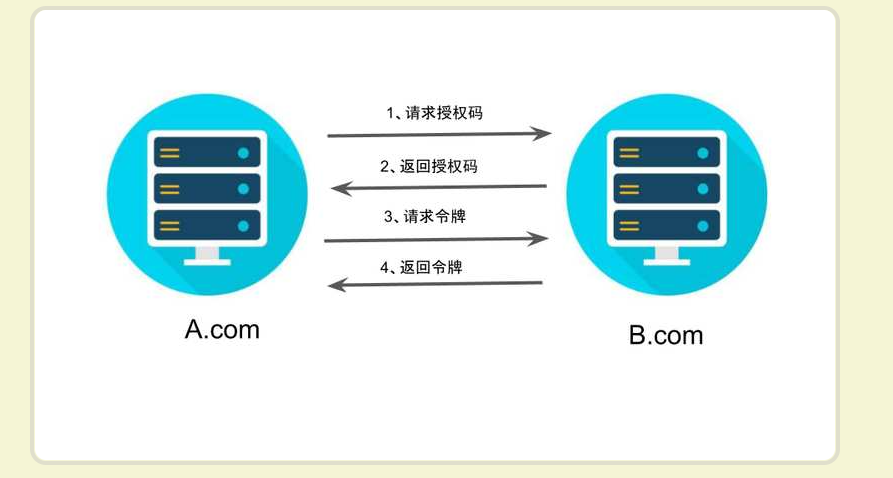
**第三方调用示例说明**

2020-11-12

了解oauth2.0: <http://www.ruanyifeng.com/blog/2019/04/oauth-grant-types.html>



认证服务使用的是oauth认证中的授权码方式，第三方系统要使用认证服务需要提供跳转的地址，由认证系统管理员注册一下拿到 client\_id和client\_secret。

# 一、认证登陆

## 1.获取授权码

获取授权码[http://58.218.126.21:29812/server/oauth/authorize](http://localhost:8080/server/oauth/authorize)

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 是否必须 | 数据类型 | 说明 |
| response\_type | 是 | string | 响应类型，在使用时传code,例如：response\_type=code |
| scope | 是 | String | 使用中传scope=read |
| client\_id | 是 | String | 注册成功由管理员提供 |
| redirect\_uri | 是 | String | 第三方应用跳转地址 |
| 使用示例：  [http://58.218.126.21:29812/server/oauth/authorize?response\_type=code&scope=read&client\_id=111&redirect\_uri=http://localhost:8081/client1/local](http://localhost:8080/server/oauth/authorize?response_type=code&scope=read&client_id=111&redirect_uri=http://localhost:8081/client1/local)  请求成功会回跳到redirect\_uri地址并在地址后面追加code,这时就拿到了授权码  如：http://localhost:8081/client1/local?code=adbdf | | | |

## 2.获取令牌

上面拿到了授权码，接着就是使用授权码去获取token

[http://58.218.126.21:29812/server/oauth/token](http://localhost:8080/server/oauth/token) 请求方式post 数据类型json

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 是否必须 | 数据类型 | 说明 |
| grant\_type | 是 | string | 响应类型，在使用时传authorization\_code,例如：grant\_type =authorization\_code |
| code | 是 | String | 使用中传code(上一个地址请求中获取到的) |
| client\_id | 是 | String | 注册成功由管理员提供 |
| client\_secret | 是 |  |  |
| redirect\_uri | 是 | String | 第三方应用跳转地址 |
| 使用示例：  [http://58.218.126.21:29812/server/oauth/token?grant\_type=authorization\_code&code =adb&client\_id=111&client\_secret=222&redirect\_uri=http://localhost:8081/client1/local](http://localhost:8080/server/oauth/token?grant_type=authorization_code&code%20=adb&client_id=111&client_secret=222&redirect_uri=http://localhost:8081/client1/local)  成功返回：  {     **"access\_token"**:**"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsic29zLXJlc291cmNlIl0sImdyYW50X3R5cGUiOiJhdXRob3JpemF0aW9uX2NvZGUiLCJ1c2VyX25hbWUiOiJhZG1pbiIsInNjb3BlIjpbInJlYWQiXSwiZXhwIjoxNjA1MjE1ODIxLCJhdXRob3JpdGllcyI6WyJST0xFX1VTRVIiLCJST0xFX0FETUlOIl0sImp0aSI6ImMyYjdkYmYwLTk0ZDItNDZkOS1iZTAyLTI5MjQyZDQzMDU5NCIsImNsaWVudF9pZCI6IjBjMDc5MGFhMjllZTRhNWNhYjk1ZTQyZGJmNGJhYmVkIn0.VEVMVs6YTYQ6qzi9wAqLe3p-rSgp1DSYkVNvhlP7Ook"**,     **"token\_type"**:**"bearer"**,     **"expires\_in"**:**43199**,     **"scope"**:**"read"**,     **"jti"**:**"c2b7dbf0-94d2-46d9-be02-29242d430594"** }  **access\_token就是我们需要的。使用access\_token 去获取用户信息。** | | | |

