单点登录设计

1 关键概念

* Central Autherization Server，CAS认证中心服务。
* CAS Client，CAS 客户端，也就是使用CAS登录的应用，本文档称为APP 1、APP 2 等。

2 单点登录

假设CAS Server的域名为cas.com，CAS客户端为app1.com和app2.com，则单点登录的流程如下：

2.1 首次访问APP1应用

1. 首次访问APP 1，对应域名下没有Token信息，应用侧会判定用户未登录，返回错误。

2. APP 1 前端收到返回错误，跳转到认证中心主页，同时带上APP 1的跳转地址。如下：

```
GET https://cas.com?app_id=<app id>
```

3. 首次访问认证中心，对应域名下没有Token信息，认证中心侧判定用户未登录，返回错误，注意response中需要带上跳转地址service。

```
{
”service”:”https://app1.com/cas/validate”,
}
```

4. 认证中心前端跳转到用户登录页，由用户填写登录（用户名/密码）信息并提交。
5. 发送登录请求到认证中心服务，注意需要带上跳转参数service。如下：

```
POST https://cas.com/cas/login?app_id=<app id>
```

6. 认证服务收到登录请求，校验app_id、用户名/密码，用户是否有app_id的权限，通过后生成双Token（access_token+refresh_token），以及ST（Service Ticket），返回结果。注意response中需要带上双Token、ST、跳转地址service。

```
{
    ”access_token”:”<access_token>”,
    ”refresh_token”:”<refresh_token>”,
    ”st”:”<ST>”,
}
```

7. 认证中心前端将双Token存储在cas.com域名对应的localStorage中。发起APP校验接口请求，请求地址为response中的service。如下：

```
GET https://app1.com/cas/validate?app_id=<app id>&st=<ST>
```

8. APP 1服务接收到validate请求，向认证中心发起校验ST的请求，如下：

```
GET https://cas.com/cas/validate?app_id=<app id>&st=<ST>
```

9. 认证中心校验ST，校验成功后，需要将缓存的ST参数删除。返回校验结果和用户uid。

```
{
    ”uid”:<uid>,
}
```

10. APP1服务确认st校验通过，生成双Token，关联uid，并返回结果。

```
{
    ”access_token”:”<access_token>”,
    ”refres_token”:”<refresh_token>”,
}
```

11. APP1 前端将双Token存储在app1.com域名对应的LocalStorage中。
12. 浏览器跳转到上述APP1的主页地址。

以上流程完成后，APP1就完成了用户登录。

2.2 APP1应用登录后请求

在完成了登录流程后，APP1在后续的业务请求中，会在header中带上Authorazation，对应值即为access_token。

由APP1服务进行access_token的校验。

2.3 首次访问APP2应用

基于 2.1 完成的前提，以下为APP2首次访问的流程：

1. 首次访问APP 2，对应域名下没有Token信息，应用侧会判定用户未登录，返回错误。
2. APP 2 前端收到返回错误，跳转到认证中心主页。如下：

```
GET https://cas.com?app_id=<app id>
```

3. 非首次访问认证中心，对应域名下已有之前完成登录生成的Token信息，认证中心侧判定用户已登录，生成ST（Service Ticket），返回结果。

注意response中需要带上ST、跳转地址service。

```
{
    ”st”:”<ST>”,
}
```

此处需要注意，ST的生成是有频次控制的，防止在ST校验失败前端跳转处理将逻辑拉入一个死循环中。

4. 浏览器跳转到上述APP 2的校验接口，APP 2服务接收到validate请求，向认证中心发起校验ST的请求，如下：

```
GET https://cas.com/cas/validate?app_id=<app id>&st=<ST>
```

5. 认证中心校验ST，校验成功后，需要将缓存的ST参数删除。返回校验结果和uid。

```
{
    ”uid”:<uid>,
}
```

6. APP 2 服务确认st校验通过，生成双Token，并返回结果。

```
{
    ”access_token”:”<access_token>”,
    ”refres_token”:”<refresh_token>”,
}
```

7. **APP 2 前端将双Token存储在app2.com域名对应的LocalStorage中。**

3 单点登出
======

只要有一个CAS Client应用执行了logout操作，则所有应用都判定为已登出，并且在认证中心看到的用户也为登出状态。

实现方案为在原有的jwt校验机制基础上，服务端侧增加一个middleware，校验Token的有效期：如果用户在CAS Client应用执行了登出操作，则在应用服务记录用户的Token过期时间为当前时间；如果在认证中心侧执行了强制用户下线操作，则在认证中心记录用户的Token过期时间为当前时间；

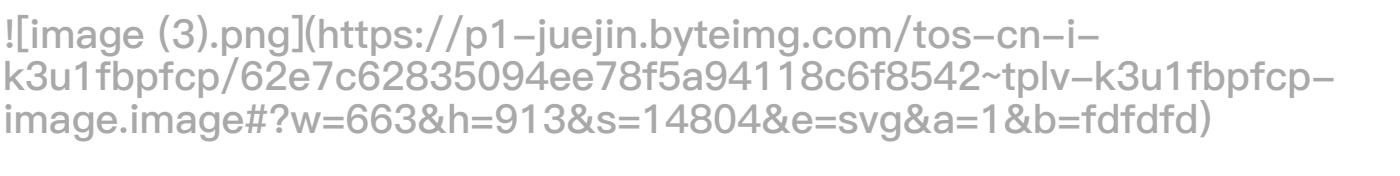
二者的底层实现一致，都是以key-value的方式更新Token的过期时间；

过期时间缓存的生命周期与refresh_token一致。（如果认证中心Token和应用Token的refresh_token不一致，取大的值。）

4 接口请求

APP与Cas Server的接口请求，使用AppId+AppSecret方式进行签名/验签，防止中间人伪造或篡改请求。

4.1 签名生成与验证



5 总结

大部分企业内部会存在多套业务系统，而多套系统的用户管理需要统一协调处理，否则就会出现繁杂的用户信息管理工作。本文力求剖析单点登录的实现，希望对大家有帮助～

原文链接: https://juejin.cn/post/7380296428143116288