

Implementing OAuth (Open Authorization) In Java Web Application

OAuth is a protocol using which a third party application can access a part of user's account information like name, age, dob, friend list etc. after authorization by the user.

Players involved in OAuth

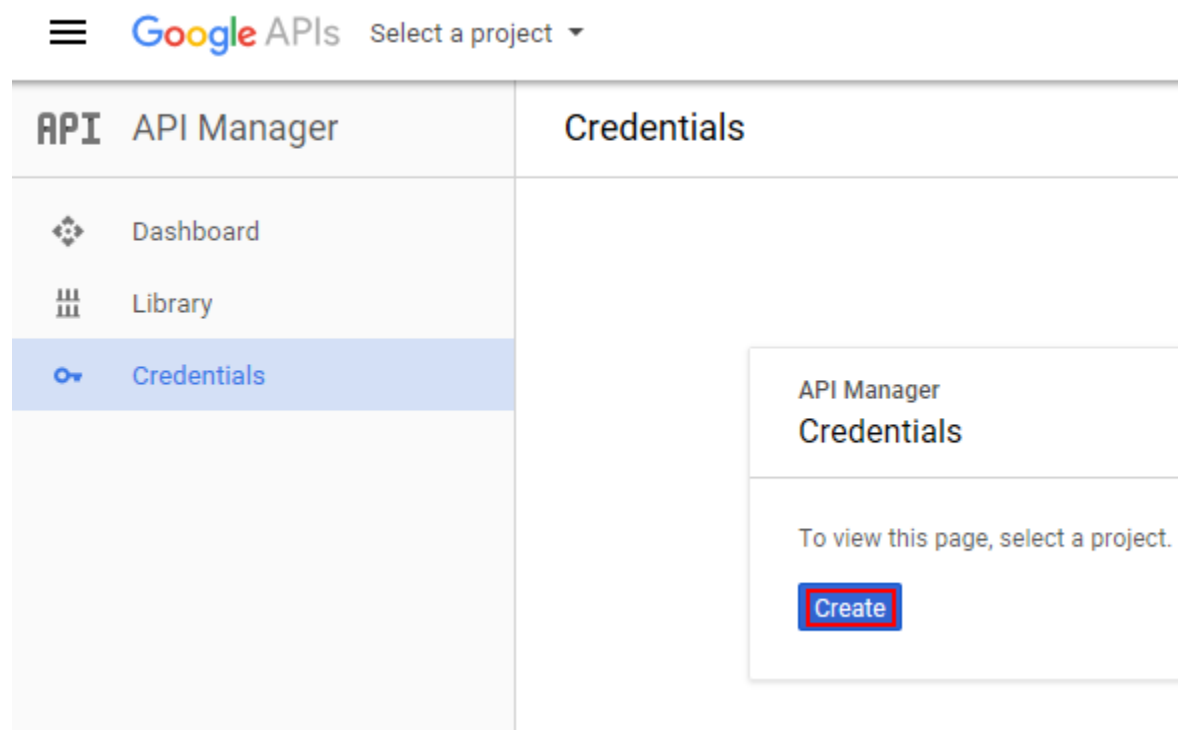
1. Service provider – an online service provider with which user has her/his account information. E.g. Google, Facebook, Github etc.
2. Third-party application (client application) – application which wants to access a part of user's account information like name or email address etc.
3. User - who authorizes a third-party application to access a part of his account information.

Let us take an example to see the end-to-end process where a third party application (client application) - "OAuth client" wants to access name and email address of a user from Google account.