## 3.获取用户信息

获取用户信息[http://58.218.126.21:29812/server/unity/user\_info](http://localhost:8080/server/unity/user_info) 请求方式get请求。

|  |  |  |
| --- | --- | --- |
| 参数 | 是否必须 | 说明 |
| access\_token | 是 | 上面获取到的 |
| 请求示例：  [http://58.218.126.21:29812/server/unity/user\_info?access\_token=accessToken](http://localhost:8080/server/unity/user_info?access_token=accessToken)  成功返回  {     **"guid"**:**"-3394461689692140820"**,     **"username"**:**"yanyi"**,     **"phone"**:**"15198789987"**,     **"realname"**:**"研一"** }  guid就是用户的唯一标识 第三方系统根据之前协商好的 去关联获取用户的信息，完成自己系统的登陆操作。 | | |

例如注册时：提供的第三方系统地址为<http://localhost:8081/client1/local> 此地址访问的是一个servlet类，如下：

## 4.Java代码示例

Java代码示例（只是作为参考，上面提供的都是接口，每个语言调用接口的方式可能会不一样，请自行完成接口调用，可以百度。）**仅供参考**

|  |
| --- |
| package cn.com.zhou.localServer;  import com.alibaba.fastjson.JSONObject;  import org.apache.http.HttpEntity;  import org.apache.http.HttpResponse;  import org.apache.http.HttpStatus;  import org.apache.http.client.methods.CloseableHttpResponse;  import org.apache.http.client.methods.HttpGet;  import org.apache.http.client.methods.HttpPost;  import org.apache.http.entity.StringEntity;  import org.apache.http.impl.client.CloseableHttpClient;  import org.apache.http.impl.client.HttpClients;  import org.apache.http.util.EntityUtils;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import java.io.IOException;  import java.util.Map;  public class OauthLocalServer extends HttpServlet {  @Override  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  String clientId = "0c0790aa29ee4a5cab95e42dbf4babed";  String oauthUrl = "http://58.218.126.21:29812/server/oauth/authorize";  String redirect\_uri = "http://localhost:8081/client1/local";  String code = req.getParameter("code");  if (null == code) {  resp.sendRedirect(oauthUrl + "?response\_type=code&scope=read&client\_id=" + clientId + "&redirect\_uri=" + redirect\_uri);  } else {  togetToken(code);  }  }  public void togetToken(String code) {  String client\_id = "0c0790aa29ee4a5cab95e42dbf4babed";  String client\_secret = "rnXoirfQftzythZgUl1tzsI7JsMsjUfT";  String url = "http://58.218.126.21:29812/server/oauth/token?client\_id="+client\_id+"&client\_secret="+client\_secret+"&grant\_type=authorization\_code&code=" + code + "&redirect\_uri=http://localhost:8081/client1/local";  CloseableHttpClient client = HttpClients.createDefault();  HttpPost httpPost = null;  HttpResponse response = null;  httpPost = new HttpPost(url);  httpPost.setHeader("Content-Type", "application/json;charset=UTF-8");  try {  response = client.execute(httpPost);  response.setHeader("Cache-Control", "no-cache");  System.out.println(response.getStatusLine().getStatusCode());  if (response.getStatusLine().getStatusCode() == HttpStatus.SC\_OK) {  HttpEntity entity = response.getEntity();  String resultString = EntityUtils.toString(response.getEntity(), "utf-8").replaceAll(" ", "");  System.out.println(resultString);  Map<String, Object> m = (Map<String, Object>) JSONObject.parse(resultString);  System.out.println(m.get("access\_token"));  toGet((String) m.get("access\_token"));  }  } catch (Exception e) {  e.printStackTrace();  }  }  public void toGet(String accessToken) throws IOException {  CloseableHttpClient client = HttpClients.createDefault();  HttpGet get = new HttpGet("http://58.218.126.21:29812/server/unity/user\_info?access\_token=" + accessToken);  //设置post请求头  // 使用HttpClient发起请求，返回response  CloseableHttpResponse response = client.execute(get);  response.setHeader("Cache-Control", "no-cache");  String resultString = "";  if (response.getStatusLine().getStatusCode() == HttpStatus.SC\_OK) {  resultString = EntityUtils.toString(response.getEntity(), "utf-8").replaceAll(" ", "");  System.out.println(resultString);  }  }  @Override  protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  doGet(req, resp);  }  } |

