

alpsCloud 统一认证中心使用说明

版本号:V1.0.1

@Date:2019/12/12

技术: springsecurity+oauth2 + RBAC

微服务 Provider:

1. alps-system-admin --- RBAC 权限服务
2. alps-oauth-uaa-admin --- OAuth2 管理服务
3. alps-oauth-server --- OAuth2 基础服务

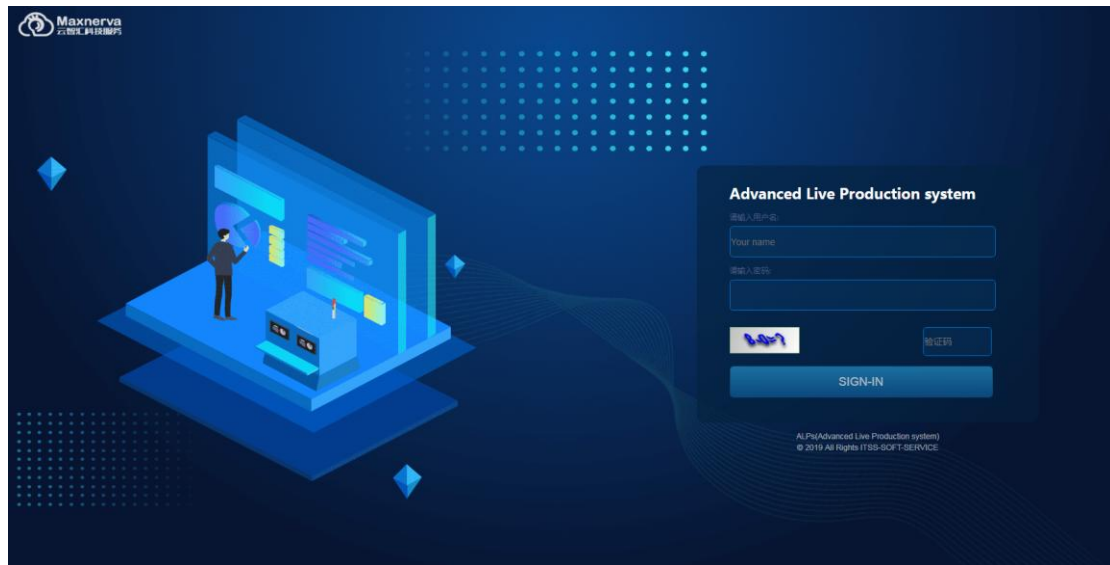
一: RBAC 权限服务

提供以下功能:

- | | | |
|-----------|------|----------------|
| 1. 用户管理 | ---- | 用户/账号 操作 |
| 2. 角色管理 | ---- | 角色操作,权限设定 |
| 3. 租户管理 | ---- | 租户管理 |
| 4. 部门管理 | ---- | 部门管理 |
| 5. 应用管理 | ---- | 应用操作,接口设定,权限管控 |
| 6. 岗位管理 | ---- | 岗位=位管理 |
| 7. 数据字典 | ---- | 数据字典 |
| 8. 账号在线管理 | ---- | 账号在线管理 |

1. 启动 8001 端口的 sysPro 服务.
2. 输入网址 <http://10.202.86.138:8001/> 输入用户名和密码进入 RBAC 系统

可在系统下面设定所有 RBAC 相关操作.



输入用户名和密码进入



进入到系统管理下面可进行所有的 RBAC 操作。



二:oauth2 使用

*暂无操作页面,如有 Application 先手动生成 app 的 key,secret,password,并手动添加到以下 Table.

---- 执行 KeyRandomGen.java

```

3 /**
4  * @author:Yujie.lee
5  * Date:2019年12月11日
6  * Todo 手动生成 appKey,appSecretKey以及password
7  */
8 public class KeyRandomGen {
9
10     /**
11      *date:2019年12月11日
12      *Author:Yujie.lee
13      */
14     public static void main(String[] args) {
15         // TODO Auto-generated method stub
16
17         String appKey = RandomValueUtils.randomAlphanumeric(32);
18         String appSecretKey = RandomValueUtils.randomAlphanumeric(32);
19         String password = encryptPassword(appSecretKey);
20         System.out.println("appKey === > " + appKey);
21         System.out.println("appSecretKey === > " + appSecretKey);
22         System.out.println("password === > " + password);
23
24     }
25
26     public static PasswordEncoder passwordEncoder() {
27         return new BCryptPasswordEncoder();
28     }
29 }

