**第三方调用示例说明**

2020-11-12

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。

认证地址：

<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://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 | | | |

上面拿到了授权码，接着就是使用授权码去获取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://localhost:8080/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。** | | | |

获取用户信息

<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://localhost:8080/server/unity/user_info?access_token=accessToken>  成功返回  {     **"guid"**:**"-3394461689692140820"**,     **"archived"**:**false**,     **"username"**:**"yanyi"**,     **"phone"**:**"18136001664"**,     **"email"**:**"121212"**,     **"privileges"**:[         **"UNITY"**     ] }  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：

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://localhost:8080/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://localhost:8080/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://localhost:8080/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);  }  } |