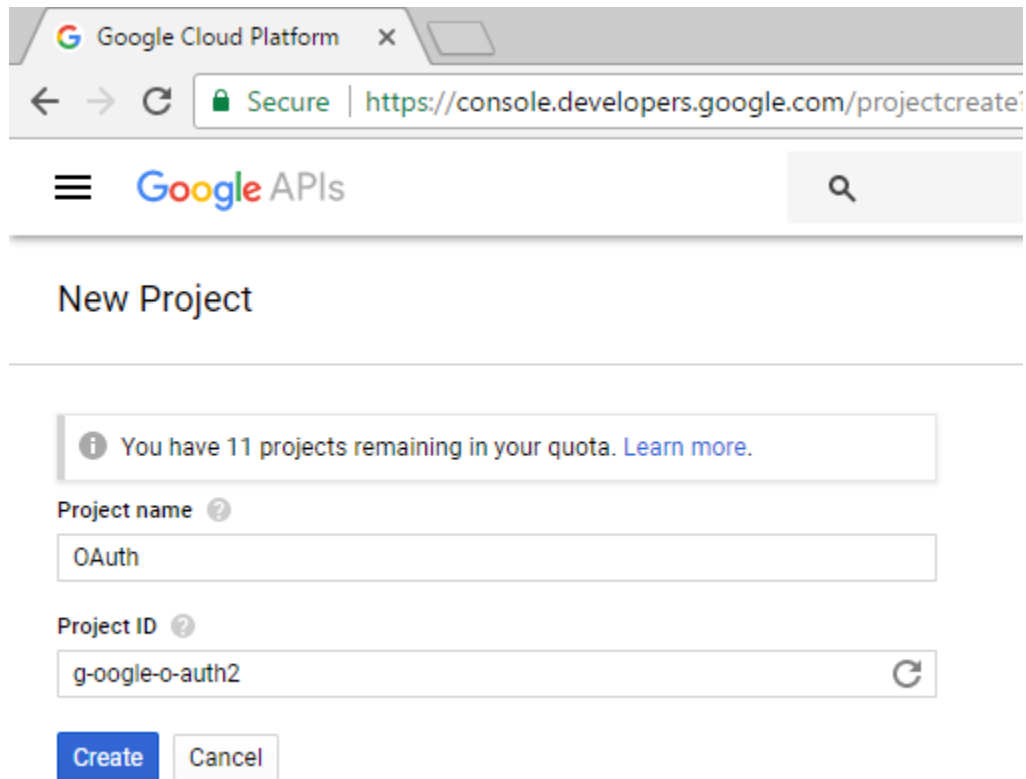
Step 1 – Register third-party application "OAuth client" with Google.

To register an application we need to create a project in google developer console

<https://console.developers.google.com/projectselector/apis/dashboard?authuser=1&organizationId=0>



Project name can be anything but project id has to be unique



Google Cloud Platform x

Secure | https://console.developers.google.com/projectcreate

Google APIs

New Project

i You have 11 projects remaining in your quota. [Learn more.](#)

Project name ?

