说明** |
| **业务参数** | | | | |
| CAS票据 | \_const\_cas\_assertion\_ | 是 | Assertion | 票据结构:  attributes(Map<String,object>)  {用户信息:UserInfo(Json形式)}  数据：  loginName:用户工号  userName:用户姓名  userRole:用户角色 |

# CAS修改密码接口

### 3.1 CAS修改密码接口请求参数列表

请求url：**http://www.castest.cc/cas/changePwd**

通过get请求

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **变量名** | **必填** | **类型** | **说明** |
| **业务参数** | | | | |
| 返回URL | service | 是 | String | 不能为空,设置为空时无法返回 |

### 6.2返回结果

页面返回通过请求中的返回URL进行

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **变量名** | **必填** | **类型** | **说明** |
| **业务参数** | | | | |
| CAS票据 | \_const\_cas\_assertion\_ | 是 | Assertion | 票据结构:  attributes(Map<String,object>)  {用户信息:UserInfo(Json形式)} |

# CAS注销接口

### 7.1 CAS注销接口请求参数列表

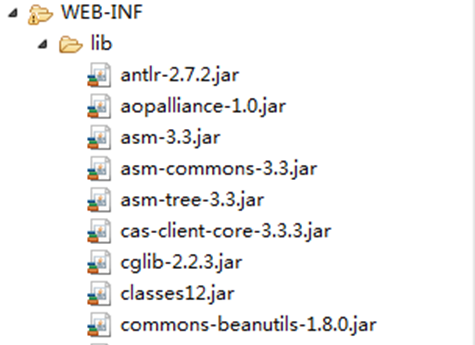
请求url：**http://www.castest.cc/cas/logout**

通过get请求

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **变量名** | **必填** | **类型** | **说明** |
| **业务参数** | | | | |
| 返回URL | service | 是 | String | 不能为空,设置为空时无法返回 |

# CAS JAVA客户端配置

### 8.1 业务系统导入CAS客户端LIB: cas-client-core-3.3.3.jar



### 8.2 业务系统导入CAS 客户端用Filter配置

**Java工程的配置文件[Web.xml]里添加CAS 客户端用Filter配置**

8.2.1 单点登录注销(可选)

<listener>

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

</listener-class>

</listener>

<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>

8.2.2 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)

<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>

8.2.3 用户认证(必须)

<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://www.castest.cc/cas/login</param-value>

</init-param>

<init-param>

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

<param-value>http://www.castest.cc</param-value>

</init-param>

</filter>

<filter-mapping>

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

<url-pattern>/login</url-pattern>

</filter-mapping>

8.2.4 Ticket校验(必须)

<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://www.castest.cc/cas</param-value>

</init-param>

<init-param>

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

<param-value>http://www.castest.cc</param-value>

</init-param>

</filter>

<filter-mapping>

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

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

</filter-mapping>

8.2.5允许开发者通过AssertionHolder获得SSO登录用户的登录名(必须)

<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>

8.2.6 自动根据单点登录的结果设置本系统的用户信息(可选)

\* AutoSetUserAdapterFilter为商户系统自定义文件,请参考商户调用范例

<filter>

<display-name>AutoSetUserAdapterFilter</display-name>

<filter-name>AutoSetUserAdapterFilter</filter-name>

<filter-class>

商户系统.AutoSetUserAdapterFilter

</filter-class>

</filter>

<filter-mapping>

<filter-name>AutoSetUserAdapterFilter</filter-name>

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

</filter-mapping>

### 8.3 商户系统调用范例

8.3.1 登录

https://www.castest.cc/cas/login?service=http://www.castest.cc/yw/index

8.3.2 注销

https://www.castest.cc/cas/logout?service=http://www.castest.cc/yw/index

8.3.3 修改密码

https://www.castest.cc/cas/changePwd?service=http://www.castest.cc/ yw/index

8.3.5自动根据单点登录的结果设置本系统的用户信息(范例,业务系统自行定义内容)

