一、不落俗套的开始

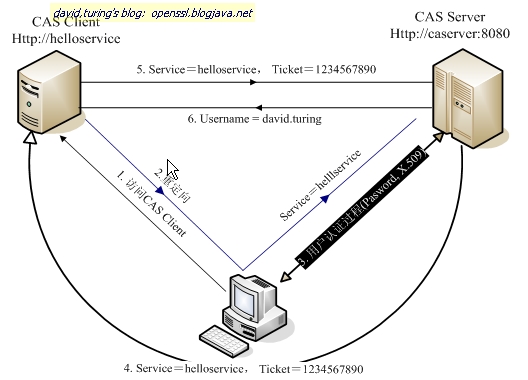
1、背景介绍

单点登录：Single Sign On,简称SSO，SSO使得在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。

CAS框架：CAS（Central Authentication Service）是实现SSO单点登录的框架。

2、盗一张学习CAS绝大多都看过的图以及执行部分分析

注：已分不清原创，此处就不给出地址了。



从结构上看，CAS包含两个部分：CAS Server 和CAS Client需要独立部署，主要负责对用户的认证工作；CAS

Client负责处理对客户端受保护资源的访问请求，需要登录时，重定向到CAS Server.图1是CAS最基本的协议过程：

CAS Client 与受保护的客户端应用部署在一起，以Filter方式保护 Web 应用的受保护资源，过滤从客户端过来的每一个 Web

请求，同时， CAS Client会分析HTTP 请求中是否包请求 Service Ticket( 上图中的 Ticket)

，如果没有，则说明该用户是没有经过认证的，于是，CAS Client会重定向用户请求到CAS Server（ Step 2 ）。 Step

3是用户认证过程，如果用户提供了正确的Credentials， CAS Server 会产生一个随机的 Service Ticket

，然后，缓存该 Ticket ，并且重定向用户到CAS Client（附带刚才产生的Service Ticket）， Service

Ticket 是不可以伪造的，最后， Step 5 和 Step6是 CAS Client 和 CAS

Server之间完成了一个对用户的身份核实，用Ticket查到 Username ，因为 Ticket是 CAS Server

产生的，因此，所以 CAS Server 的判断是毋庸置疑的。

该协议完成了一个很简单的任务，所有与CAS的交互均采用SSL协议，确保ST和TGC的安全性。协议工作过程会有2此重定向过程，但是CAS

Client与CAS Server之间进行ticket验证的过程对于用户是透明的。

总结一下，如下：

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。

定向认证： SSO 客户端会重定向用户请求到 SSO 服务器。

用户认证：用户身份认证。

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

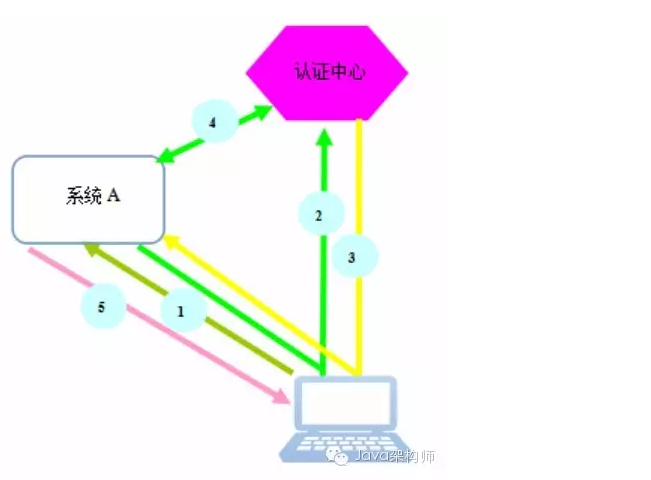
验证票据： SSO 服务器验证票据 Service Ticket 的合法性，验证通过后，允许客户端访问服务。

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。

二、在云里雾里进一步搜索、探究

先给出此部分内容出处： 《SSO CAS单点系列》之 实现一个SSO认证服务器是这样的，以下标红部分为自己的疑问。

1. 登录信息传递



用户首次登录时流程如下：

1)、用户浏览器访问系统A需登录受限资源，此时进行登录检查，发现未登录，然后进行获取票据操作，发现没有票据。

2)、系统A发现该请求需要登录，将请求重定向到认证中心，获取全局票据操作，没有，进行登录。

3)、认证中心呈现登录页面，用户登录，登录成功后，认证中心重定向请求到系统A，并附上认证通过令牌，此时认证中心同时生成了全局票据。

4)、此时再次进行登录检查，发现未登录，然后再次获取票据操作，此时可以获得票据(令牌)，系统A与认证中心通信，验证令牌有效,证明用户已登录。

5)、系统A将受限资源返给用户。

已登录用户首次访问应用群中系统B时：

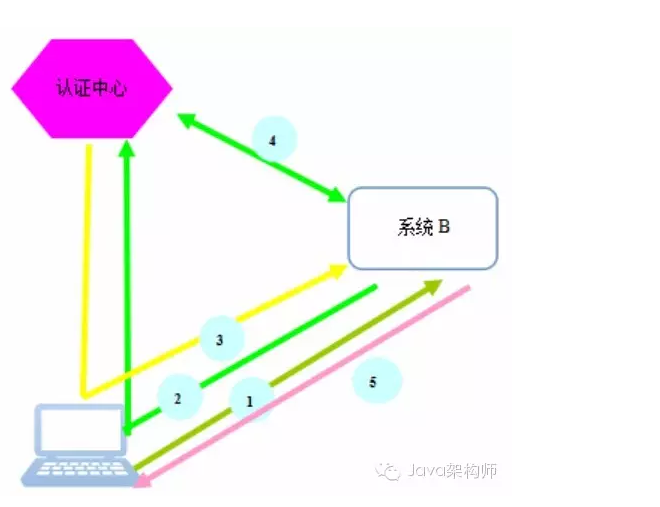
1)、浏览器访问另一应用B需登录受限资源，此时进行登录检查，发现未登录，然后进行获取票据操作，发现没有票据。

