**CAS统一身份认证集成方案**

# 1. CAS主要特性

1、   开源的、多协议的 SSO 解决方案； Protocols ： Custom Protocol 、 CAS 、 OAuth 、 OpenID 、 RESTful API 、 SAML1.1 、 SAML2.0 等。

2、   支持多种认证机制： Active Directory 、 JAAS 、 JDBC 、 LDAP 、 X.509 Certificates 等；

3、   安全策略：使用票据（ Ticket ）来实现支持的认证协议；

4、   支持授权：可以决定哪些服务可以请求和验证服务票据（ Service Ticket ）；

5、   提供高可用性：通过把认证过的状态数据存储在 TicketRegistry 组件中，这些组件有很多支持分布式环境的实现，如： BerkleyDB 、 Default 、 EhcacheTicketRegistry 、 JDBCTicketRegistry 、 JBOSS TreeCache 、 JpaTicketRegistry 、 MemcacheTicketRegistry 等；

6、   支持多种客户端： Java 、 .Net 、 PHP 、 Perl 、 Apache, uPortal 等。

# 2. CAS基本原理

## 2.1  结构体系

从结构体系看， CAS 包括两部分： CAS Server 和 CAS Client 。

### 2.1.1      CAS Server

CAS Server 负责完成对用户的认证工作 , 需要独立部署 , CAS Server 会处理用户名 / 密码等凭证 (Credentials) 。

### 2.1.2      CAS Client

负责处理对客户端受保护资源的访问请求，需要对请求方进行身份认证时，重定向到 CAS Server 进行认证。（原则上，客户端应用不再接受任何的用户名密码等 Credentials ）。

CAS Client 与受保护的客户端应用部署在一起，以 Filter 方式保护受保护的资源。

## 2.2  CAS 原理和协议

1. 访问服务： SSO 客户端发送请求访问应用系统提供的服务资源。

2. 定向认证： SSO 客户端会重定向用户请求到 SSO 服务器。

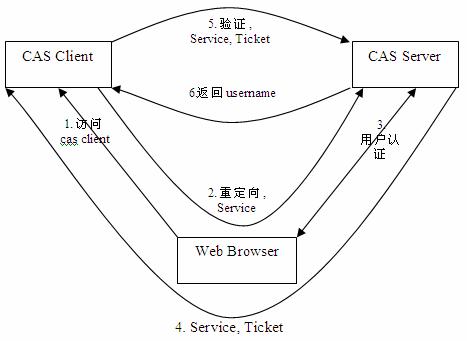
3. 用户认证：用户身份认证。

4. 发放票据： SSO 服务器会产生一个随机的 Service Ticket 。

5. 验证票据： SSO 服务器验证票据 Service Ticket 的合法性，验证通过后，允许客户端访问服务。

6. 传输用户信息： SSO 服务器验证票据通过后，传输用户认证结果信息给客户端。

下面是 CAS 最基本的协议过程：



# 3. CAS单点登录客户端部署方案（java）

## 3.1依赖包引用

CAS客户端版本3.2.1。

### 3.1.1 Maven形式（推荐）

在自身项目pom.xml文件中，加入以下代码

<dependency>

<groupId>org.jasig.cas.client</groupId>

<artifactId>cas-client-core</artifactId>

<version>3.2.1</version>

</dependency>

### 3.1.2 引用JAR包形式

将cas-client-core-3.2.1.jar放入自身项目中。



## 3.2 CAS客户端配置代码

在项目web.xml中添加以下代码。

配置项：

CAS Server 项目地址 例：http://localhost:8080/cas

CAS Server 登录地址 例：http://localhost:8080/cas/login

CAS Client (自身项目) 地址 例：http://localhost:9099

<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>

<init-param>

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

<param-value>http://localhost:8080/cas</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CAS Single Sign Out Filter</filter-name>

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

</filter-mapping>

<filter>

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

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

<!-- CAS验证服务器地址-->

<init-param>

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

<param-value> http://localhost:8080/cas/login </param-value>

</init-param>

<init-param>

<param-name>renew</param-name>

<param-value>false</param-value>

</init-param>

<init-param>

<param-name>gateway</param-name>

<param-value>false</param-value>

</init-param>

<!--客户端应用服务器地址-->

<init-param>

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

<param-value>http://localhost:9099</param-value>

</init-param>

</filter>

<!--负责Ticket校验-->

<filter>

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

<filter-class> org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter </filter-class>

<!-- CAS验证服务器地址(上下文) -->

<init-param>

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

<param-value>http://localhost:8080/cas</param-value>

</init-param>

<!--客户端应用服务器地址-->

<init-param>

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

<param-value>http://localhost:9099</param-value>

</init-param>

<init-param>

<param-name>useSession</param-name>

<param-value>true</param-value>

</init-param>

<init-param>

<param-name>redirectAfterValidation</param-name>

<param-value>true</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>

<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 Authentication 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 HttpServletRequest Wrapper Filter</filter-name>

