[**#**](https://pig4cloud.com/#%E5%B0%9D%E8%AF%95%E4%BD%BF%E7%94%A8) **尝试使用**

pig 提供了一个SSO的客户端,**pig-sso-demo** ![](images/screenshot\_1527031474205.png =400x400)

* 注意看PigxSsoClientDemoApplication注释

/\*\*

\* @author lengleng

\* @date 2018年11月15日17:06:26

\* <p>

\* 单点登录客户端

\* 1. 启动实例访问:http://localhost:4040/sso1/ 提示登录，然后获取到用户信息

\* 2. 再启动一个实例： http://localhost:4041/sso1/ 不需要登录即可获取当前用户信息

\*/

@EnableOAuth2Sso

@SpringCloudApplication

public class PigxSsoClientDemoApplication {

public static void main(String[] args) {

SpringApplication.run(PigxSsoClientDemoApplication.class, args);

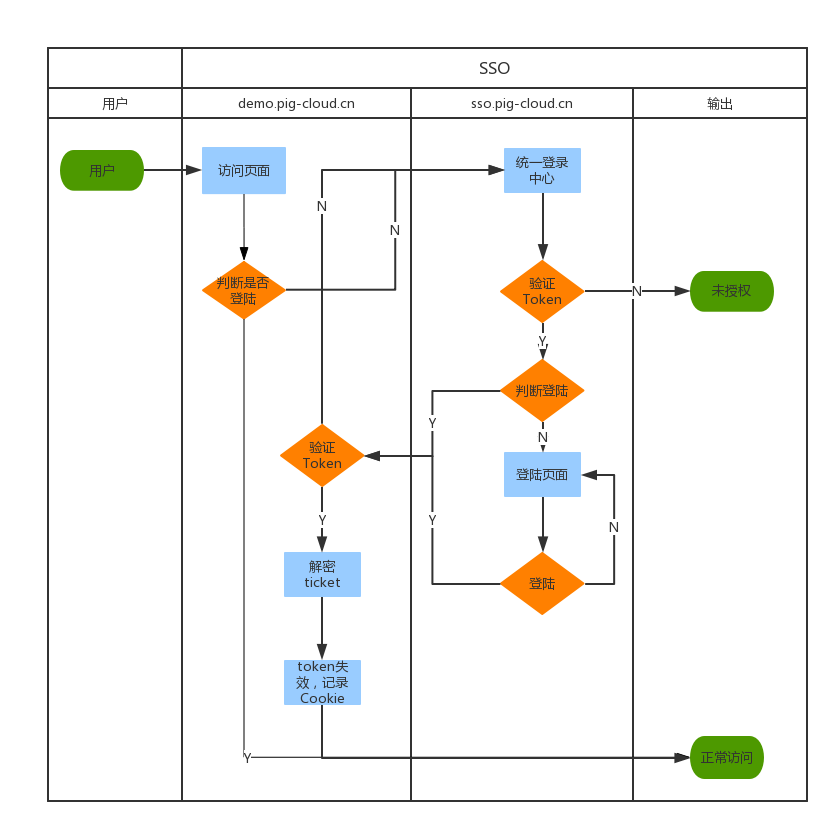
}

}

1. 访问单点登录：http://localhost:4040/sso1/

2. 跳转至统一认证界面 （3000 统一认证）

3. 重定向回 http://localhost:4040/sso1/ （4040携带用户信息）



[**#**](https://pig4cloud.com/#%E5%8D%95%E7%82%B9%E7%99%BB%E5%BD%95%E6%A6%82%E5%BF%B5) **单点登录概念**

单点登录（Single Sign On），简称为 SSO，是目前比较流行的企业业务整合的解决方案之一。SSO的定义是在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。登录逻辑如上图

[**#**](https://pig4cloud.com/#%E5%9F%BA%E4%BA%8Espring-%E5%85%A8%E5%AE%B6%E6%A1%B6%E7%9A%84%E5%AE%9E%E7%8E%B0) **基于Spring 全家桶的实现**

技术选型：

Spring Boot

Spring Cloud

Spring Security oAuth2

[**#**](https://pig4cloud.com/#%E5%AE%A2%E6%88%B7%E7%AB%AF%EF%BC%9A) **客户端：**

maven依赖

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.security.oauth</groupId>

<artifactId>spring-security-oauth2</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-jwt</artifactId>

</dependency>

[**#**](https://pig4cloud.com/#enableoauth2sso-%E6%B3%A8%E8%A7%A3) **EnableOAuth2Sso 注解**

入口类配置@@EnableOAuth2Sso

@SpringBootApplication

public class PigSsoClientDemoApplication {

public static void main(String[] args) {

SpringApplication.run(PigSsoClientDemoApplication.class, args);

}

}

[**#**](https://pig4cloud.com/#%E9%85%8D%E7%BD%AE%E6%96%87%E4%BB%B6) **配置文件**

security:

oauth2:

client:

client-id: pig

client-secret: pig

user-authorization-uri: http://localhost:3000/oauth/authorize

access-token-uri: http://localhost:3000/oauth/token

scope: serve

resource:

jwt:

key-uri: http://localhost:3000/oauth/token\\_key

sessions: neve

[**#**](https://pig4cloud.com/#sso%E8%AE%A4%E8%AF%81%E6%9C%8D%E5%8A%A1%E5%99%A8) **SSO认证服务器**

[**#**](https://pig4cloud.com/#%E8%AE%A4%E8%AF%81%E6%9C%8D%E5%8A%A1%E5%99%A8%E9%85%8D%E7%BD%AE) **认证服务器配置**

@Configuration

@Order(Integer.MIN\\_VALUE)

@EnableAuthorizationServe

public class PigAuthorizationConfig extends AuthorizationServerConfigurerAdapter {

@Override

public void configure(ClientDetailsServiceConfigurer clients) throws Exception {

clients.inMemory()

.withClient(authServerConfig.getClientId())

.secret(authServerConfig.getClientSecret())

.authorizedGrantTypes(SecurityConstants.REFRESH\\_TOKEN, SecurityConstants.PASSWORD,SecurityConstants.AUTHORIZATION\\_CODE)

.scopes(authServerConfig.getScope());

}

@Override

public void configure(AuthorizationServerEndpointsConfigurer endpoints) {

endpoints

.tokenStore(new RedisTokenStore(redisConnectionFactory))

.accessTokenConverter(jwtAccessTokenConverter())

.authenticationManager(authenticationManager)

.exceptionTranslator(pigWebResponseExceptionTranslator)

.reuseRefreshTokens(false)

.userDetailsService(userDetailsService);

}

@Override

public void configure(AuthorizationServerSecurityConfigurer security) throws Exception {

security

.allowFormAuthenticationForClients()

.tokenKeyAccess("isAuthenticated()")

.checkTokenAccess("permitAll()");

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public JwtAccessTokenConverter jwtAccessTokenConverter() {

JwtAccessTokenConverter jwtAccessTokenConverter = new JwtAccessTokenConverter();

jwtAccessTokenConverter.setSigningKey(CommonConstant.SIGN\\_KEY);

return jwtAccessTokenConverter;

}

}

[**#**](https://pig4cloud.com/#%E9%85%8D%E7%BD%AE%E5%AE%8C%E6%88%90%E4%BD%93%E9%AA%8C) **配置完成体验**

1. 访问SSO客户端的 index.html
2. 重定向到SSO服务端的 Basic 认证
3. 输入账号密码又重定向到原请求的 客户端index资源