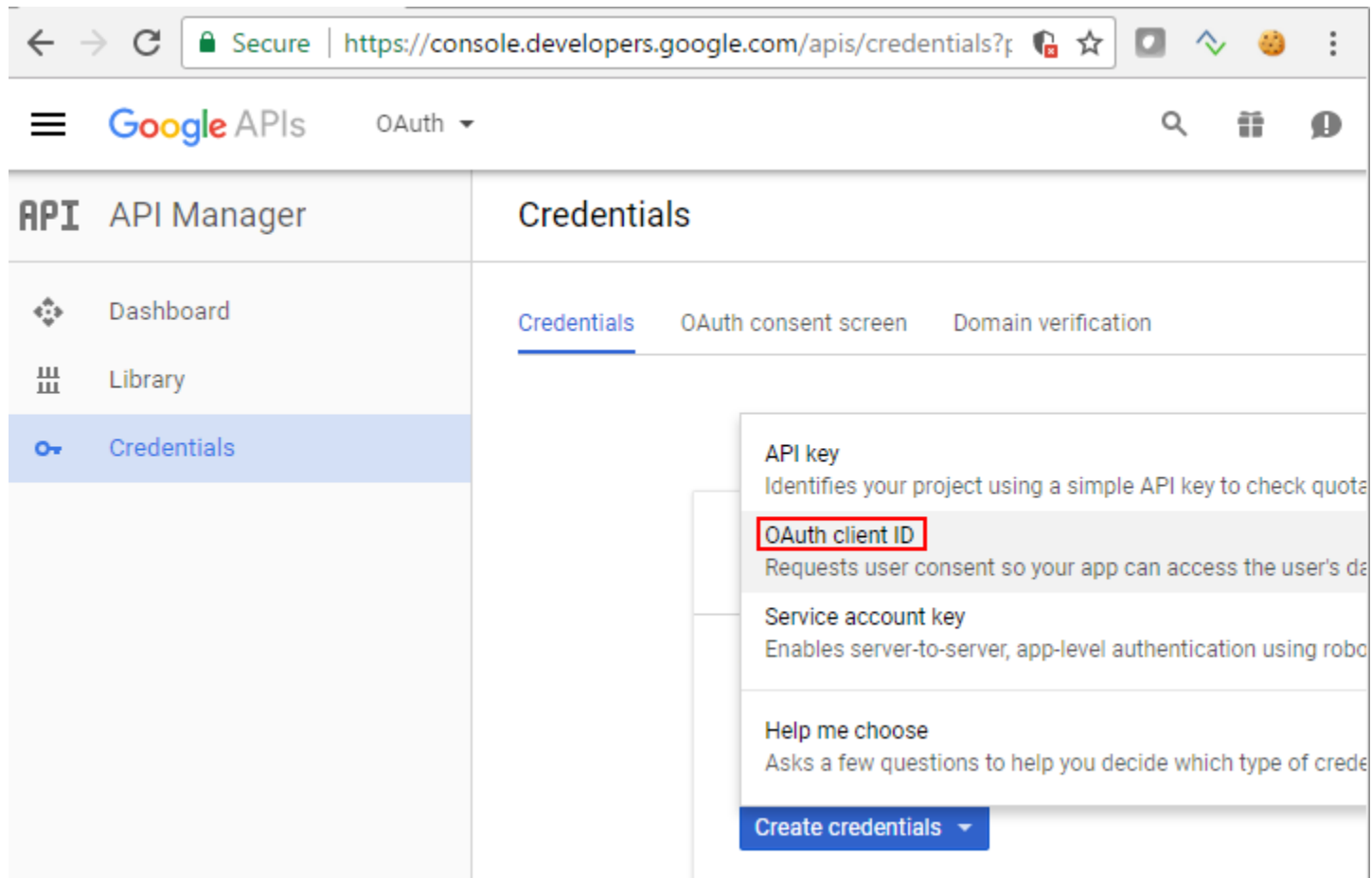
OAuth

Project ID ?

g-oogle-o-auth2 ↻

Create Cancel

Configure the **consent screen** (screen presented to the user asking for permission to access his account information), **name of web application** and **redirect URI** (URI to which user will be redirected to after her/his authorization)



Configure the [consent screen](#)

Consent screen - OAuth

Secure | https://console.developers.google.com/apis/credentials/consent?createCl

Google APIs

OAuth

API

API Manager

Dashboard

Library

Credentials

Credentials

OAuth consent screen

Domain verification

Email address

@gmail.com

Product name shown to users

OAuth - Client

Homepage URL (Optional)

https:// or http://

Product logo URL (Optional)

http://www.example.com/logo.png

This is how your logo will look to end users
Max size: 120x120 px

Privacy policy URL

Optional until you deploy your app

←

→

↻

Secure | https://console.developers.google.com/apis/credentials/c

Google APIs

OAuth ▾

API

API Manager

Dashboard

Library

Credentials

←

Create client ID

Application type

☒ Web application

☐ Android [Learn more](#)

☐ Chrome App [Learn more](#)

☐ iOS [Learn more](#)

☐ PlayStation 4

☐ Other

Name

OAuth

Restrictions

Enter JavaScript origins, redirect URIs, or both

Authorized JavaScript origins

For use with requests from a browser. This is the origin l
(http://*.example.com) or a path (http://example.com/sl
it in the origin URI.

http://www.example.com

Authorized redirect URIs

For use with requests from a web server. This is the path
have authenticated with Google. The path will be append
protocol. Cannot contain URL fragments or relative path:

http://127.0.0.1:8080/OAuth/redirect

Credentials to access user information

←

→

↻

Secure | https://console.developers.google.com/apis/credentials?r

Google APIs

OAuth ▾

API

API Manager

Dashboard

Library

Credentials

Credentials

Credentials OAuth consent screen Domain verification

Create credentials ▾Delete

Create credentials to access your enabled APIs. [Refer to the API documentation](#) for c

OAuth 2.0 client IDs

<input type="checkbox"/>	Name	Creation date ▾	Type	Client ID
<input type="checkbox"/>	OAuth	Jul 25, 2017	Web application	984169855535-rmsfbv11ikina3hrb3r

Once done with above steps you can see your client id, client secret and redirect URI. Make sure to keep your client secret confidential else someone can impersonate your application.