```

将生成的 appKey, appSecretKey 以及 password 以及 app 信息添加到相应的 App 及 Oauth2 Table 中.

```

appKey === > f4tPX71UQoas4K5JJeLXyPINLXCFcCC7
appSecretKey === > ZCsYZUOYQwTj4qwGaKxBgxAWD4zODOrK
password === > $2a$10$Pj9AeOVH/Yf7nVYGC.cqieNLJED6r31B8.WPWR0ZYk0CfNE7finQO

```

Alps oauth2 目前支持:

- I. 密码(password)授权模式
- II. 客户端(Client)授权模式
- III. 授权码模式
- IV. 隐式授权码模式

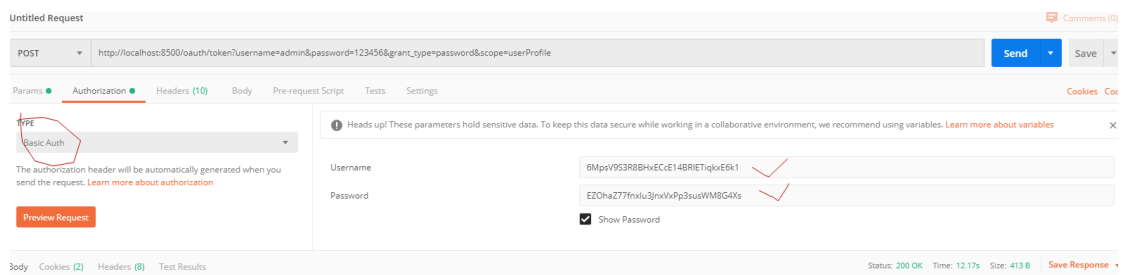
以下分别介绍 4 种 Oauth2 的 4 种使用方法

1. 密码(password)模式

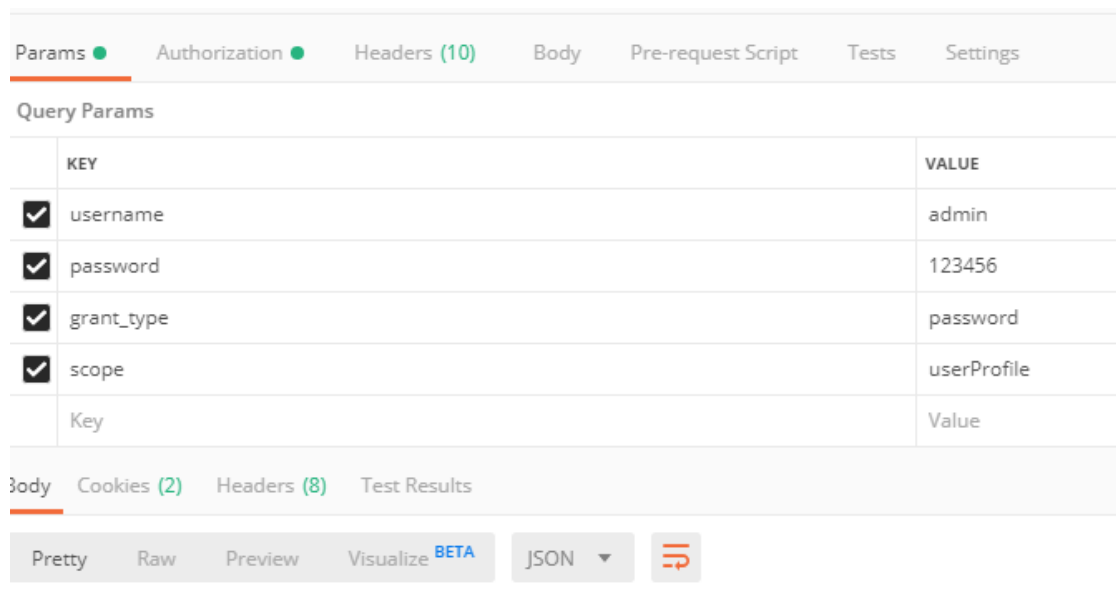
URI: <http://localhost:8500/oauth/token>

利用 postman

先设置请求头 basic 方式配置 client_id 和 client_secret



再设置 params



如果都正确确认请求会返回所需要的 Token

```
1 {
2   "access_token": "1ba572d3-5eed-4424-afc1-f9456978c9e1",
3   "token_type": "bearer",
4   "expires_in": 42013,
5   "scope": "userProfile",
6   "openid": 1,
7   "domain": "@admin.com"
8 }
```

拿到 Token 后,就可以访问相应的接口获取数据了.