**import java.io.IOException;**

**import java.util.Map;**

**import javax.servlet.Filter;**

**import javax.servlet.FilterChain;**

**import javax.servlet.FilterConfig;**

**import javax.servlet.ServletException;**

**import javax.servlet.ServletRequest;**

**import javax.servlet.ServletResponse;**

**import javax.servlet.http.HttpServletRequest;**

**import org.jasig.cas.client.util.AbstractCasFilter;**

**import org.jasig.cas.client.validation.Assertion;**

**public class AutoSetUserAdapterFilter implements Filter{**

**@Override**

**/\*\***

**\* 过滤逻辑：首先判断单点登录的账户是否已经存在本系统中，**

**\* 如果不存在使用用户查询接口查询出用户对象并设置在Session中**

**\* @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)**

**\*/**

**public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException,**

**ServletException {**

**HttpServletRequest httpRequest = (HttpServletRequest) request;**

**// \_const\_cas\_assertion\_是CAS中存放登录用户名的session标志**

**//获取CAS服务端设置的用户信息**

**Object object = httpRequest.getSession().getAttribute(AbstractCasFilter.CONST\_CAS\_ASSERTION);**

**if (object != null) {**

**//获取用户信息**

**Assertion assertion = (Assertion) object;**

**//获取用户名**

**String loginName = assertion.getPrincipal().getName();**

**//业务系统可通过上面获取的用户名进行本系统的认证处理**

**} else**

**{**

**System.out.println("null Name");**

**}**

**chain.doFilter(request, response);**

**}**

**@Override**

**public void destroy() {**

**// TODO Auto-generated method stub**

**}**

**@Override**

**public void init(FilterConfig arg0) throws ServletException {**

**// TODO Auto-generated method stub**

**}**

**}**

### 8.4 采用SHIRO权限管理的商户系统单点统一注销调用范例

8.4.1 商户系统导入CasLogoutFilter.java

**package** com.thinkgem.javamg.common.security.shiro;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**import** javax.servlet.http.HttpServletRequest;

**import** org.apache.shiro.SecurityUtils;

**import** org.apache.shiro.session.Session;

**import** org.apache.shiro.session.SessionException;

**import** org.apache.shiro.session.mgt.SessionManager;

**import** org.apache.shiro.subject.Subject;

**import** org.apache.shiro.web.servlet.AdviceFilter;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**public** **class** CasLogoutFilter **extends** AdviceFilter{

**private** **static** **final** Logger ***log*** = LoggerFactory.*getLogger*(CasLogoutFilter.**class**);

**private** **static** **final** SingleSignOutHandler ***HANDLER*** = **new** SingleSignOutHandler();

**private** SessionManager sessionManager;

**public** **void** setSessionManager(SessionManager sessionManager) {

**this**.sessionManager = sessionManager;

}

/\*\*

\* 如果请求中包含了ticket参数，记录ticket和sessionID的映射

\* 如果请求中包含logoutRequest参数，标记session为无效

\* 如果session不为空，且被标记为无效，则登出

\*

\* **@param** request the incoming ServletRequest

\* **@param** response the outgoing ServletResponse

\* **@return** 是logoutRequest请求返回false，否则返回true

\* **@throws** Exception if there is any error.

\*/

@Override

**protected** **boolean** preHandle(ServletRequest request, ServletResponse response) **throws** Exception {

HttpServletRequest req = (HttpServletRequest)request;

**if** (***HANDLER***.isTokenRequest((HttpServletRequest)req)) {

//通过浏览器发送的请求，链接中含有token参数，记录token和sessionID

***HANDLER***.recordSession(req);

**return** **true**;

} **else** **if** (***HANDLER***.isLogoutRequest(req)) {

//cas服务器发送的请求，链接中含有logoutRequest参数，在之前记录的session中设置logoutRequest参数为true

//因为Subject是和线程是绑定的，所以无法获取登录的Subject直接logout

***HANDLER***.invalidateSession(req,sessionManager);

// Do not continue up filter chain

**return** **false**;

} **else** {

***log***.trace("Ignoring URI " + req.getRequestURI());

}

Subject subject = SecurityUtils.*getSubject*();