← → ↻ Secure | https://console.developers.google.com/apis/credentials/

Google APIs OAuth ▾

API API Manager

← Client ...cation [DOWNLOAD JSON](#)

Client ID	984169855535-rmsfbv11ikina3hrb3n85h0nt93hc
Client secret	[REDACTED]
Creation date	Jul 25, 2017, 5:56:59 PM

Name

OAuth

Restrictions

Enter JavaScript origins, redirect URIs, or both

Authorized JavaScript origins
For use with requests from a browser. This is the origin URI of the client appl (http://*.example.com) or a path (http://example.com/subdir). If you're using it in the origin URI.

http://www.example.com

Authorized redirect URIs
For use with requests from a web server. This is the path in your application have authenticated with Google. The path will be appended with the authoriz protocol. Cannot contain URL fragments or relative paths. Cannot be a public

http://127.0.0.1:8080/OAuth/redirect

http://www.example.com/oauth2callback

[Save](#) [Cancel](#)

Step 2 – Go to API dashboard and enable API access

Dashboard - OAuth x

Secure | <https://console.developers.google.com/apis/dashboard?project=g-oog->

Google APIs OAuth ▾

API Dashboard [+ ENABLE APIS AND SERVICES](#)

Enabled APIs and services
Some APIs and services are enabled automatically

Activity for the last hour 1 hour 6 hours 12 hours 1 day

Traffic

Requests/sec

Errors

Percent of requests


API Library - OAuth x

Secure | <https://console.developers.google.com/apis/library?project=g-oog->

Google APIs OAuth ▾


API Library

More

 **Mobile APIs**

- [Google Cloud Messaging](#)
- [Google Play Game Services](#)
- [Google Play Developer API](#)
- [Google Places API for Android](#)

More

 **Social APIs**

- [Google+ API](#)
- [Blogger API](#)
- [Google+ Pages API](#)
- [Google+ Domains API](#)

APIs & services - OAuth

Secure | https://console.developers.google.com/apis/api/plus.google

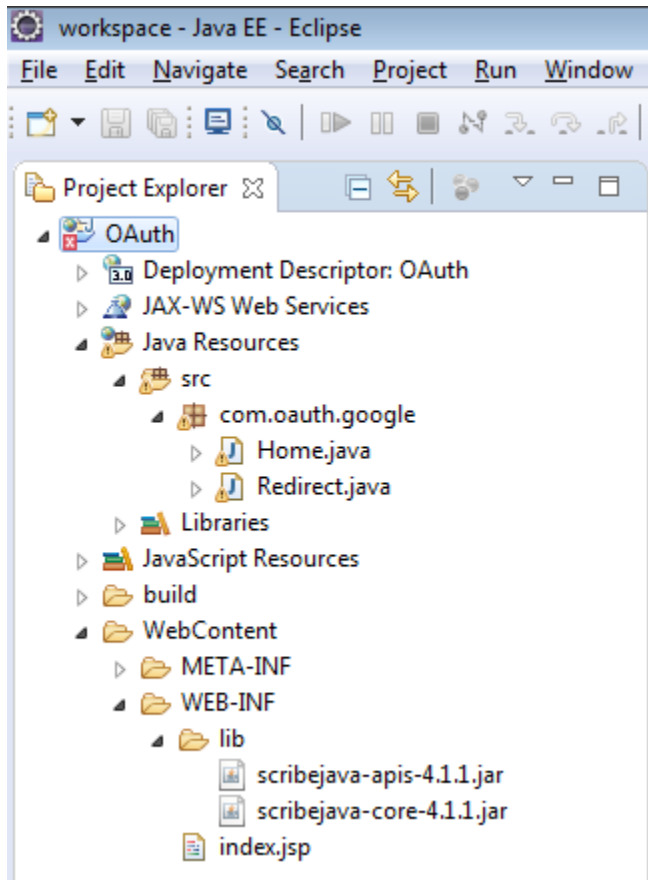
Google APIs

OAuth

API	<div><div>←</div><div>Google+ API</div><div>■ DISABLE</div></div>
<div><div>⚙</div><div>Overview</div><div>Quotas</div></div>	
<div><div>☰</div><div>⚙</div></div>	
	About this API

Step 3 – Create a web application to access a user's basic info (name and email address) and display it back to the end user. I am using scribe java library [scribe java library](#) to implement OAuth.

