# CAS单点登录系统集成

## CAS简介

### 基础

JA-SIG中央认证服务（Central Authentication Service，简称CAS）是一个开源的企业级单点登录Java服务器组件。

CAS项目于2001年在耶鲁大学兴起。最初，支持集中式单点登录。而后逐渐发展并完善了可代理的单点登录。

CAS分为cas server和cas client两部分，cas server以Java语言开发，主要用于用户身份认证。cas client有java、.net、PHP等多个版本，主要用于方便系统与CAS的集成。



### URL资源管理

在使用CAS的过程中，需要将系统的URL资源分为三种类型：

1. 不需要进行用户验证也能访问的URL资源，例如：\*.js，\*.css等。由于不需要进行用户验证也能访问，所以这些URL资源不需要CAS Client管理。
2. 需要进行用户验证才能访问的URL资源，例如：\*.do，\*.action等。由于需要进行用户验证才能访问，所以这些URL资源需要CAS Client管理。管理的原理是，在用户访问这些URL资源时，会由CAS Client提供的拦截器进行拦截。在拦截请求后，拦截器验证用户是否已经进行过登陆操作，如果没有，则通过重定向跳转到CAS Server端进行用户身份认证。
3. 不需要进行用户验证也能访问但资源的内容与用户是否登录有关的URL资源。这类URL资源虽然不需要进行用户验证也能访问，但由于其内容与用户是否登录有关，故在访问这些URL资源的时候，系统需要知道用户的登陆状态及用户信息。所以，这类资源需要交由CAS Client进行管理。用户在访问这些资源时，CAS Client需要与CAS Server进行通信，了解用户登陆状态。如果用户已经进行过登陆，则需要获取用户信息；如果用户未进行过登陆，则返回用户未登陆的结果，不要求用户进行登陆操作。

### 模式

#### 基础模式



#### 代理模式



### 实现原理

CAS Server在进行用户身份验证后，会将TGT（Ticket Granting Ticket）以Cookie的方式写入到用户的浏览器，并将ST（Service Ticket）以URL参数的形式返回给子系统。子系统获取到ST后，以HTTP协议与CAS Server通信，获取用户信息，并将其写入子系统Session。

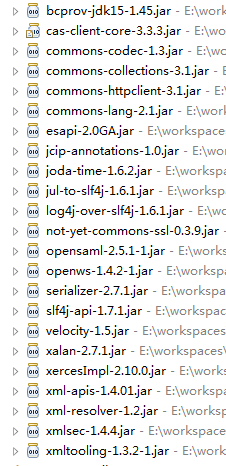
子系统在请求CAS Server进行用户身份验证的时候使用的是重定向的方式，在重定向的过程中，如果用户浏览器中有TGC（Ticket Granting Cookie，TGT的Cookie形式）。

## JAVA集成

### 引入Jar包

#### 普通web应用

将以下jar包加入到应用的classpath中：



将platform-cas-client包加入到应用的classpath中。

#### Maven应用

在应用的pom.xml文件中加入以下配置：

<dependency>

<groupId>com.gta.platform</groupId>

<artifactId>platform-cas-client</artifactId>

<version>${针对使用的具体版本进行配置}</version>

</dependency>

### 配置

#### web.xml配置

在web.xml中按照如下顺序进行配置：

<!-- 该过滤器用于进行单点登出控制 -->

<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>

<!-- 该监听器用于在本地Session失效时移除对该Session的管理 -->

<listener>

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

</listener>

<!-- 该过滤器用于判断在本系统中是否有用户验证信息，如果没有，则重定向到CAS Server端进行用户身份验证 -->

<filter>

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

<filter-class>com.gta.platform.cas.client.authentication.AuthenticationFilter</filter-class>

</filter>

<filter-mapping>

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

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

</filter-mapping>

<!-- 该拦截器为子系统获取ST后进行ST的验证并获取用户信息的拦截器 🡪

<filter>

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

<!-- 如果此处扩展了CAS Client的功能，则需要配置为扩展类的类名 🡪

<filter-class>com.gta.platform.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>