2)、系统B发现该请求需要登录，将请求重定向到认证中心，获取全局票据操作，获取全局票据，可以获得，认证中心发现已经登录。

3)、认证中心发放临时票据(令牌)，并携带该令牌重定向到系统B。

4)、此时再次进行登录检查，发现未登录，然后再次获取票据操作，此时可以获得票据(令牌)，系统B与认证中心通信，验证令牌有效,证明用户已登录。

1. 、系统B将受限资源返回给客户端。



2、登录状态判断

用户到认证中心登录后，用户和认证中心之间建立起了会话，我们把这个会话称为全局会话。当用户后续访问系统应用时，我们不可能每次应用请求都到认证中心去判定是否登录，这样效率非常低下，这也是单Web应用不需要考虑的。

我们可以在系统应用和用户浏览器之间建立起局部会话，局部会话保持了客户端与该系统应用的登录状态，局部会话依附于全局会话存在，全局会话消失，局部会话必须消失。

用户访问应用时，首先判断局部会话是否存在，如存在，即认为是登录状态，无需再到认证中心去判断。如不存在，就重定向到认证中心判断全局会话是否存在，如存在，按1提到的方式通知该应用，该应用与客户端就建立起它们之间局部会话，下次请求该应用，就不去认证中心验证了。

3、登出

用户在一个系统登出了，访问其它子系统，也应该是登出状态。要想做到这一点，应用除结束本地局部会话外，还应该通知认证中心该用户登出。

认证中心接到登出通知，即可结束全局会话，同时需要通知所有已建立局部会话的子系统，将它们的局部会话销毁。这样，用户访问其它应用时，都显示已登出状态。

整个登出流程如下：

1)、客户端向应用A发送登出Logout请求。

2)、应用A取消本地会话，同时通知认证中心，用户已登出。

3)、应用A返回客户端登出请求。

4)、认证中心通知所有用户登录访问的应用，用户已登出。

三、拨开云雾见青天

1、对上面所有标红疑问一一解释

1)、问：系统A是如何发现该请求需要登录重定向到认证中心的？

答：用户通过浏览器地址栏访问系统A，系统A(也可以称为CAS客户端)去Cookie中拿JSESSION，即在Cookie中维护的当前回话session的id，如果拿到了，说明用户已经登录，如果未拿到，说明用户未登录。

2)、问：系统A重定向到认证中心，发送了什么信息或者地址变成了什么？

答：假如系统A的地址为http://a:8080/，CAS认证中心的服务地址为http://cas.server:8080/，那么重点向前后地址变化为：http://a:8080/————>ttp://cas.server:8080/?service=http://a:8080/，由此可知，重点向到认证中心，认证中心拿到了当前访问客户端的地址。

3)、问：登录成功后，认证中心重定向请求到系统A，认证通过令牌是如何附加发送给系统A的？

答：重定向之后的地址栏变成：http://a:8080/?ticket=ST-XXXX-XXX，将票据以ticket为参数名的方式通过地址栏发送给系统A

4)、问：系统A验证令牌，怎样操作证明用户登录的？

答：系统A通过地址栏获取ticket的参数值ST票据，然后从后台将ST发送给CAS server认证中心验证，验证ST有效后，CAS server返回当前用户登录的相关信息，系统A接收到返回的用户信息，并为该用户创建session会话，会话id由cookie维护，来证明其已登录。

5)、问：登录B系统，认证中心是如何判断用户已经登录的？

答：在系统A登录成功后，用户和认证中心之间建立起了全局会话，这个全局会话就是TGT(Ticket Granting Ticket)，TGT位于CAS服务器端，TGT并没有放在Session中，也就是说，CAS全局会话的实现并没有直接使用Session机制，而是利用了Cookie自己实现的，这个Cookie叫做TGC(Ticket Granting Cookie)，它存放了TGT的id,保存在用户浏览器上。

相关内容分析可以查看：《SSO CAS单点系列》之 实操！轻松玩转SSO CAS就这么简单(相识篇)

用户发送登录系统B的请求，首先会去Cookie中拿JSESSION，因为系统B并未登录过，session会话还未创建，JSESSION的值是拿不到的，然后将请求重定向到CAS认证中心，CAS认证中心先去用户浏览器中拿TGC的值，也就是全局会话id，如果存在则代表用户在认证中心已经登录，附带上认证令牌重定向到系统B。

上面登录状态判断也是这个逻辑。

6)、问：登出的过程，各个系统对当前用户都做了什么？

答：认证中心清除当前用户的全局会话TGT，同时清掉cookie中TGT的id：TGC；

然后是各个客户端系统，比如系统A、系统B，清除局部会话session，同时清掉cookie中session会话id：jsession

1. 对系统A登录流程图添加注释，查看