Untitled Request

GET http://localhost:8500/current/user

Params Authorization Headers (9) Body Pre-request Script Tests Settings

TYPE OAuth 2.0

The authorization data will be automatically generated when you send the request. [Learn more about authorization](#)

Add authorization data to Request Headers

Preview Request

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment

Access Token 03c7896e-15e5-4e10-b780-1a97d8fbd32f

Get New Access Token

Body Cookies (2) Headers (10) Test Results

如果访问没有授权的资源会返回权限不足.

GET http://localhost:8500/current/user

Params Authorization Headers (9) Body Pre-request Script Tests Settings

TYPE OAuth 2.0

The authorization data will be automatically generated when you send the request. [Learn more about authorization](#)

Add authorization data to Request Headers

Preview Request

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment

Access Token

Body Cookies (2) Headers (9) Test Results

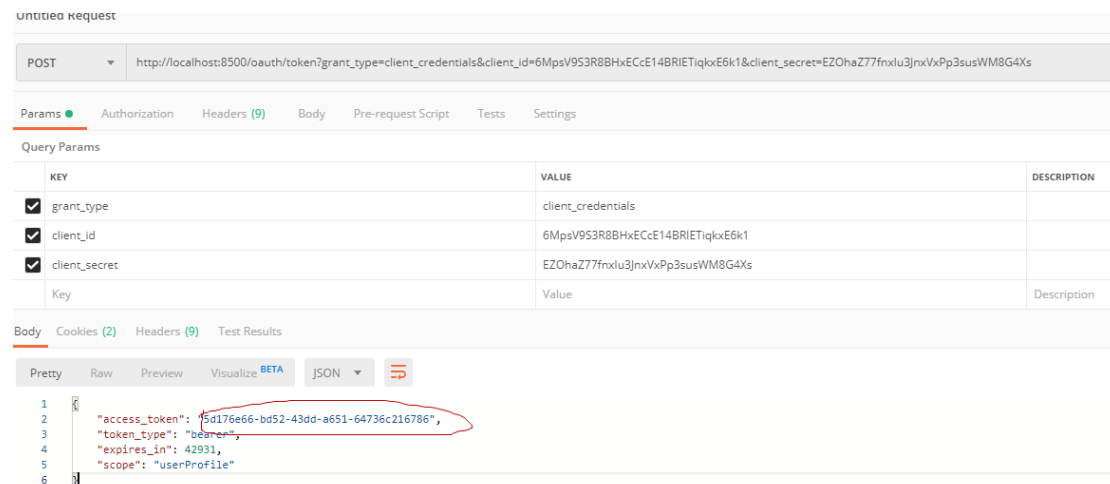
```
1 {
2   "error": "invalid_token",
3   "error_description": "Invalid access token: 03c7896e-15e5-4e10-b780-1a97d8fbd32f"
4 }
```