Project structure



Home.java

```

1 package com.oauth.google;
2
3 import java.io.IOException;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20 @WebServlet("/home")
21 public class Home extends HttpServlet {
22
23     protected void doGet(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25
26         final String cliendId = "984169855535-rmsfbv11ikina3hrb3n85h0nt93hdb2t.apps.googleusercontent.com";
27         final String clientSecret = " ";
28         final String redirectUri = "http://127.0.0.1:8080/OAuth/redirect";
29         final String scope = "https://www.googleapis.com/auth/userinfo.email https://www.googleapis.com/auth/plus.login";
30         final String secretState = new BigInteger(130, new SecureRandom()).toString(32);
31
32         // Construct Google authorization URL
33         OAuth20Service authUrl = new ServiceBuilder().apiKey(cliendId).apiSecret(clientSecret).scope(scope).state(secretState).callback(redirectUri).build(GoogleApi20.instance());
34
35         // Add additional parameters (--NOT MANDATORY--)
36         final Map<String, String> additionalParams = new HashMap<>();
37         additionalParams.put("access_type", "offline");
38         additionalParams.put("prompt", "consent");
39
40         // Obtain authorization URL
41         String authorizationUrl = authUrl.getAuthorizationUrl(additionalParams);
42         System.out.println("Authorization Url - " + authorizationUrl + "\n");
43
44         // Add authUrl to session variable to be used in Redirect.java
45         HttpSession session = request.getSession();
46         session.setAttribute("authUrl", authUrl);
47         response.sendRedirect(authorizationUrl);
48     }
49 }
50

```

Explanation

[cliendId](#) – id that we got at the time of registering our application with the service provider

[cliendSecret](#) – secret that we got at the time of registering our application with the service provider

[redirectURI](#) – redirect URI that we have given at the time of registering our application

scope – scope defines which part of the account information we want to access. E.g., email address, name and age range etc. Try [Google Playground](#) to learn more about scopes

secretState – a random string to prevent [CSRF attack](#), it will be verified later to ensure that the user actually wants to perform the intended operation.

[access_type](#) = “offline”. It means you will get a refresh token from Google, which can be exchanged to get an access token. This is used to access user’s information when he is offline.

authURL – construct the Google authorization URL to redirect to when the user tries to access the home page of the application

e.g. -

https://accounts.google.com/o/OAuth2/auth?access_type=offline&prompt=consent&response_type=code&client_id=984169855535-rmsfbv11ikina3hrb3n85h0nt93hdb2t.apps.googleusercontent.com&redirect_uri=http%3A%2F%2F127.0.0.1%3A8080%2FOAuth2Fredirect&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fplus.login&state=ou0af6lnfdnhu1ssd7na1onvqm (URL encoded)

authURL session variable – create session variable to be accessed in the redirect page (Redirect.java)

Note – Instead of doing all that above you can simplify it by hardcoding the authorization URL and then redirecting the user to that URL as given below. However, hardcoding is not the recommended way to program.

String authorizationURL =

[“https://accounts.google.com/o/OAuth2/auth?access_type=offline&prompt=consent&response_type=code&client_id=984169855535-rmsfbv11ikina3hrb3n85h0nt93hdb2t.apps.googleusercontent.com&redirect_uri=http%3A%2F%2F127.0.0.1%3A8080%2FOAuth2Fredirect&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fplus.login&state=ou0af6lnfdnhu1ssd7na1onvqm”](https://accounts.google.com/o/OAuth2/auth?access_type=offline&prompt=consent&response_type=code&client_id=984169855535-rmsfbv11ikina3hrb3n85h0nt93hdb2t.apps.googleusercontent.com&redirect_uri=http%3A%2F%2F127.0.0.1%3A8080%2FOAuth2Fredirect&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fplus.login&state=ou0af6lnfdnhu1ssd7na1onvqm);

response.sendRedirect(authorizationURL);