Session session = subject.getSession(**false**);

**if** (session!=**null**&&session.getAttribute(***HANDLER***.getLogoutParameterName())!=**null**) {

**try** {

subject.logout();

} **catch** (SessionException ise) {

***log***.debug("Encountered session exception during logout. This can generally safely be ignored.", ise);

}

}

**return** **true**;

}

}

8.4.2 商户系统导入HashMapBackedSessionMappingStorage.java

**package com.thinkgem.javamg.common.security.shiro;**

**import org.apache.shiro.session.Session;**

**import java.io.Serializable;**

**import java.util.HashMap;**

**import java.util.Map;**

**/\*\***

**\* 存储ticket到sessionID的映射**

**\*/**

**public final class HashMapBackedSessionMappingStorage {**

**/\*\***

**\* Maps the ID from the CAS server to the Session ID.**

**\*/**

**private final Map<String,Serializable> MANAGED\_SESSIONS\_ID = new HashMap<String,Serializable>();**

**public synchronized void addSessionById(String mappingId, Session session) {**

**MANAGED\_SESSIONS\_ID.put(mappingId, session.getId());**

**}**

**public synchronized Serializable getSessionIDByMappingId(String mappingId) {**

**return MANAGED\_SESSIONS\_ID.get(mappingId);**

**}**

**}**

8.4.3 商户系统导入SingleSignOutHandler.java

**package** com.thinkgem.javamg.common.security.shiro;

**import** java.io.Serializable;

**import** javax.servlet.http.HttpServletRequest;

**import** org.apache.commons.logging.Log;

**import** org.apache.commons.logging.LogFactory;

**import** org.apache.shiro.SecurityUtils;

**import** org.apache.shiro.session.Session;

**import** org.apache.shiro.session.mgt.DefaultSessionKey;

**import** org.apache.shiro.session.mgt.SessionManager;

**import** org.jasig.cas.client.util.CommonUtils;

**import** org.jasig.cas.client.util.XmlUtils;

/\*\*

\* Performs CAS single sign-out operations in an API-agnostic fashion.

\*

\*/

**public** **final** **class** SingleSignOutHandler {

/\*\* Logger instance \*/

**private** **final** Log log = LogFactory.*getLog*(getClass());

/\*\* The name of the artifact parameter. This is used to capture the session identifier. \*/

**private** String artifactParameterName = "ticket";

/\*\* Parameter name that stores logout request \*/

**private** String logoutParameterName = "logoutRequest";

**private** HashMapBackedSessionMappingStorage storage = **new** HashMapBackedSessionMappingStorage();

**protected** SingleSignOutHandler(){

init();

}

/\*\*

\* **@param** name Name of the authentication token parameter.

\*/

**public** **void** setArtifactParameterName(**final** String name) {

**this**.artifactParameterName = name;

}

/\*\*

\* **@param** name Name of parameter containing CAS logout request message.

\*/

**public** **void** setLogoutParameterName(**final** String name) {

**this**.logoutParameterName = name;

}

**protected** String getLogoutParameterName() {

**return** **this**.logoutParameterName;

}

/\*\*

\* Initializes the component for use.

\*/

**public** **void** init() {

CommonUtils.*assertNotNull*(**this**.artifactParameterName, "artifactParameterName cannot be null.");

CommonUtils.*assertNotNull*(**this**.logoutParameterName, "logoutParameterName cannot be null.");

}

/\*\*

\* Determines whether the given request contains an authentication token.

\*

\* **@param** request HTTP reqest.

\*

\* **@return** True if request contains authentication token, false otherwise.