2. 客户端模式

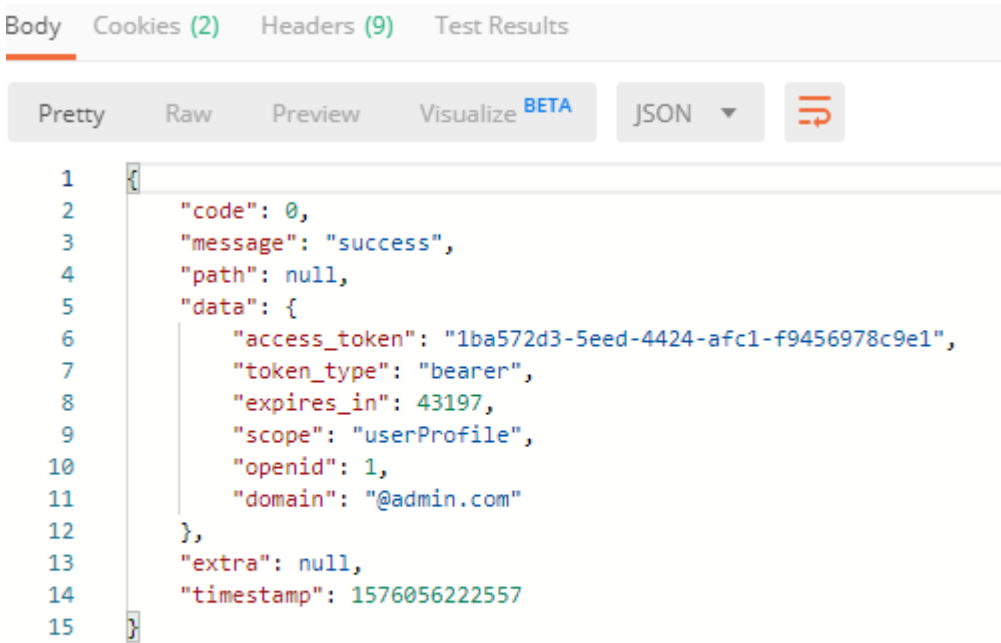
URI: `http://localhost:8500/oauth/token?grant_type=client_credentials&client_id=6MpsV9S3R8BHxECCe14BRIETiqkxE6k1&client_secret=EZOhaZ77fnxlu3JnxVxPp3susWM8G4Xs`

通过 **Client** 的 **key** 与 **secret** 获取 **Token** 进行访问

如下操作



如果都正确请求会返回所需要的 **Token**



3. 授权码模式

给第三方授权,并支持 Token 刷新

URI:

http://localhost:8500/oauth/authorize?response_type=code&client_id=6MpsV9S3R8BHxECCe14BRIETiqkxE6k1&redirect_uri=http://hr.maxnerva.cn:9000

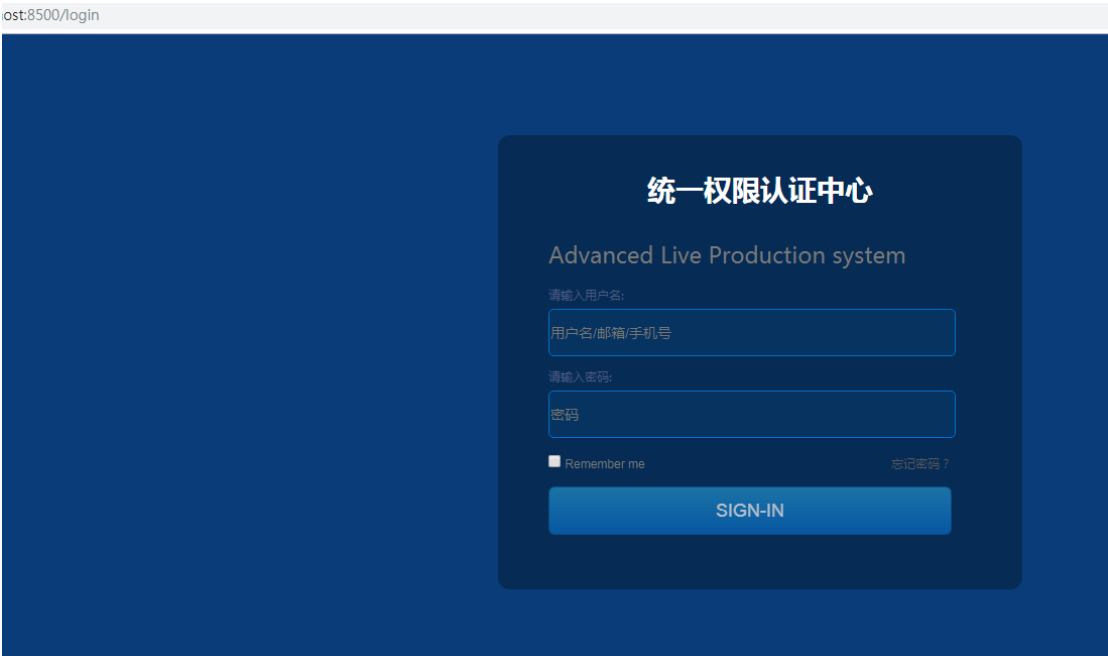
首先的 APP Table 增加相应的 APP 信息.

