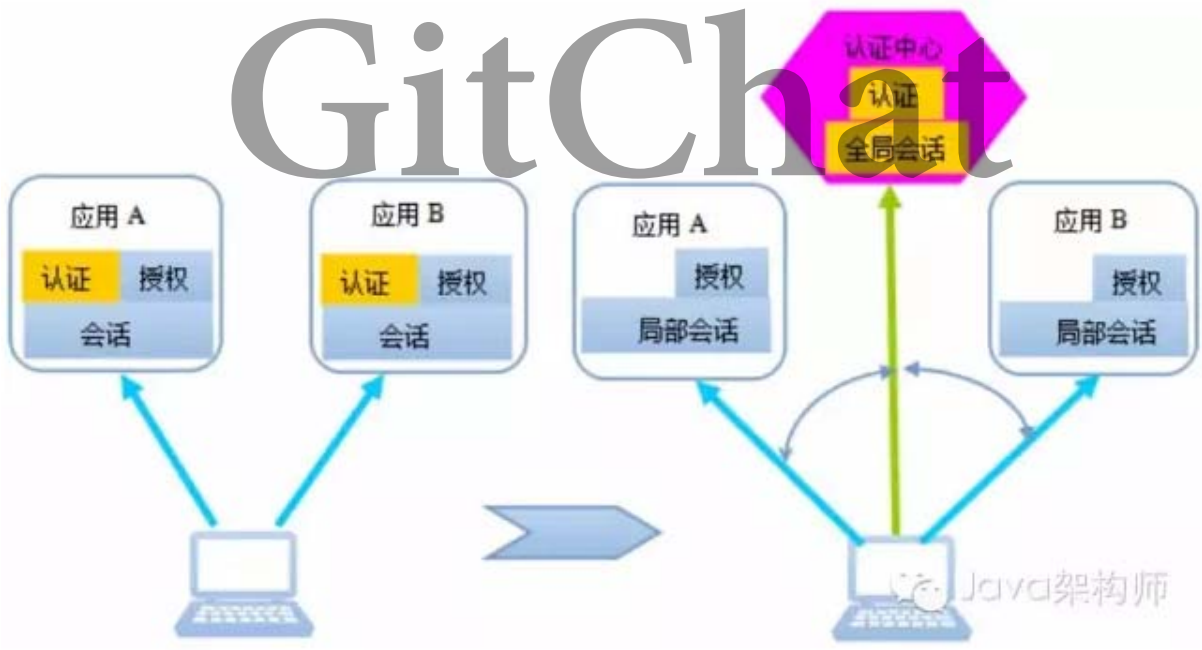
电商系统中的单点登录

SSO 系统产生背景

在单 Web 应用系统中，我们只需要考虑客户端与该服务器的单一会话即可，实现起来也比较容易，只需要登录成功后写入 cookie，所以每次请求该 Web 应用都会携带 cookie，服务器端只考虑验证这个 cookie 是否有效即可判断是否登录。随着业务增长，出现了系统协同工作，那么每个应用只维持自己的会话会出现如下问题。