# 二、认证退出

## 1.对客户端要求

**第一点**：第三方系统在获取用户信息时会拿到令牌也就是token，需要客户端暂存token，方便后面的接口调用使用，在第三方系统退出时需要清除这个token，这个token的存储可以使用map 存储，使用获取用户信息接口返回的guid关联自己系统的登陆名称作为map的key，token作为map的值，这样做的目的只是为了在第二点拦截器中通过当前登录账号获取token去认证服务做验证。

**第二点**：第三方系统要在自己系统中添加一个拦截器，拦截器所做的事情就是实时去认证服务验证token 是否过期及账号的登陆状态。

## 2.验证token是否过期

请求地址：http:// 58.218.126.21:29812/server/oauth/check\_token

请求方式：post

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **数据类型** | **是否必须** | **说明** |
| token | 字符串 | 是 | 登陆时获取到令牌（access\_token） |
| 请求示例：[http:// 58.218.126.21:29812/server/oauth/check\_token?token](http://localhost:8080/server/oauth/check_token?token)=xxxxxxxxxxxx  返回结果可以不用关心，只需关心响应状态值是否为200，为200说明token未失效。  返回结果：  {      "aud": [          "sos-resource"      ],      "user\_name": "yanyi",      "scope": [          "read"      ],      "active": **true**,      "exp": 1610607297,      "authorities": [          "ROLE\_USER",          "ROLE\_UNITY"    ],      "jti": "7e865739-aee0-47eb-847b-6c654e32ca00",      "client\_id": "0091cc5cff0b4df8899b1c7c0ac05ac1"  } | | | |

## 3.验证用户是否已退出

请求地址：http:// 58.218.126.21:29812/server/unity/check\_user\_status

请求方式：get

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **数据类型** | **是否必须** | **说明** |
| access\_token | 字符串 | 是 | 登陆时获取到的令牌  access\_token |
| 请求示例：http:// 58.218.126.21:29812/server/unity/check\_user\_status? access\_token=xxxxx  返回结果：  在线：返回 true  离线：返回 false | | | |

## 4.调用认证服务退出接口

请求地址：http:// 58.218.126.21:29812/server/logout? returnUrl=http://xxxxxx

请求方式：response重定向

请求示例：

**在自身系统中的退出接口中添加如下(保险起见需要试连服务地址，看看服务是否可以连通)：**

|  |
| --- |
| try {  URL urlObj = new URL(ssoLogout);  HttpURLConnection oc = (HttpURLConnection) urlObj.openConnection();  oc.setUseCaches(false);  oc.setConnectTimeout(3000); // 设置超时时间  int status = oc.getResponseCode();// 请求状态  if (status != HttpStatus.SC\_OK) {  //这个地址是自身系统退出跳转的页面  response.sendRedirect(SystemEnvironment.getContextPath() + destination);  } else {  response.sendRedirect("http://58.218.126.21:29812/server/logout?returnUrl=" + URLEncoder.encode("http://自己系统的ip:端口号 /登录页地址"))  }  } catch (Exception e) {  //这个地址是自身系统退出跳转的页面  response.sendRedirect(SystemEnvironment.getContextPath() + destination);  } |