Redirect.java

Redirect.java

```
1 package com.oauth.google;
2
3 import java.io.IOException;
4
22 @WebServlet(urlPatterns = { "/redirect" }, asyncSupported = true)
24 public class Redirect extends HttpServlet
25 {
26     private static final String resourceUrl = "https://www.googleapis.com/oauth2/v2/userinfo";
27
28     protected void doGet(HttpServletRequest request,
29         HttpServletResponse response)
30         throws ServletException, IOException
31     {
32         System.out.println("=== in the redirect page ===");
33         String code = request.getParameter("code");
34         System.out.println("code - " + code + "\n");
35
36         String userInformation="{ }";
37         String responseCode="000";
```

```

38
39 // Check if the user has given authorization. Code parameter will not be null.
40 if (code != null)
41 {
42     HttpSession session = request.getSession();
43     OAuth20Service authUrl=(OAuth20Service) session.getAttribute("authUrl");
44
45     try
46     {
47         // Get the access token by sending code in the request
48         OAuth2AccessToken accessToken = authUrl.getAccessToken(code);
49         System.out.println("accessToken - " + accessToken+"\n");
50
51         // Sign the request using access token
52         final OAuthRequest authRequest = new OAuthRequest(Verb.GET, authUrl);
53         authUrl.signRequest(accessToken, authRequest);
54
55         //Get user information
56         final Response resp = authUrl.execute(authRequest);
57         userInformation=resp.getBody();
58         responseCode=Integer.toString(resp.getStatusCode());
59
60         System.out.println("response code - "+responseCode);
61         System.out.println("response body \n"+userInformation);
62     }
63     catch (Exception e)
64     {
65         System.out.println("oops! an error occured");
66     }
67 }
68 request.setAttribute("responseCode", responseCode);
69 request.setAttribute("userInformation", userInformation);
70
71 RequestDispatcher rd = request
72     .getRequestDispatcher("/WEB-INF/index.jsp");
73 rd.forward(request, response);
74 }

```

Explanation

Code - a unique code that Google has sent, take it from the request parameter
 e.g. - 4/sdfJnfetXGuylu15sdfIE37P3xsdfPVW5sdfsdsdf

authURL session variable - take session variable from the session that was created in Home.java

accessToken - pass the code in the request to get the access token

e.g. - GoogleToken{**access_token**=ya29.Glu8BAJrFkW-xGEaKG66sdfsq3lgozYUQsZwSoaYn5mPPEsdfor04eOnCzr3rR3fgldfkWd2EtRhqYMRVq-2DVB6puJNAIW3yicxZDe2uKsbpFBwBVCRFSIVCQH02, **token_type**=Bearer, **expires_in**=3600, **refresh_token**=1/BQB10JcUU440TMJAUK4L8rTt07V-RQdfRCVXKtkEi1Y, scope=null, open_id_token=eyJhbGciOiJSUzI1NiIsImtpZCI6ImM3OTc2ZTVmYTk0MjM5ZmFlOWI4NjY3MTgxMDIxMWNlZWQ0NjhhMTYifQ.eyJhenAiOiI5OsfDQxNjk4NTU1MzUtdm1zZmJ2MTFpa2luYTNocmIzbg1aDBudDkzaGRiMnQuYXBwcy5nb29nbGV1c2VyY29udGVudC5jb20iLCJhdWQiOiI5ODQxNjk4NTU1MzUtdm1zZmJ2MTFpa2luYTNocmIzbg1aDBudDkzaGRiMnQuYXBwcy5nb29nbGV1c2VyY29udGVudC5jb20iLCJzdWIiOiIxMDk}

authRequest - create open-auth request to access users information and specify the resource URL and the method. Pass the access-token to get the user's information. Without valid access-token, user information cannot be accessed.

All the hard work done, it's time to access user's information

<http://127.0.0.1:8080/OAuth/home>

When a user tries to access above URL he/she is redirected to the Gmail login page. A status code of 302 is set in the response header along with the redirect URL.