1. 每个系统都要维护一套认证逻辑，造成冗余；
2. 跨系统之后，认证信息失效，需要各个系统之间兼容。

那么，就需要将公共模块抽象出来，组成一个通用的认证系统，承担起所有业务系统的登录认证功能，也就是我们所说的 SSO 系统，如下图所示。

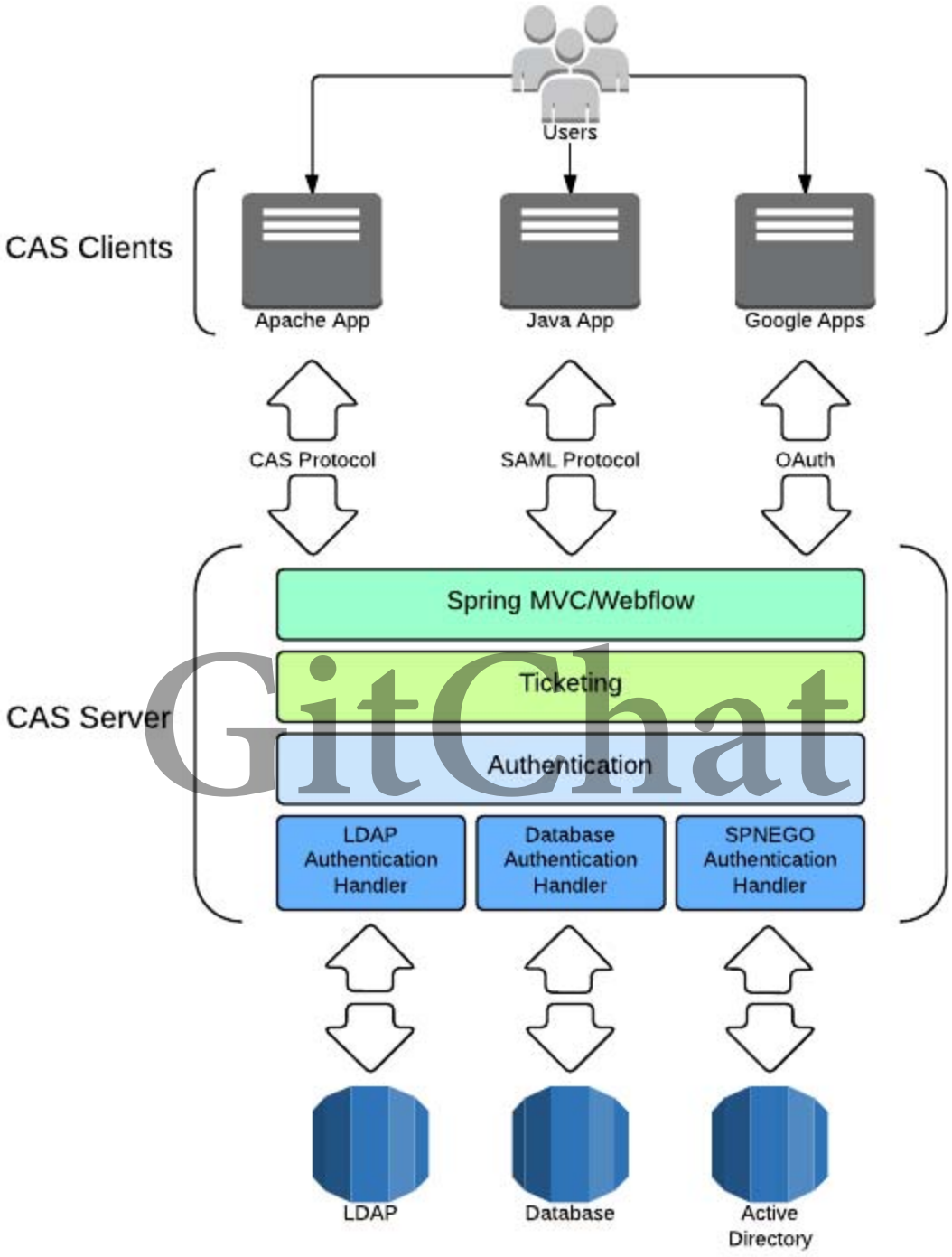


一般的 SSO 系统模型

如上图所示，抽象出认证系统之后，单点登录系统需要完成两个主要工作，全局会话的保持和局部会话的保持。客户端与业务系统之间是局部会话，与 SSO 系统之间是全局会话。我们会将 SSO 系统分为两部分，SSO 服务端和 SSO 客户端，SSO 服务端则是我们的 SSO 认证系统，SSO 客户端将集成进入业务系统，负责局部会话的新增、删除、验证。

开源框架 CAS 的原理

CAS 太过经典，所以基本上所有的 SSO 系统，所会对 CAS 有所借鉴。



上图展示了 CAS 的整体架构，同样分为客户端和服务端。客户端支持多种服务器应用，同时也支持多语言，包括GO、Python、PHP、Java、.NET，可以看到对市面上的主要语言都有支持。

我主要是从事 Java 开发，本文只会着重说一下 Java 的实现，如果读者有兴趣，可以去 GitHub 查看其他语言的实现。

可以从图中看出服务端的技术实现，首先是 Spring MVC + Spring Web Flow，Web Flow主要用于将组件串行执行，往下是票据组件、认证组件、认证组件支持的存储容器，可以