\*/

**public** **boolean** isTokenRequest(**final** HttpServletRequest request) {

**return** CommonUtils.*isNotBlank*(CommonUtils.*safeGetParameter*(request, **this**.artifactParameterName));

}

/\*\*

\* Determines whether the given request is a CAS logout request.

\*

\* **@param** request HTTP request.

\*

\* **@return** True if request is logout request, false otherwise.

\*/

**public** **boolean** isLogoutRequest(**final** HttpServletRequest request) {

**return** "POST".equals(request.getMethod()) && !isMultipartRequest(request) &&

CommonUtils.*isNotBlank*(CommonUtils.*safeGetParameter*(request, **this**.logoutParameterName));

}

/\*\*

\* 记录请求中的token和sessionID的映射对

\*

\* **@param** request HTTP request containing an authentication token.

\*/

**public** **void** recordSession(**final** HttpServletRequest request) {

Session session = SecurityUtils.*getSubject*().getSession();

**final** String token = CommonUtils.*safeGetParameter*(request, **this**.artifactParameterName);

**if** (log.isDebugEnabled()) {

log.debug("Recording session for token " + token);

}

storage.addSessionById(token, session);

}

/\*\*

\* 从logoutRequest参数中解析出token，根据token获取到sessionID，再根据sessionID获取到session，设置logoutRequest参数为true

\* 从而标记此session已经失效。

\*

\* **@param** request HTTP request containing a CAS logout message.

\*/

**public** **void** invalidateSession(**final** HttpServletRequest request, **final** SessionManager sessionManager) {

**final** String logoutMessage = CommonUtils.*safeGetParameter*(request, **this**.logoutParameterName);

**if** (log.isTraceEnabled()) {

log.trace ("Logout request:\n" + logoutMessage);

}

**final** String token = XmlUtils.*getTextForElement*(logoutMessage, "SessionIndex");

**if** (CommonUtils.*isNotBlank*(token)) {

Serializable sessionId = storage.getSessionIDByMappingId(token);

**if** (sessionId!=**null**) {

**try** {

Session session = sessionManager.getSession(**new** DefaultSessionKey(sessionId));

**if**(session != **null**) {

//设置会话的logoutParameterName 属性表示无效了，这里直接使用了request的参数名

session.setAttribute(logoutParameterName, **true**);

**if** (log.isDebugEnabled()) {

log.debug ("Invalidating session [" + sessionId + "] for token [" + token + "]");

}

}

} **catch** (Exception e) {

}

}

}

}

**private** **boolean** isMultipartRequest(**final** HttpServletRequest request) {

**return** request.getContentType() != **null** && request.getContentType().toLowerCase().startsWith("multipart");

}

}

8.4.4 商户系统的shiro权限配置xml（例：spring-context-shiro.xml）添加如下内容，\*为导入java文件的路径

<bean id=*"casLogoutFilter"* class=*"\*.CasLogoutFilter"*>

<property name=*"sessionManager"* ref=*"sessionManager"*/>

</bean>

8.4.5 商户系统的shiro权限配置xml（例：spring-context-shiro.xml）的安全认证过滤器中增加casLogout这一项

<!-- 安全认证过滤器 -->

<bean id=*"shiroFilter"* class=*"org.apache.shiro.spring.web.ShiroFilterFactoryBean"*>

<property name=*"securityManager"* ref=*"securityManager"* />

<property name=*"loginUrl"* value=*"${cas.server.url}?service=${cas.project.url}${adminPath}/cas"* />

<!-- <property name="loginUrl" value="${adminPath}/login" />

<property name="successUrl" value="${adminPath}?login" />-->

<property name=*"filters"*>

<map>

<entry key=*"cas"* value-ref=*"casFilter"*/>

<entry key=*"authc"* value-ref=*"formAuthenticationFilter"*/>

<entry key=*"casLogout"* value-ref=*"casLogoutFilter"* />

</map>

</property>

<property name=*"filterChainDefinitions"*>

<ref bean=*"shiroFilterChainDefinitions"*/>

</property>

</bean>

8.4.6 商户系统的shiro权限配置xml（例：spring-context-shiro.xml）的过滤器定义中给需要进行单点统一注销的路径添加casLogout

