背景：

现有laravel实现sso的扩展有2个

- laravel-sso(https://github.com/cblink/laravel-sso)，最近更新时间为2年前，不清楚新版的兼容性，无法实现同步登出
- laravel-cas(https://github.com/leo108/laravel_cas_server)，最近更新时间也是2年前，同样没有实现同步登出（有issue提示问题，作者也表示了不支持），只支持5.5以下laravel，5.5以上安装会报错

基于现有情况，下面是根据cas原理的laravel实现：

主体：

应用A：单独应用A，有自己的用户系统，当启用SSO服务时，密码字段不生效（自身非SSO服务），前后端分离项目，这里指后端。

SSO服务：可能与任意一个系统相结合，但不影响流程，用户系统仅记录关键用户信息（用户名，密码，邮箱等），用户系统可能是A或B中其中一个。

应用B：同应用A

流程：

应用A登录，此时sso未登录

1. 应用A前端检测token不存在，向应用A后端请求token。
2. 应用A后端判断token不存在，引导用户跳转至登录界面/登录接口**(必须是跳转，不是异步访问)**。
    - 此处引导的意思是指将401返回到前端，前端访问token用的异步，后端不能执行302跳转。
    - 启动sso后，返回的redirect_to字段为sso的登录接口，否则，由前端控制自身跳转。
3. 跳转至sso登录，sso判断当前无用户，引导至登录界面
    - 此登录界面是sso系统的统一登录界面，必须由sso控制。需要传递**当前应用的应用标识。**
    - 若未启用sso的应用，应当拥有自己的登录界面。
4. sso登录，生成用户session，将session_id存储在用户cookie中
    - 其实就是最传统的cookie-session登录验证
    - 建议不做前后端分离，直接用后端会方便许多。
5. sso登录验证，验证成功后生成一次性ticket，按照配置好的redirect返回到应用A后端，并附加ticket参数。
    - ticket生成规则 md5($this->client_ip . $this->guid . $this->time())

- sso服务器需暂存该ticket到缓存以ticket:['time' => time(), 'session_id' => session_id]的形式存储, ticket作为缓存的值。

6. 应用A接收到ticket后，访问sso的用户数据获取接口，获取用户数据
   - ticket一次有效，且具有有效时长，sso从缓存中取出ticket，并将存储的时间与当前时间对比
   - 应用A与sso的用户系统存在唯一关联（如用户名，邮箱等），此字段用于关联二者的用户系统，在应用与sso的用户系统中该字段均为唯一且不可更改。
7. sso基于cookie验证当前登录用户，返回用户信息
8. 应用A后端向前端返回用户信息和api_token
   - api_token为应用A自身生成的token，与上面生成的ticket没有关系
   - 应用A自身用户系统应该存储用户生成（即不存在需要创建），唯一关联字段必须保留
9. 应用A将生成的token传递给sso保存起来
   - 参数为生成的api_token（或应用的session_id）和ticket。
   - 这里可以使用消息队列实现。
   - 完成保存后sso服务器将该ticket缓存删除
10. 前端接收到api_token与用户信息，应该存储起来，以后每次向后端请求时必须带有token

应用B登录，此时sso已登录

1. 应用B前端检测token不存在，向应用B后端请求token。
2. 应用B后端判断token不存在，引导用户跳转至登录界面/登录接口(必须是跳转，不是异步访问）。
3. sso基于cookie验证当前登录用户，发现当前有登录的用户。
4. 基于当前登录用户sso生成ticket返回到应用B后端回调接口。

以后步骤同上6-10

关键点：

- 前端需要存储token，有效时间应与后端一致（或比后端略短）
- 前端接收到401响应码，代表验证无token或token已失效，此时前端需根据返回信息控制**跳转（302），由于前后端分端的情况下，后端无法直接控制跳转，需由前端控制。**根据返回的redirect参数控制跳转的页面，未启用sso时，返回中redirect不存在或为空，此时前端跳转至自身的登录页面。
- 应用要实现单点登出必须有自己有用户系统，并能实现自己的登录（不一定是基于token）

单点登出：

1. 应用A需要登出，首先访问sso登出接口**（302跳转）**
2. 依次将sso中保存的各应用凭证发送给所有接入sso的应用的登出接口（预先记录的）
3. sso自身登出，返回sso登录页面。

注：

- 由于sso通过cookie-session验证登录，在不跳转的情况下不一定能够访问到cookie，如果访问不到自然也就不能实现异步的登出。
- sso通知其他应用退出应当使用队列，异步执行。
- 其他的应用退出也分为2种情况
  - 前后端分离：发送的是token，应用在后端将token失效即可。

```
//将token附在请求头上
Auth::user()->token()->delete();
```