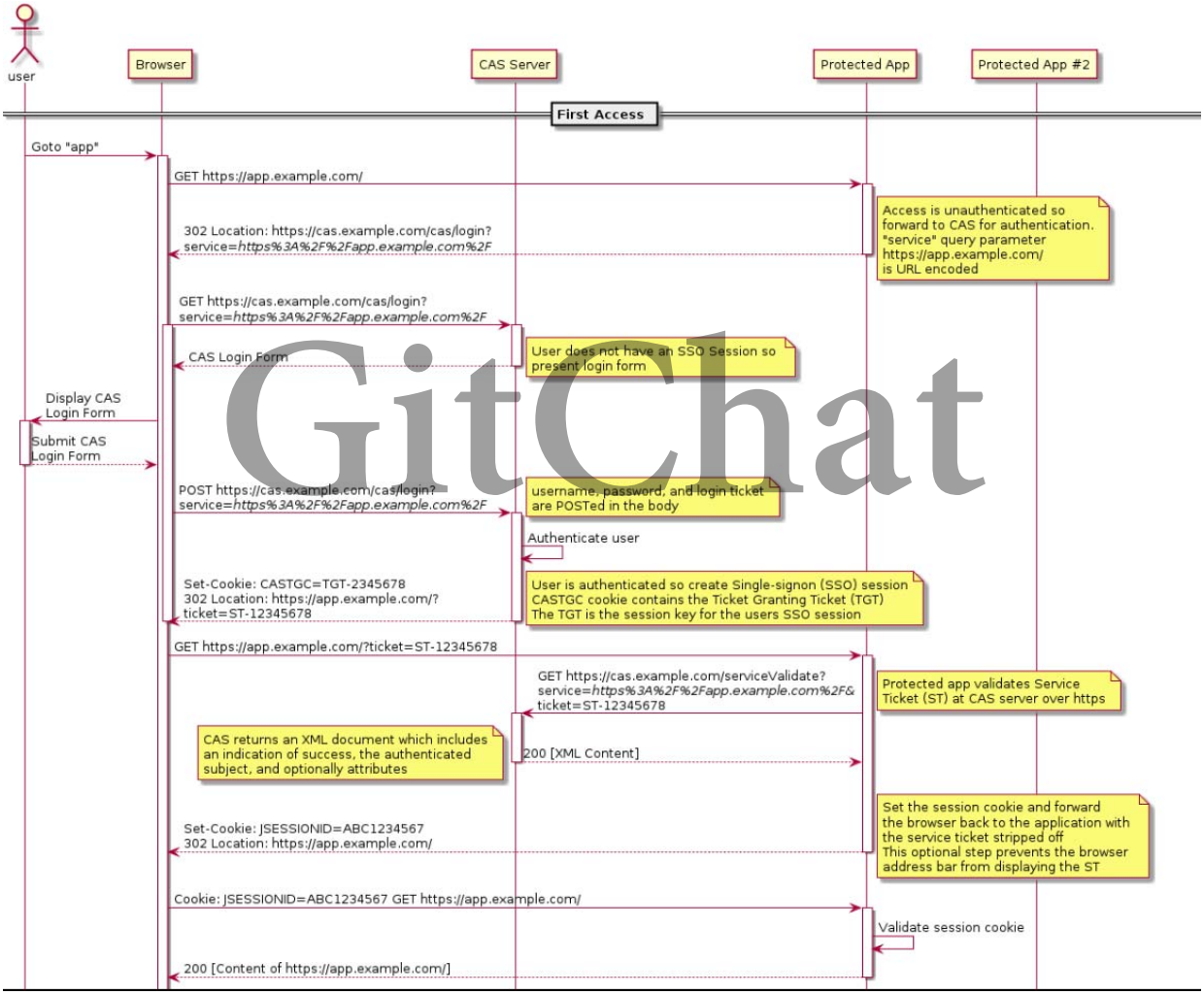
是 LDAP、数据库、活动目录，我们基本上的认证思路就是关系数据库 + Redis 或 Memcached 来配合实现。

下面我将简要说一下整体的认证流程。

上图主要通过三种情况，详细描述了 CAS 的认证过程。

1.首次访问受限资源时

首次访问时，重定向到 SSO 服务端登录页，返回登录表单给浏览器，用户提交用户名密码，SSO 服务端验证，成功后携带 ticket 重定向会 SSO 客户端，客户端与 SSO 验证 ticket 有效性，返回验证信息，SSO 客户端写局部会话 cookie，重定向回原地址，业务系统返回资源。如下图所示



客户端关键代码如下。

```
final Assertion assertion = session != null ? (Assertion)
session.getAttribute(CONST_CAS_ASSERTION) : null;

if (assertion != null) {
    filterChain.doFilter(request, response);
    return;
}
```

