**[#](https://pig4cloud.com/" \l "token-%E6%9C%89%E6%95%88%E6%9C%9F%E8%AF%B4%E6%98%8E) Token 有效期说明**

// 默认刷新token 的有效期

private int refreshTokenValiditySeconds = 60 \* 60 \* 24 \* 30; // default 30 days.

// 默认token 的有效期

private int accessTokenValiditySeconds = 60 \* 60 \* 12; // default 12 hours.

每个终端配置为空，去默认的时间设置 刷新token 30天，令牌12小时。  
前端使用的客户端是pig，设置令牌失效等。 

[**#**](https://pig4cloud.com/#%E4%B8%AA%E6%80%A7%E5%8C%96token-%E7%9B%AE%E7%9A%84) **个性化Token 目的**

* 默认通过调用 /oauth/token 返回的报文格式包含以下参数

{

"access\_token": "e6669cdf-b6cd-43fe-af5c-f91a65041382",

"token\_type": "bearer",

"refresh\_token": "da91294d-446c-4a89-bdcf-88aee15a75e8",

"expires\_in": 43199,

"scope": "server"

}

并没包含用户的业务信息比如用户信息、租户信息等。

* 扩展生成包含业务信息（如下）,避免系统多次调用，直接可以通过认证接口获取到用户信息等，大大提高系统性能

{

"access\_token":"a6f3b6d6-93e6-4eb8-a97d-3ae72240a7b0",

"token\_type":"bearer",

"refresh\_token":"710ab162-a482-41cd-8bad-26456af38e4f",

"expires\_in":42396,

"scope":"server",

"tenant\_id":1,

"license":"made by pigx",

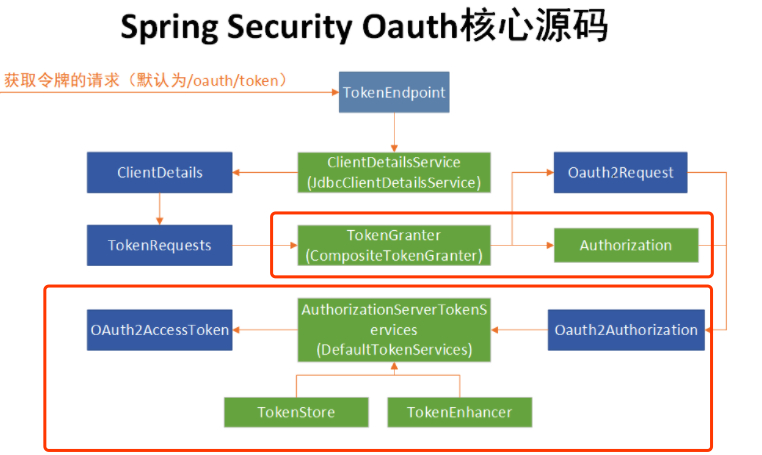
"dept\_id":1,

"user\_id":1,

"username":"admin"

}

[**#**](https://pig4cloud.com/#%E5%AF%86%E7%A0%81%E6%A8%A1%E5%BC%8F%E7%94%9F%E6%88%90token-%E6%BA%90%E7%A0%81%E8%A7%A3%E6%9E%90) **密码模式生成Token 源码解析**



​ 主页参考红框部分

* ResourceOwnerPasswordTokenGranter （密码模式）根据用户的请求信息，进行认证得到当前用户上下文信息
* protected OAuth2Authentication getOAuth2Authentication(ClientDetails client, TokenRequest tokenRequest) {
* Map<String, String> parameters = new LinkedHashMap<String, String>(tokenRequest.getRequestParameters());
* String username = parameters.get("username");
* String password = parameters.get("password");
* // Protect from downstream leaks of password
* parameters.remove("password");
* Authentication userAuth = new UsernamePasswordAuthenticationToken(username, password);
* ((AbstractAuthenticationToken) userAuth).setDetails(parameters);
* userAuth = authenticationManager.authenticate(userAuth);
* OAuth2Request storedOAuth2Request = getRequestFactory().createOAuth2Request(client, tokenRequest);
* return new OAuth2Authentication(storedOAuth2Request, userAuth);
* }
* 然后调用AbstractTokenGranter.getAccessToken() 获取OAuth2AccessToken
* protected OAuth2AccessToken getAccessToken(ClientDetails client, TokenRequest tokenRequest) {
* return tokenServices.createAccessToken(getOAuth2Authentication(client, tokenRequest));
* }
* 默认使用DefaultTokenServices来获取token
* public OAuth2AccessToken createAccessToken(OAuth2Authentication authentication) throws AuthenticationException {
* ... 一系列判断 ，合法性、是否过期等判断
* OAuth2AccessToken accessToken = createAccessToken(authentication, refreshToken);
* tokenStore.storeAccessToken(accessToken, authentication);
* // In case it was modified
* refreshToken = accessToken.getRefreshToken();
* if (refreshToken != null) {
* tokenStore.storeRefreshToken(refreshToken, authentication);
* }
* return accessToken;
* }
* createAccessToken 核心逻辑
* // 默认刷新token 的有效期
* private int refreshTokenValiditySeconds = 60 \* 60 \* 24 \* 30; // default 30 days.
* // 默认token 的有效期
* private int accessTokenValiditySeconds = 60 \* 60 \* 12; // default 12 hours.
* private OAuth2AccessToken createAccessToken(OAuth2Authentication authentication, OAuth2RefreshToken refreshToken) {
* DefaultOAuth2AccessToken token = new DefaultOAuth2AccessToken(uuid);
* token.setExpiration(Date)
* token.setRefreshToken(refreshToken);
* token.setScope(authentication.getOAuth2Request().getScope());
* return accessTokenEnhancer != null ? accessTokenEnhancer.enhance(token, authentication) : token;
* }

如上代码，在拼装好token对象后会调用认证服务器配置TokenEnhancer( 增强器) 来对默认的token进行增强。

* TokenEnhancer.enhance 通过上下文中的用户信息来个性化Token
* public OAuth2AccessToken enhance(OAuth2AccessToken accessToken, OAuth2Authentication authentication) {
* final Map<String, Object> additionalInfo = new HashMap<>(8);
* PigxUser pigxUser = (PigxUser) authentication.getUserAuthentication().getPrincipal();
* additionalInfo.put("user\_id", pigxUser.getId());
* additionalInfo.put("username", pigxUser.getUsername());
* additionalInfo.put("dept\_id", pigxUser.getDeptId());
* additionalInfo.put("tenant\_id", pigxUser.getTenantId());
* additionalInfo.put("license", SecurityConstants.PIGX\_LICENSE);
* ((DefaultOAuth2AccessToken) accessToken).setAdditionalInformation(additionalInfo);
* return accessToken;
* }

[**#**](https://pig4cloud.com/#%E5%9F%BA%E4%BA%8Epig-%E7%9C%8B%E4%B8%8B%E6%9C%80%E7%BB%88%E7%9A%84%E5%AE%9E%E7%8E%B0%E6%95%88%E6%9E%9C) **基于pig 看下最终的实现效果**

Pig 基于Spring Cloud、oAuth2.0开发基于Vue前后分离的开发平台，支持账号、短信、SSO等多种登录，提供配套视频开发教程。  
<https://gitee.com/log4j/pig>