## 5.java版本拦截代码示例

（代码仅供参考。）

|  |
| --- |
| package com.seeyon.apps.ext.oauthLogin.filter;  import com.alibaba.fastjson.JSONObject;  import com.seeyon.apps.ext.oauthLogin.util.MapCacheUtil;  import com.seeyon.apps.ext.oauthLogin.util.PropUtils;  import com.seeyon.ctp.common.AppContext;  import com.seeyon.ctp.common.authenticate.domain.User;  import org.apache.commons.logging.Log;  import org.apache.commons.logging.LogFactory;  import org.apache.http.HttpEntity;  import org.apache.http.HttpResponse;  import org.apache.http.HttpStatus;  import org.apache.http.NameValuePair;  import org.apache.http.client.entity.UrlEncodedFormEntity;  import org.apache.http.client.methods.CloseableHttpResponse;  import org.apache.http.client.methods.HttpGet;  import org.apache.http.client.methods.HttpPost;  import org.apache.http.impl.client.CloseableHttpClient;  import org.apache.http.impl.client.HttpClients;  import org.apache.http.message.BasicNameValuePair;  import org.apache.http.util.EntityUtils;  import org.springframework.http.ResponseEntity;  import org.springframework.util.LinkedMultiValueMap;  import org.springframework.util.MultiValueMap;  import org.springframework.web.client.RestClientException;  import org.springframework.web.client.RestTemplate;  import javax.servlet.\*;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import java.io.IOException;  import java.io.InputStream;  import java.net.URLEncoder;  import java.util.ArrayList;  import java.util.List;  import java.util.Map;  public class OauthFilter implements Filter {  private Log log = LogFactory.getLog(OauthFilter.class);  private PropUtils propUtils = new PropUtils();  @Override  public void init(FilterConfig filterConfig) throws ServletException {  }  @Override  public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOException, ServletException {  HttpServletRequest request = (HttpServletRequest) servletRequest;  HttpServletResponse response = (HttpServletResponse) servletResponse;  User user = AppContext.getCurrentUser();  if (null != user) {  //RestTemplate rest = new RestTemplate();可以使用spring中的RestTemplate调用http接口 ，调用方式简单，代码简洁  MultiValueMap map = new LinkedMultiValueMap();  String t = MapCacheUtil.cache.get(user.getLoginName());  if (null != t && !"".equals(t)) {  String m = request.getParameter("method");  HttpPost httpPost = null;  CloseableHttpResponse httpResponse = null;  try (CloseableHttpClient client = HttpClients.createDefault()) {  //验证token 是否失效  // MultiValueMap pmap = new LinkedMultiValueMap();  // pmap.add("token", t);  // ResponseEntity<Object> result = rest.postForEntity(propUtils.getCheckToken(), pmap, Object.class);  List<NameValuePair> list = new ArrayList<NameValuePair>();  list.add(new BasicNameValuePair("token", t));  httpPost = new HttpPost(propUtils.getCheckToken());  httpPost.setEntity(new UrlEncodedFormEntity(list));  //开始验证token是否已经失效  try {  //认证服务连接出现异常，直接放行，不走认证。  httpResponse = client.execute(httpPost);  } catch (IOException e) {  filterChain.doFilter(request, response);  }  int StatusCode = httpResponse.getStatusLine().getStatusCode();  if (StatusCode != HttpStatus.SC\_OK) {//不等于200说明token过期了在服务端不存在了,所以调用接口报错  if (m.equals("logout") || m.equals("index")) {  filterChain.doFilter(request, response);  return;  } else {  String servername = request.getServerName();  response.sendRedirect("http://" + servername + "/seeyon/main.do?method=logout");  }  } else {  map.add(user.getLoginName(), MapCacheUtil.cache.get(user.getLoginName()));  String token = MapCacheUtil.cache.get(user.getLoginName());  // ResponseEntity<String> responseEntity = rest.getForEntity(propUtils.getCheckUserStatus() + "?access\_token=" + token, String.class);  //验证账号是否已经离线  HttpGet httpGet = new HttpGet(propUtils.getCheckUserStatus() + "?access\_token=" + token);  httpResponse = client.execute(httpGet);  if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC\_OK) {  String resultString = EntityUtils.toString(httpResponse.getEntity(), "utf-8").replaceAll(" ", "");  if ("true".equals(resultString)) {  filterChain.doFilter(request, response);  } else {  if (m.equals("logout") || m.equals("index")) {  filterChain.doFilter(request, response);  return;  } else {  String servername = request.getServerName();  response.sendRedirect("http://" + servername + "/seeyon/main.do?method=logout");  }  }  }  }  } catch (RestClientException e) {  log.error("调用验证token 接口出错了：" + e.getMessage());  } finally {  if(null != httpResponse){  httpResponse.close();  }  }  } else {  filterChain.doFilter(request, response);  }  } else {  filterChain.doFilter(request, response);  }  }  @Override  public void destroy() { }  } |