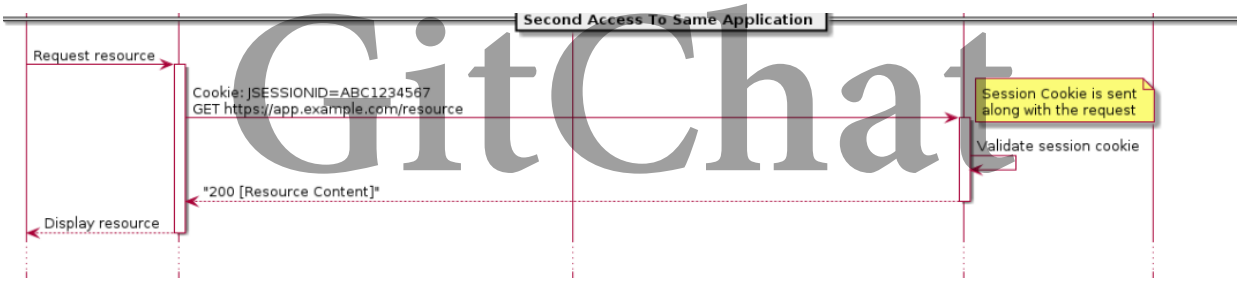
```
final String serviceUrl = constructServiceUrl(request, response);
final String ticket =
CommonUtils.safeGetParameter(request,getArtifactParameterName());
final boolean wasGatewayed =
this.gatewayStorage.hasGatewayedAlready(request, serviceUrl);

if (CommonUtils.isNotBlank(ticket) || wasGatewayed) {
    filterChain.doFilter(request, response);
    return;
}
```