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

</filter-mapping>

<filter-mapping>

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

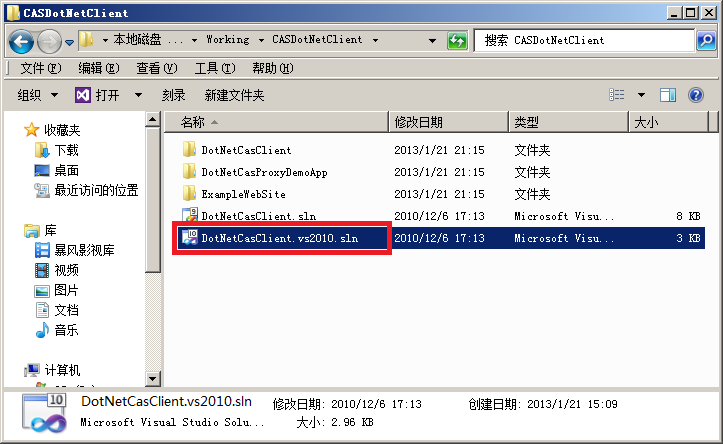
<url-pattern>/\*</url-pattern>

</filter-mapping>

# 4. CAS单点登录客户端部署方案（.net）

## 4.1 下载依赖包。

****压缩文件。

[](http://images.cnitblog.com/blog/1404/201301/22102811-a076f0015b5c4513b3a897c71326a07a.png)

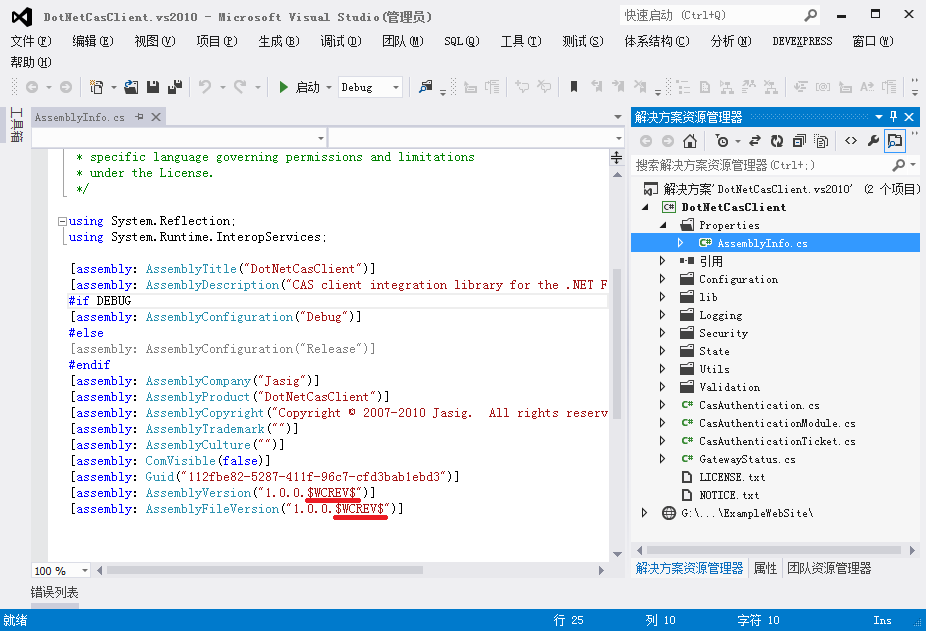
## 4.2 配置 CAS DotNetClient

以管理员身份启动Visual Studio（目的为了随后可以直接将网站发布到IIS），打开“DotNetCasClient.vs2010.sln”解决方案。

（1）项目“DotNetCasProxyDemoApp”暂时用不到，从解决方案中移除。

（2）将“DotNetCasClient”项目中“Properties”文件夹下的“AssemblyInfo.cs”删除，将“AssemblyInfo.cs.tmpl”重命名为“AssemblyInfo.cs”。

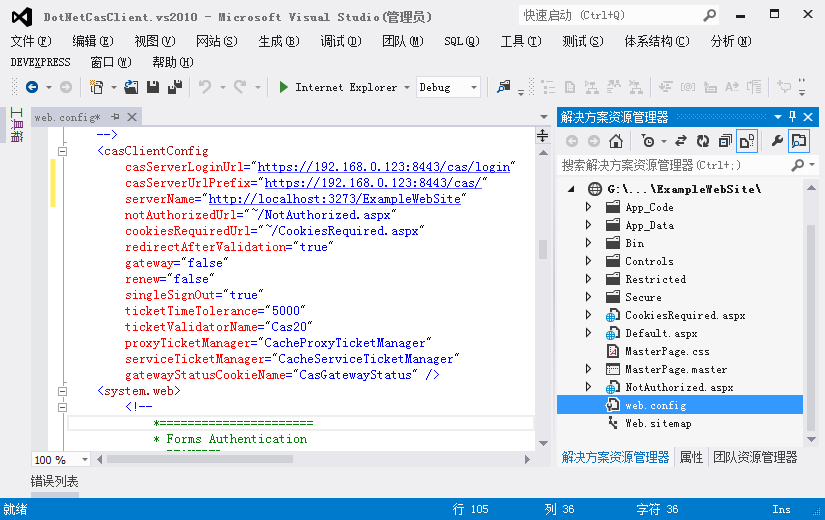
（3）打开将“DotNetCasClient”项目中“Properties”文件夹下的“AssemblyInfo.cs”，将所有“$WCREV$”替换成“0”（或其它表示版本的数字）。