</filter>

<filter-mapping>

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

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

</filter-mapping>

<!-- 此拦截器用于对从CAS服务器获取的Assertion信息进行包装🡪

<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>

<!-- 此拦截器用于将Assertion信息放置到ThreadLocal，以方便访问🡪

<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>

#### 配置中心配置

在配置中心的平台配置中配置platform.cas.server.url，该配置项配置单点登录服务器的内网登陆地址；如果存在内外网的区别，则配置platform.cas.server.url\_outside，该配置项配置单点登录服务器外网登陆地址。具体值的格式为：http://${CAS服务器名}:${CAS服务器端口号}/${CAS服务上下文}/login

在配置中心的平台配置中配置platform.cas.server.url.prefix，该配置项配置单点登录服务器的内网地址；如果存在内外网的区别，则配置platform.cas.server.url.prefix \_outside，该配置项配置单点登录服务器外网地址。具体值的格式为：http://${CAS服务器名}:${CAS服务器端口号}/${CAS服务上下文}

在配置中心的系统节点配置中配置${系统标识}.cas.client.servername，该配置项配置本系统内网地址信息；如果存在内外网的区别，则配置${系统标识}.cas.client.servername\_outside，该配置项配置本系统外网地址信息。具体值的格式为：http://${本系统服务器名}:${本系统端口号}

#### 代码修改

* 扩展com.gta.platform.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter，获取用户详细信息，并进行其他操作。
* 开发一个过滤器，用于在用户每次与服务器交互时，刷新缓存，记录用户仍在操作系统。缓存的Key记录为platform\_userLoginActiveTime\_{userId}

#### 单点登出

单点登出时，只需要请求CAS Server的单点登出URL并传递回调URL就可以了。如果单点登录服务器有内外网的区别，则需要根据用户内外网的实际情况获取CAS服务器名。

URL格式：

http://${CAS服务器名}:${CAS服务器端口号}/${CAS服务上下文}/logout?service=${回调URL}

## .NET集成

### 下载 .NET CAS client

https://wiki.jasig.org/display/CASC/.Net+Cas+Client http://downloads.jasig.org/cas-clients/dotnet/dotnet-client-1.0.2-bin.zip

#### 普通web应用

将DotNetCasClient.dll文件引用到应用的工程文件中：

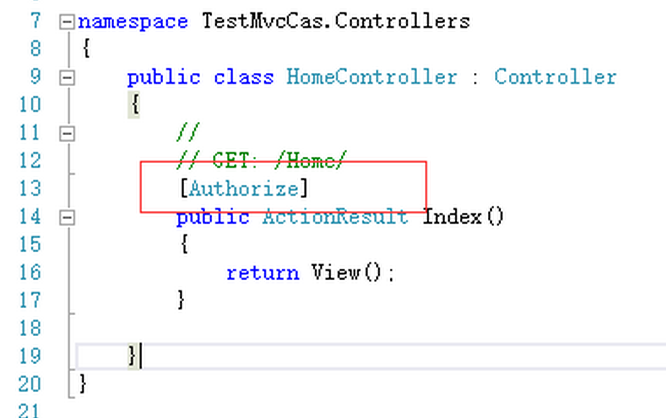


### 配置

#### web. config配置



#### 配置页面权限认证



1. 运行解决方案，会看到未登陆的用户，会自行跳转到cas登陆界面

2. 后台代码中可在Authorize中通过httpContext.User.Identity.IsAuthenticated验证是否登录成功, httpContext.User.Identity.Name可获取CAS服务端返回的用户信息。

#### 单点登出

单点登出时，只需要请求CAS Server的单点登出URL并传递回调URL就可以了。

URL格式：

http://${CAS服务器名}:${CAS服务器端口号}/${CAS服务上下文}/logout?service=${回调URL}