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

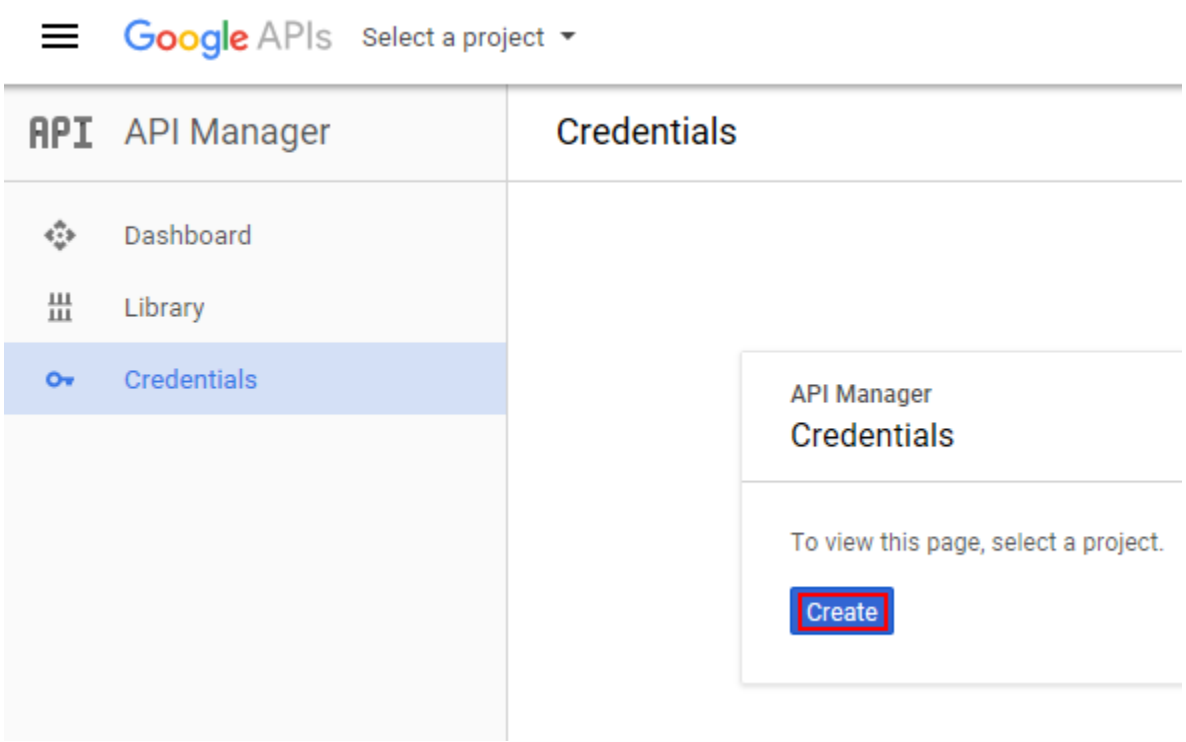
1. Service provider – an online service provider with which user has her/his account information. E.g. Google, Facebook, Github etc.
2. Third-party application (client application) – application which wants to access a part of user's account information like name or email address etc.
3. User - who authorizes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

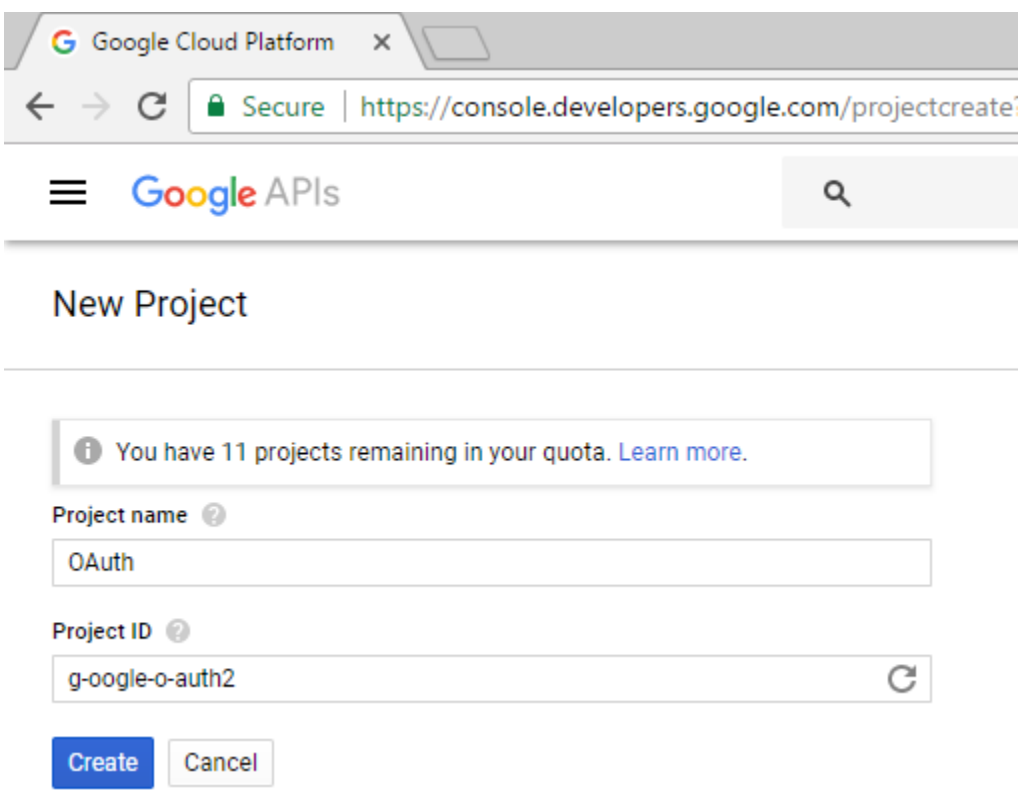
Step 1 – Register third-party application "OAuth client" with Google.