[](http://images.cnitblog.com/blog/1404/201301/22102813-0c6b587a9dcc45aba31f44dcb69b2a26.png)

（4）将“ExampleWebSite”项目设置为启动项。

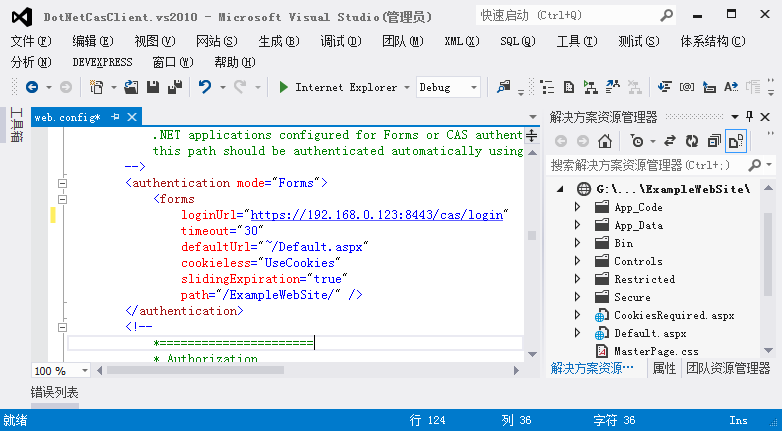
（5）将“ExampleWebSite”项目根文件夹下的“web.config.sample”重命名为“web.config”

（6）打开“web.config”文件，找到“casClientConfig”节点，将“**casServerLoginUrl**”属性设置为“https://192.168.0.123:8443/cas/login”，将“**casServerUrlPrefix**”属性设置为“https://192.168.0.123:8443/cas/”，将“**serverName**”属性设置为“http://localhost:3273/ExampleWebSite”。

[](http://images.cnitblog.com/blog/1404/201301/22102816-0bc3172b827b44cc8922d0f4dd23250e.png)

说明：192.168.0.123是前面我们配置好的CAS服务器IP地 址；“serverName”属性中3273为运行该项目时IISExpress自动分配的端口号，如果项目发布到客户端IIS上，建议将 “serverName”属性更改为“http://192.168.0.153/ExampleWebSite”，其中192.168.0.153为客 户端IP地址。注意：“serverName”属性中网络地址最后不要加“/”。

（7）从“web.config”文件中找到“authentication”节点，将“**loginUrl**”属性设置为“https://192.168.0.123:8443/cas/login”，将“**path**”属性设置为“/ExampleWebSite/”。

[](http://images.cnitblog.com/blog/1404/201301/22102818-5828dd672774400486588d8262594e97.png)

1. 保存全部修改，重新编译解决方案。

## 4.3参考网址

<http://www.cnblogs.com/zhenyulu/archive/2013/01/22/2870936.html>

官网下载与使用说明：<https://github.com/Jasig/dotnet-cas-client>

# 5. CAS单点登录客户端部署完毕验证

## 5.1 验证统一登录

1. 访问自身项目，查看项目是否跳转到CAS Service登录处。若没有跳转，则表示配置错误。
2. 跳转成功后，在此登录处进行登录，若登录成功后不能跳转到自身访问的项目中，则表示配置错误。

## 5.2 验证统一退出

登录成功之后，在CAS Service项目中退出登录，此时刷新自身项目，查看是否跳转到CAS Service 登录处。若能，则表示配置成功，反之不成功。

# 获取当前 CAS登录者信息

## 步骤1

访问该CAS网址。

格式为：cas登录地址?service=自身业务地址

例：http://localhost:8080/dmm\_cas/login?service=http://localhost:9099/dmm\_logistics/common/ssologion

情况A：若CAS未登录，则进入CAS登录页面，进行登录。若登录成功，则转换成“情况B”。

情况B：若CAS已登录，则返回该URL。并附加ticket值。

返回的URL格式为：自身业务地址?ticket=ticket值

例：http://localhost:9099/dmm\_logistics/common/ssologion?ticket=ST-76-AdJxGbBYPfjrYCpFGy5Z-cas01.example.org

## 步骤2

获取“步骤1”中返回的ticket值。

并访URL：cas验证地址?ticket=ticket值&service=自身业务地址

例：http://localhost:8080/dmm\_cas/serviceValidate?ticket=ST-78-3b3dtJWVcp9VdPqlf9gX-cas01.example.org&service=http://localhost:9099/dmm\_logistics/common/ssologion

获得返回的XML：

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>

<cas:authenticationSuccess>

<cas:user>用户名（学生为学号，教师为工号）</cas:user>

<cas:attributes>

<cas:school>所属学校代码</cas:school>

<cas:idno>身份证号</cas:idno>

</cas:attributes>

</cas:authenticationSuccess>

</cas:serviceResponse>