如果登录，直接跳转，即执行：

```
response.sendRedirect(urlToRedirectTo);
```

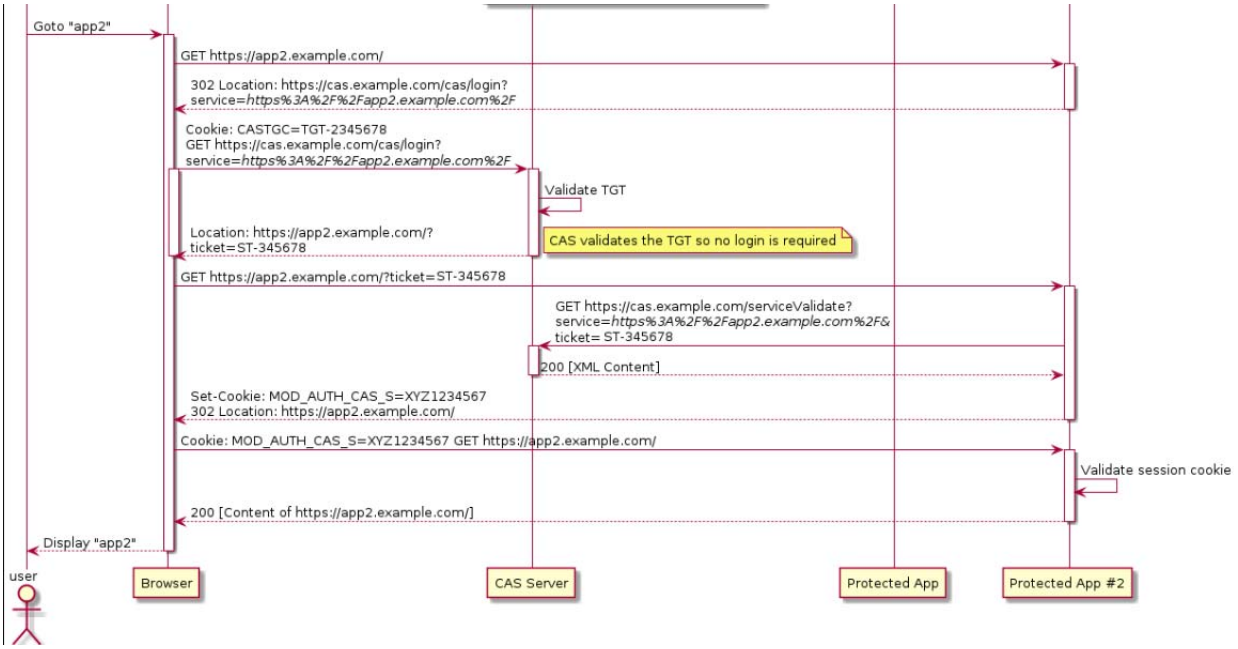
2.第二次访问该系统。

第二次访问该系统，会在该域名下存在上一步写的 cookie，请求该系统时携带 cookie，所有 filter 不会拦截该请求，直接返回资源。如下图所示。



3.首次访问其他系统。

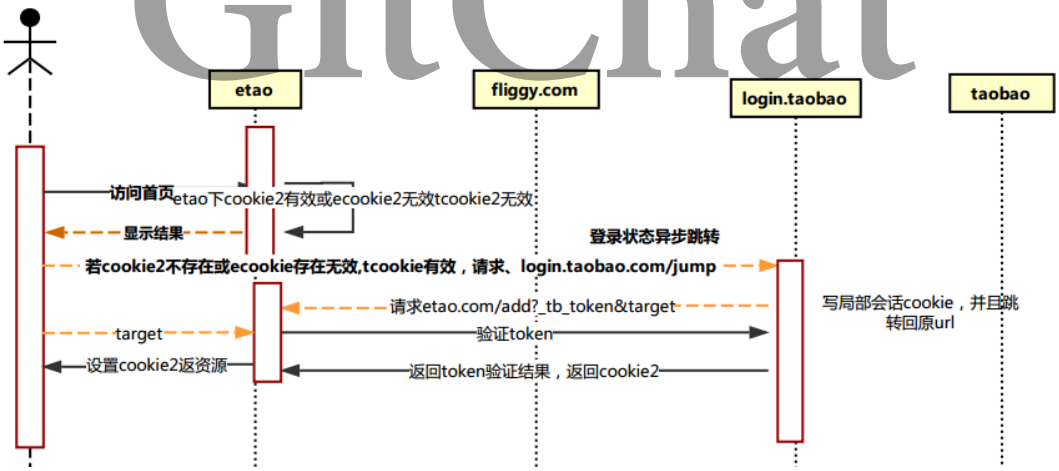
在该系统域名下是不存在局部会话的，所以重定向到 SSO 服务端，SSO 服务端会发现此客户端已经登录，所有生成 ticket，客户端与 SSO 验证 ticket 有效性，返回验证信息，SSO 客户端写局部会话 cookie，重定向回原地址，业务系统返回资源。



分享淘宝 SSO 系统架构设计以及实现

淘宝的 SSO 系统是比较有新意的，除了校验登录状态模块，还加入了同步登录状态模块，这样就让电商项目在 SSO 中变得很灵活了。

第一步，同步登录状态。

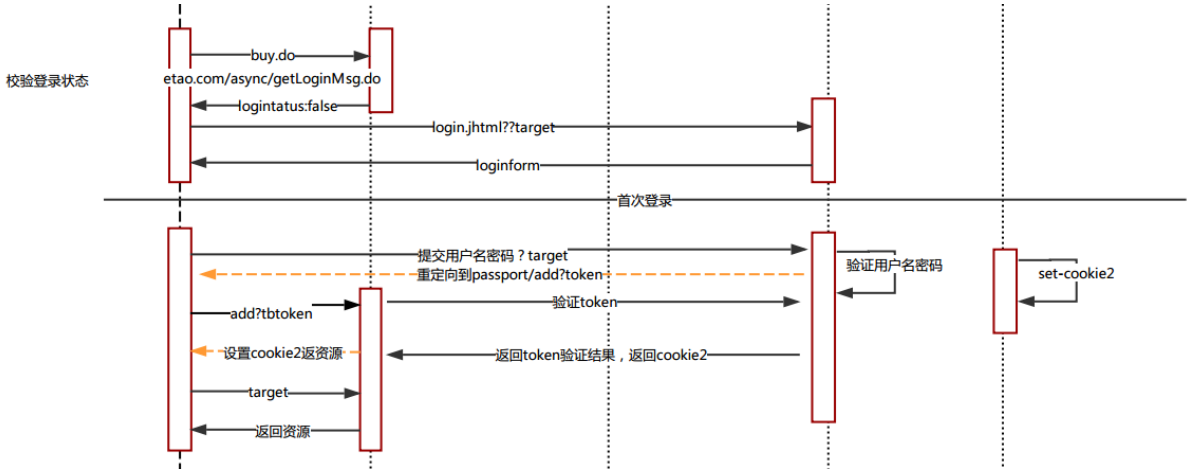


在静态页中，会异步请求后台数据，这时候会被同步登录状态的 SSO 客户端 filter 拦截。如果需要同步登录状态，filter 将重定向到 login.taobao.jump 接口，这个接口无论用户是否登录，都会重定向回 SSO 客户端的接口，在以下两个条件下发生跳转：

1. 局部会话的 cookie 不存在；
2. cookie 存在但是无效，全局会话有效。

所以，只会发生一次跳转，不会重复。中间的跳转除了携带 token 参数还会携带来源地址 rederecturl。

第二步，校验登录状态。



当用户请求到需要登录的数据资源时会被校验登录状态的 filter 拦截，出现以下两种情况：

1. 同步跳转请求，如果没有登录，直接重定向到登录页；
2. 异步 Ajax 请求，会直接返回登录状态和 rederecturl、loginurl，由 JavaScript 控制跳转到登录地址。

第三步，验证票据。

如果 SSO 服务端登录成功，会携带 token 请求回 SSO 客户端，客户端验证 token 的 filter 拦截请求，与 SSO 服务端验证 token 有效性。如果通过，则返回用户基本信息、cookie 值等，所有的 cookie 值都是由 SSO 服务端发出的。