app_id	api_key	secret_key	app_name	app_name_en
123545625	I0lqybNA2VbbsFzmVqb6DQ6PkG1t2bK	NxW9mJb95pUiw6po0BSh6MCBqBnsigSJ	test	test
1552274783265	6MpsV9S3R8BHxECCe14BRIETiqkxE6k1	EZOhaZ77fmxJu3JnxVxPp3susWM8G4Xs	平台用户认证服务器	alps-uae-admin-server
1552294656514	rOOM15Zbc3UPWgW2h71gRFvi	2Iw2B0UCMYd1cZjt8Fpr4KJUX75wQCwW	SSO单点登录	sso-demo
1552294656516	1eMFzjoVhSrhI5rpJPKt7gOdyIS5zT5678	f5mOf9RjVwV168IF7samrzgUBMAxJnmng	人事系统	HR
1552294656517	pJ6tNskHdpT3NQYOU3WVwKG2o9L0wOn	S9cr8OCzEqYYWAH2hczfIpKcXn9XvbBY	aplcloud官网	alps

再在 oauth2 的 Table oauth_client_details 中添加相关信息

开始事务 备注 筛选 排序 导入 导出				
id	client_id	client_secret	resource_ids	scope
1	1eMFzjoVhSrhI5rpJPKkI7gOdyiIS5zT5678	\$2s\$10\$tD4kD6Xpl5zs9.L6tKCb.GUoywjrxHXTXLSdIvMuJZb7zHM4ZC	(Null)	userProfile
2	6Mpsv9S3R8BHxECce14BRlETqlxE6k1	\$2s\$10\$n860AgnhpNNor7w2FQlcf.ZpXfz3xMJGr.oeo6YX7qn/Q29M9XG2		userProfile
3	IOlqybNA2VbbsFzmVqb6DQ6PksG1t2bK	\$2s\$10\$HK2rswSOFEUSl6f7yw4zM.LRe.SCEPYmJZM72eqgpZIGKNO0FTmja	(Null)	userProfile
4	pJ6tNsKhdIpT3NQYOU3WVvskG2o9L0wOn	\$2s\$10\$1HiSmNBa8BE54TkqCAYXZ.2xg9Wkh0Pp1swz1u3o1RXiBu2vJ6pKy	(Null)	userProfile
5	rOOM15Zbc3UPWgW2h71gRFvi	\$2s\$10\$NSb94skA5i9ks/F0mUBehuBs9GbvIv8wdxQOYMG3WxMRt0nyP2Xn.		userProfile

引导用户跳转到统一权限登录页面:



输入用户名和密码后,会跳转到授权页面



同意给第三方应用授权后就会中转到第三方应用,并携带上相应的授权码。

不安全 | hr.maxnerva.cn:9000/?code=4dxl4I



再利用获取的授权码 code 获取对应的 Token 进行访问

http://localhost:8500/oauth/token?grant_type=authorization_code&client_id=6MpsV9S3R8BHxECCe14BRIETiqkxE6k1&client_secret=EZOhaZ77fnxlu3JnxVxPp3susWM8G4Xs&redirect_uri=http://hr.maxnerva.cn:9000&code=lx8zoV

Untitled Request

POST http://localhost:8500/oauth/token?grant_type=authorization_code&client_id=6MpsV9S3R8BHxEcCE14BRIETiqkxE6k1&client_secret=E...

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> grant_type	authorization_code	
<input checked="" type="checkbox"/> client_id	6MpsV9S3R8BHxEcCE14BRIETiqkxE6k1	
<input checked="" type="checkbox"/> client_secret	EZOhaZ77fnxlu3JnxVp3susWM8G4Xs	
<input checked="" type="checkbox"/> redirect_uri	http://hr.maxnerva.cn:9000	
<input checked="" type="checkbox"/> code	lx8zoV	