<!-- Shiro权限过滤过滤器定义 -->

<bean name=*"shiroFilterChainDefinitions"* class=*"java.lang.String"*>

<constructor-arg>

<value>

/static/\*\* = anon

/userfiles/\*\* = anon

${adminPath}/cas = casLogout,cas

${adminPath}/login = authc

${adminPath}/logout = logout

${adminPath}/\*\* = casLogout,user

/\*\* = casLogout,user

</value>

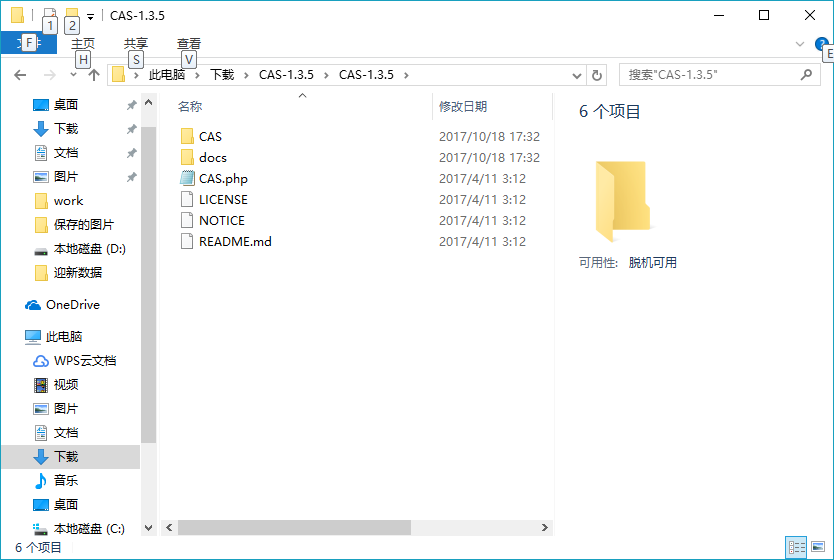
</constructor-arg>

</bean>

# CAS PHP客户端配置

### 8.1 下载 php客户端：http://developer.jasig.org/cas-clients/php/，目前最新版本为：CAS-1.3.5

### 8.2解压所下载文件，如下图所示：



### 8.3 php配置需要开启php\_curl,将CAS文件夹和CAS.php复制到工程中，修改CAS/client.php,将其中的https改为http，将docs/examples/example\_simple.php复制到工程中，修改如下：

<?php

//  
// phpCAS simple client  
//

// import phpCAS lib  
include\_once('CAS.php');

phpCAS::setDebug();

// initialize phpCAS  
phpCAS::client(CAS\_VERSION\_2\_0,'cas.nwnu.edu.cn',80,'');

// no SSL validation for the CAS server  
phpCAS::setNoCasServerValidation();

// force CAS authentication  
phpCAS::forceAuthentication();

// at this step, the user has been authenticated by the CAS server  
// and the user's login name can be read with phpCAS::getUser().

// logout if desired  
if (isset($\_REQUEST['logout'])) {

 $param=array("service"=>"[http://localhost/Phpcasclient1/example\_simple.php");//](http://local/)退出登录后返回

 phpCAS::logout($param);

}

// for this test, simply print that the authentication was successfull  
?>  
<html>  
  <head>  
    <title>phpCAS simple client</title>  
  </head>  
  <body>  
    <h1>Successfull Authentication!这是客户端1</h1>  
    <p>the user's login is <b><?php echo phpCAS::getUser(); ?></b>.</p>  
    <p>phpCAS version is <b><?php echo phpCAS::getVersion(); ?></b>.</p>  
     <p><a href="<http://192.168.18.8:8989/Casclient1/index.jsp>">去java客户端1</a></p>  
     <p><a href="?logout=">退出</a></p>  
  </body>  
</html>