  - 前后端不分离：发送的是session_id，应用将session失效即可。

```
//cookie中的session_id是加密的，由中间件自动解析
//我们存储的是解密过后的session_id所以只能通过操作session实现
Session::forget('key1');
```

细节：

sso缓存格式：

$ticket：{create_time: $time，session_id: $session_id，status: $status}

- $ticket为单次登录请求生成的ticket
- $time为ticket生成时间，ticket具有有效时间，初步设想为2分钟
- $session_id为sso本地用户session_id
- $status为ticket使用情况，ticket为单次有效。

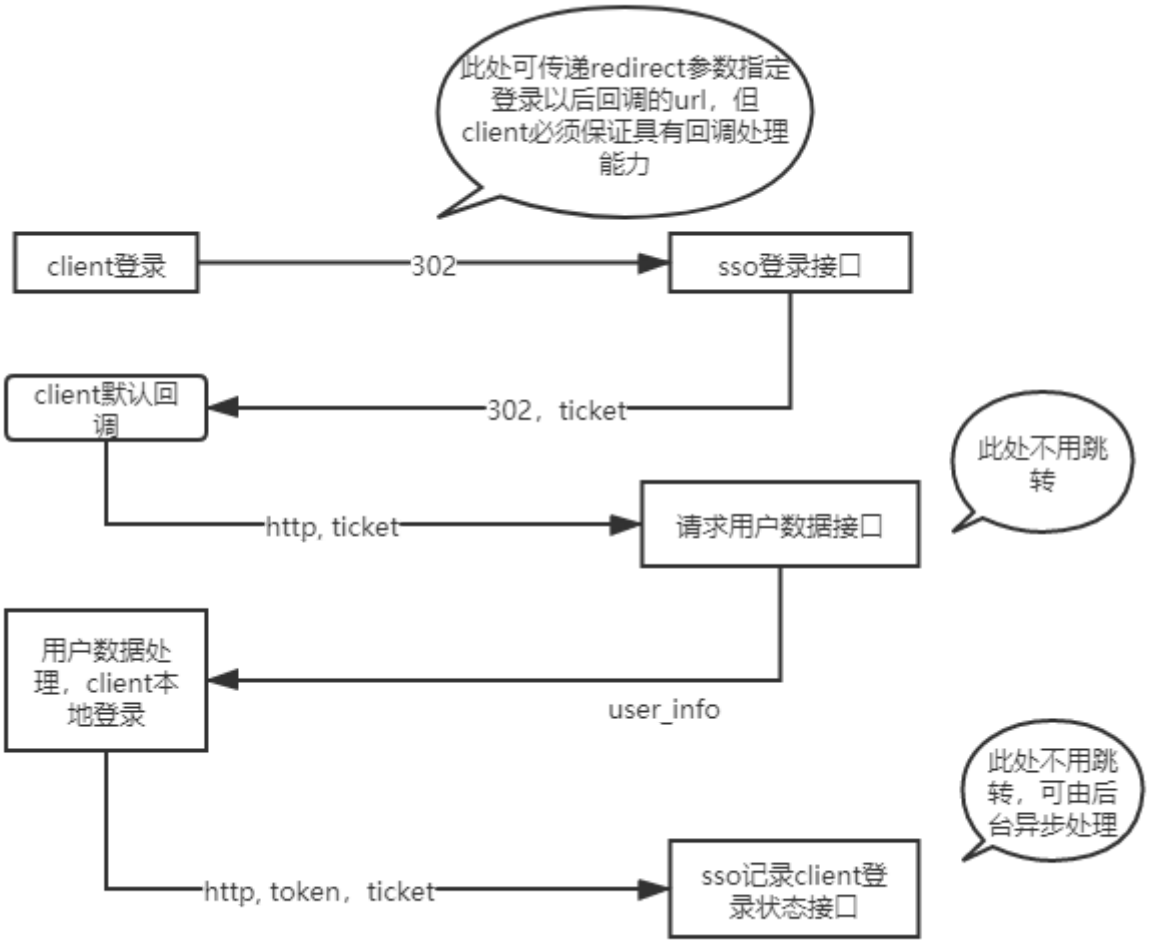
sso数据库设计：

sso_clients:
id client_id　客户端id

client_name　接入客户端名称
client_default_redirect　客户端默认回调url
client_logout_interface　客户端登出接口
is_valid　是否有效

sso_session_client
id client_id sso_session_id　sso上登录的用户id client_token　客户端上登录的用户凭证，session_id或token

登录请求过程数据传递



登出过程数据传递