Key Value Description

body Cookies (2) Headers (9) Test Results Status: 200 OK Time: 4.98s

Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "access_token": "04846ab3-165a-42f1-bc69-40bb1275061a",
3   "token_type": "bearer",
4   "expires_in": 43199,
5   "scope": "userProfile",
6   "openid": 1,
7   "domain": "@admin.com"
8 }
```

4. 隐式授权码获取 Token (参考授权码模式, 略)

5. 开发者通过平台账号密码快速接口获取平台 Token.

URI: <http://localhost:8500/client/token?username=admin&password=123456>

POST http://localhost:8500/client/token

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL BETA JSON

```
1 {
2   "username": "admin",
3   "password": "123456"
4 }
```

Body Cookies (2) Headers (9) Test Results

Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "code": 0,
3   "message": "success",
4   "path": null,
5   "data": {
6     "access_token": "fe5441df-101b-466d-ac88-63598a4f0b3d",
7     "token_type": "bearer",
8     "expires_in": 2561,
9     "scope": "userProfile",
10    "openid": 1,
11    "domain": "@admin.com"
12  },
13   "extra": null,
14   "timestamp": 1577950996993
15 }
```

6.oauth2 与网关结合

第一步,增加相应的 APP 信息.

增加 **base_app** 与 **oauth_client_details** 两个 Table 中 app 信息.

第二步,配置 gateway 中路由:

其中路由 ID 必须与 **oauth_client_details** 中的 **Client_id** 相同

```
- id: jt9ELdgfjTVpXmgIJjys4vawnvAJ0mB5
  uri: lb://test
  order: 105
  predicates:
    - Path=/test/**
  filters:
    - AlpsSwaggerHeaderFilter
    - StripPrefix=1
    - name: Hystrix
  args:
```

第三步,在请求获取 Token 需将 **Client_id** 同用户名信息一起请求 Token.生成的 Token 后续只能作为访问皮 **client** 的凭证.

POST http://localhost:9200/oauth/client/token

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL BETA JSON

```
1 {
2   "username": "admin",
3   "password": "123456"
4 }
```

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	appKey	jt9ELdgfjTVpXmgIJjys4vawnvAJ0mB5
	Key	Value

Alpscloud 授权使用方法及约定

1. 密码模式:

- 客户端(Client)要求用户登录,输入用户名和密码
- 将 Client 端的用户名,密码以及客户端约定的 client_id + client_secret 发送给授权服务器
- 授权服务器校验用户名、用户密码、client_id、client_secret,均通过返回 token 到客户端
- 客户端保存 token
- 客户请求携带 Token 访问资源服务器

使用范围: **遗留老旧系统**

支持令牌刷新: **是**

2.客户端模式:

- 客户端(Client)使用约定的 client_id + client_secret + 授权模式标识访问授权服务器
- 授权服务器校验通过返回 token 给客户端
- 客户端保存 token
- 客户请求携带 Token 访问资源服务器

使用范围: **平台内部服务/内部 app/内部高信任系统之间的 api 访问**

支持令牌刷新: **否**

3.授权码模式:

- 客户端发送 client_id + client_secret + 授权模式标识(grant_type) + 回调地址(redirect_uri)拼成 url 访问授权服务器
- 授权服务器引导用户跳转到登录界面,要求用户登录(此时用户提交的密码等直接发到授权服务器,进行校验)
- 授权服务器返回授权审批确认页面,要求用户授权
- 授权服务器接到授权后返回授权码到回调地址
- 客户端接收到授权码,并使用授权码 + client_id + client_secret 访授权服务器,并拿到返回的 Token
- 客户端保存 token
- 客户请求携带 Token 访问资源服务器

使用范围: **第三方 Web 服务器端应用与第三方原生 App**

支持令牌刷新: **是**

4.隐式授权码模式:

- 客户端发送 client_id + client_secret + 授权模式标识(grant_type) + 回调地址(redirect_uri)拼成 url 访问授权服务器
- 授权服务器引导用户跳转到登录界面，要求用户登录（此时用户提交的密码等直接发到授权服务器，进行校验）
- 授权服务器返回授权审批确认页面，要求用户授权
- ~~➤ 授权服务器接到授权后返回授权码到回调地址~~
- 客户端接收到授权码，并使用授权码 + client_id + client_secret 访授权服务器,并拿到返回的 Token
- 授权服务器接到授权后返回 Token 到回调地址
- 客户端保存 token
- 客户请求携带 Token 访问资源服务器

使用范围: 没有后台服务的纯 **Web** 页面

支持令牌刷新: **否**