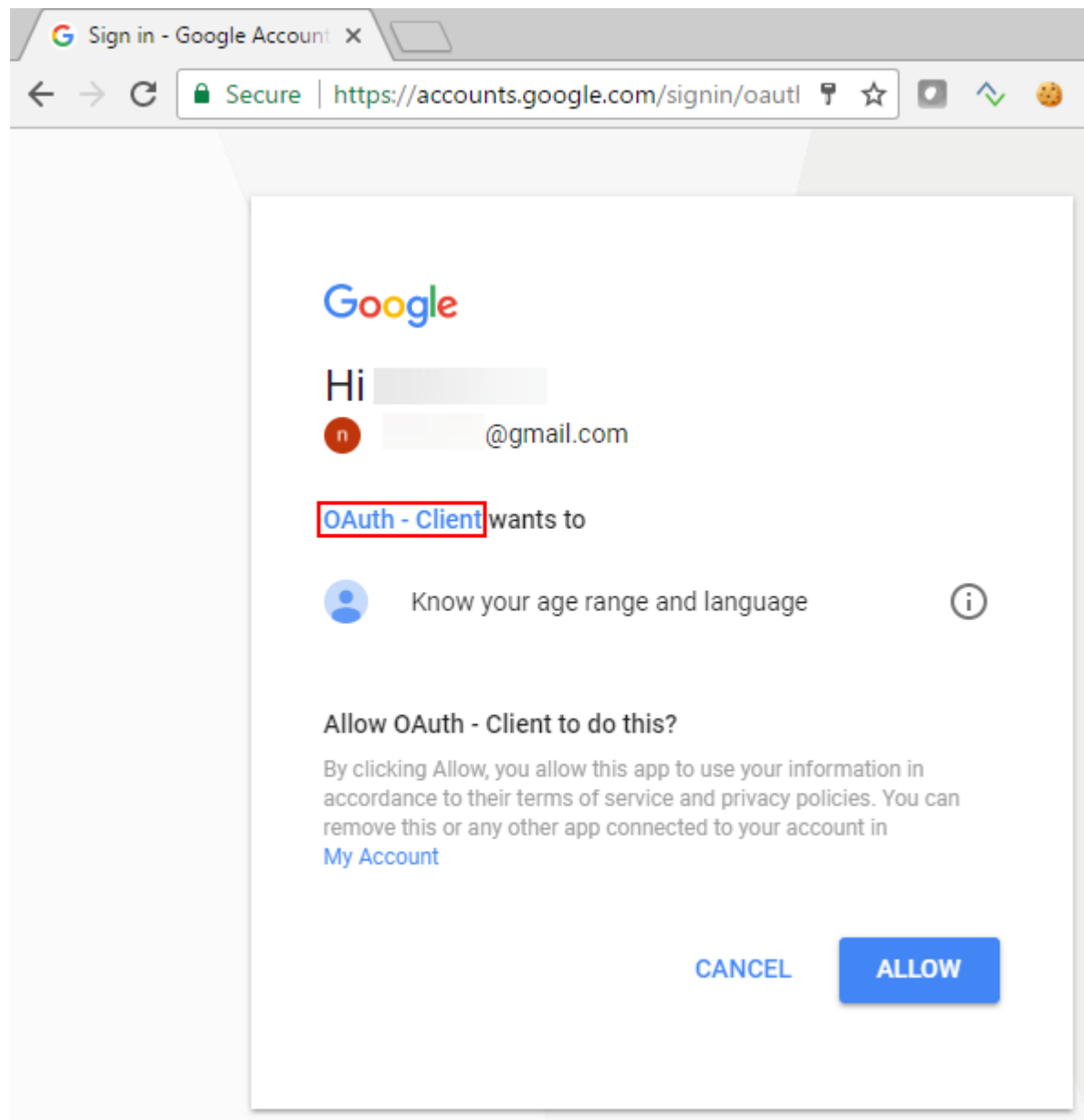
The screenshot shows a web browser window with the Google Sign in page. The address bar displays the URL `https://accounts.google.com/signin/oautil`. The page content includes the Google logo, the text "Sign in to continue to OAuth - Client", an input field for "Email or phone", and a link for "Forgot email?".

Below the page content, the browser's developer tools are open, showing the Network tab. A list of requests is displayed, with the first request, "home", selected. The details for this request are shown on the right:

- General**
 - Request URL: `http://127.0.0.1:8080/OAuth/home`
 - Request Method: GET
 - Status Code: 302
 - Remote Address: 127.0.0.1:8080
 - Referrer Policy: no-referrer-when-downgrade
- Response Headers**
 - Content-Length: 0
 - Date: Mon, 04 Sep 2017 10:01:03 GMT
 - Location: `https://accounts.google.com/o/oauth2/auth?access_type=offline&prompt=consent&response_type=code`

The status bar at the bottom indicates 16 requests, 310 KB transferred, and a load time of 874 ms.

Login with Gmail credentials and provide/deny authorization below.



You can access users email id and name. Congratulations!!

