oauth2-sso-server：是授权服务器和资源服务器和单点登录服务器；

oauth2-sso-client：是第三方系统；

授权应用：来自于oauth2-sso-server，在oauth2-sso-server中注册一个授权应用，得到密钥，第三方系统使用密钥进行参数提交；

## 1、开关单点登录

当开启单点登录的时候，访问第三方系统会先执行is\_login请求，然后server回复是否已经登录，若未登录就返回未登录的信息，第三方系统收到信息之后重定向到server的授权登录界面，输入正确的用户名密码之后，server会判断该用户是否已经授权过了当前的授权应用，如果未授权，显示确认授权的界面，授权之后，就会重定向回第三方系统，完成一个授权登录的过程。同时会把登录信息保存在server的数据库上，其他第三方系统访问的时候同样访问is\_login，只要useragent和ip相同，那么就判定为已登录了，并返回相应的信息。只需要useragent和ip就可以判断了，不需要client\_id，例如client\_id1已经授权登录，当client\_id2的应用访问is\_login的时候，server应该告诉它已有用户登录，然后看这个用户是否已经授权了client\_id2，如果已经授权了的话就直接回到client\_id2的redirect\_uri，如果未授权就显示授权确认页面，这个时候允许你更换用户，但一旦更换之后，client\_id1的应用就同样的逻辑，user2是否对client\_id1授权，授权就变为登录user2。两个系统使用各自的token（因为token决定权限）：

{

"f40a2b7785aee5e933f8":{"access\_token":"b9b3addd0c032e1f3f2acefeff90db77c1dd01b8","expires\_in":1575361204,"token\_type":"Bearer","scope":"userinfo userdetail","refresh\_token":"79b96d07a7e51ebc32045a7419684544566ff837"},

"cf0d966f5f6d48e2db03":{"access\_token":"a580c5186aef0f33f1e51fd04aa8f6309064df9e","expires\_in":1575361216,"token\_type":"Bearer","scope":"userinfo userdetail","refresh\_token":"4f482c89bf9ce6ef793a30ecac33b7b1cefacf49"}

}

如果是关闭单点登录，各个系统使用session来记录登录状态，访问接口的时候就要提供cookie参数。这个模式下，当前没有判断用户是否已经授权过某个应用，也就是每次授权登录都要点击确认授权，而上面的开启单点登录的逻辑，我是加了数据库判断的，已经授权过的话，就不会显示确认授权的界面，这个是模仿github做的，github有一个管理页面，可以显示授权过了那些第三方应用并可取消授权，而我还没有做这部分的逻辑，所以要取消授权目前只能在数据库中删除oauth\_auth\_list中对应的数据。

切换单点登录，需要修改server的配置文件config.php的$config["sso"]=true或false，然后因为我没有额外写第三方系统，所以当前的第三方系统也是可以切换是否开启单点登录的，同样是改配置文件的$config["sso"]=true或false，然后必须server和第三方系统的配置一致，如果不一致无法正常使用，因为有些接口需要的参数不一样以及整个判定逻辑都是不一样的。

## 2、代码说明

说明一下server中的代码，在autoload.php中，引入配置文件，如果是开启单点登录，就加载oauth\_sso\_lib类库，如果关闭就加载oauth\_lib类库，并且设置别名让它不管是开启还是关闭，都是通过同一个名字来调用，例如在控制器Oauth中，使用$this->oauth\_lib->authorize();，这样在维护上，控制器、模型、视图应该都是不用更改的，除非是增加功能，维护的时候只需要修改类库中的oauth\_lib.php和oauth\_sso\_lib.php以及它们各自引入的文件oauth\_server.php、oauth\_sso\_server.php。

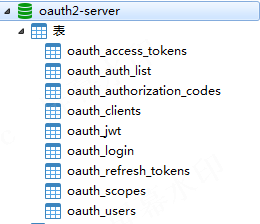
而第三方系统的代码上因为内容较少就没有使用类库的方法进行处理，只是在代码中通过判断配置是否为true，为true就执行开启单点登录的逻辑，否则就另外的逻辑。

## 3、功能说明

先登录上server（现在没有注册功能，直接登录），打开我的应用，新建一个oauth应用，申请应用需要注意Authorization callback URL，要填写redirect\_key和redirect\_uri，这个redirect\_key在某些接口上需要提交这个参数，用来一一对应redirect\_uri，成功之后得到client\_id和client\_secret，然后开发者就可以使用此client\_id和client\_secret开发自己的应用，当要显示授权的时候就会提示你登录的这个账户是否确认要信任此client\_id的应用，一旦确认，这个oauth应用就可以读取该授权用户信息，而这个oauth是开发者注册的，即开发者可以读取你在server上的信息。访问api需要token信息，这个token信息就是在确认授权之后返回给开发者，可以使用refresh\_token刷新access\_token，当两个token都丢失或过期了，就只能重新授权才能得到。

## 4、数据库说明

server中含有以下表：



oauth\_access\_token：保存access\_token的表，长期使用会有数据堆积，可以使用mysql事件定期清理；

oauth\_auth\_list：用户的授权应用列表，用于判断某个用户是否已经授权过了某个授权应用；

oauth\_authorization\_codes：授权码表，正常情况下，授权码使用之后就会删除，所以仅出现在不正常的情况下，才会有数据堆积，也可以使用事件定期清理；

oauth\_clients：授权应用表，特别要注意的是redirect\_uri，我修改过库的源码，让它能一个应用对应多个第三方系统，如{"test":"http://192.168.5.159/oauth2-sso-client/Oauth/test\_redirect", "test2":"http://192.168.5.159/oauth2-sso-client2/Oauth/test\_redirect"}，这样当相同的client\_id访问的时候，通过redirect\_key是test还是test2可以通过验证并重定向回正确的第三方系统；

oauth\_jwt：jwt验证方式时使用，目前未使用过；

oauth\_login：登录记录表，开启单点登录后会查询此表，通过useragent和ip得到数据，再判断是否已登录是否过期是否已授权token是否过期，然后给出相应的回复；

oauth\_refresh\_tokens：保存refresh\_token的表，长期使用会有数据堆积，可以使用mysql事件定期清理；

oauth\_scopes：权限表，server开发者设计，可以修改内容，然后每个api都应该调用一下require\_scope("userinfo等")来验证权限；

oauth\_users：用户表，第三方系统登录的时候要输入的就是这里的用户名密码。

第三方系统目前只有一个表：



当第三方系统使用授权登录之后，如果登录是成功的，就根据返回的某些信息如user\_id，然后md5(xxx.user\_id)，并把基本的用户信息保存进去，这样第三方系统就可以使用md5(xxx.user\_id)来连接其他的表，不过目前这个表没有标识用户是内部用户还是授权用户，内部用户是有密码的，在关闭单点登录的情况下可以使用内部账号登录系统，进行自己系统内部的各种操作，而授权用户是没有密码的，因为第三方是用于无法知道server的密码，保存进用户表仅作为与其他表连接

## 5、参考文献

[PHP下的Oauth2.0尝试 - 授权码授权(Authorization Code Grant)](https://segmentfault.com/a/1190000016408592)

[oauth类库主页](https://bshaffer.github.io/oauth2-server-php-docs/)

[GitHub OAuth 第三方登录示例教程](http://www.ruanyifeng.com/blog/2019/04/github-oauth.html)