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

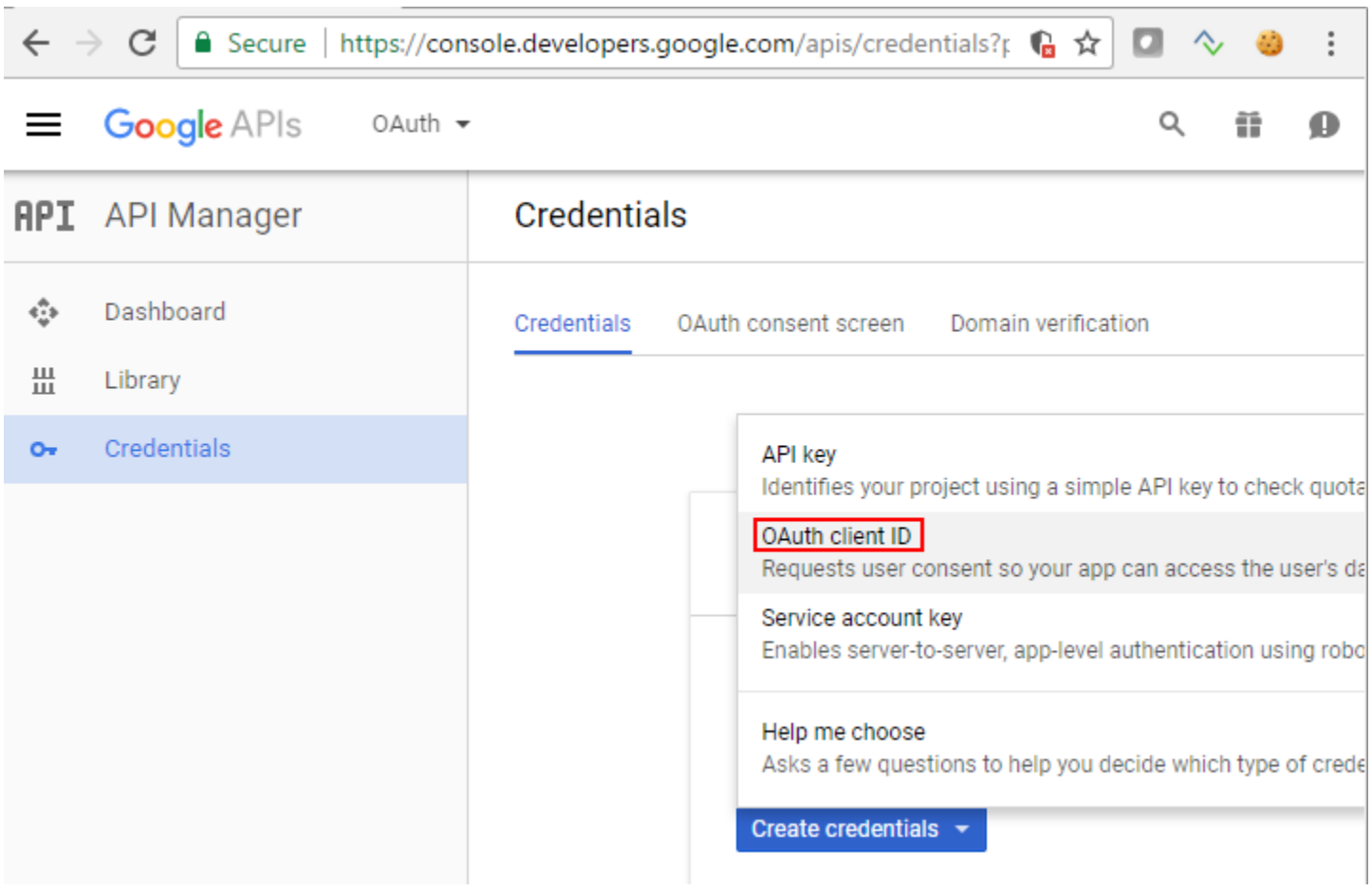
https://console.developers.google.com/projectselector/apis/dashboard?authuser=1&organizationId=0



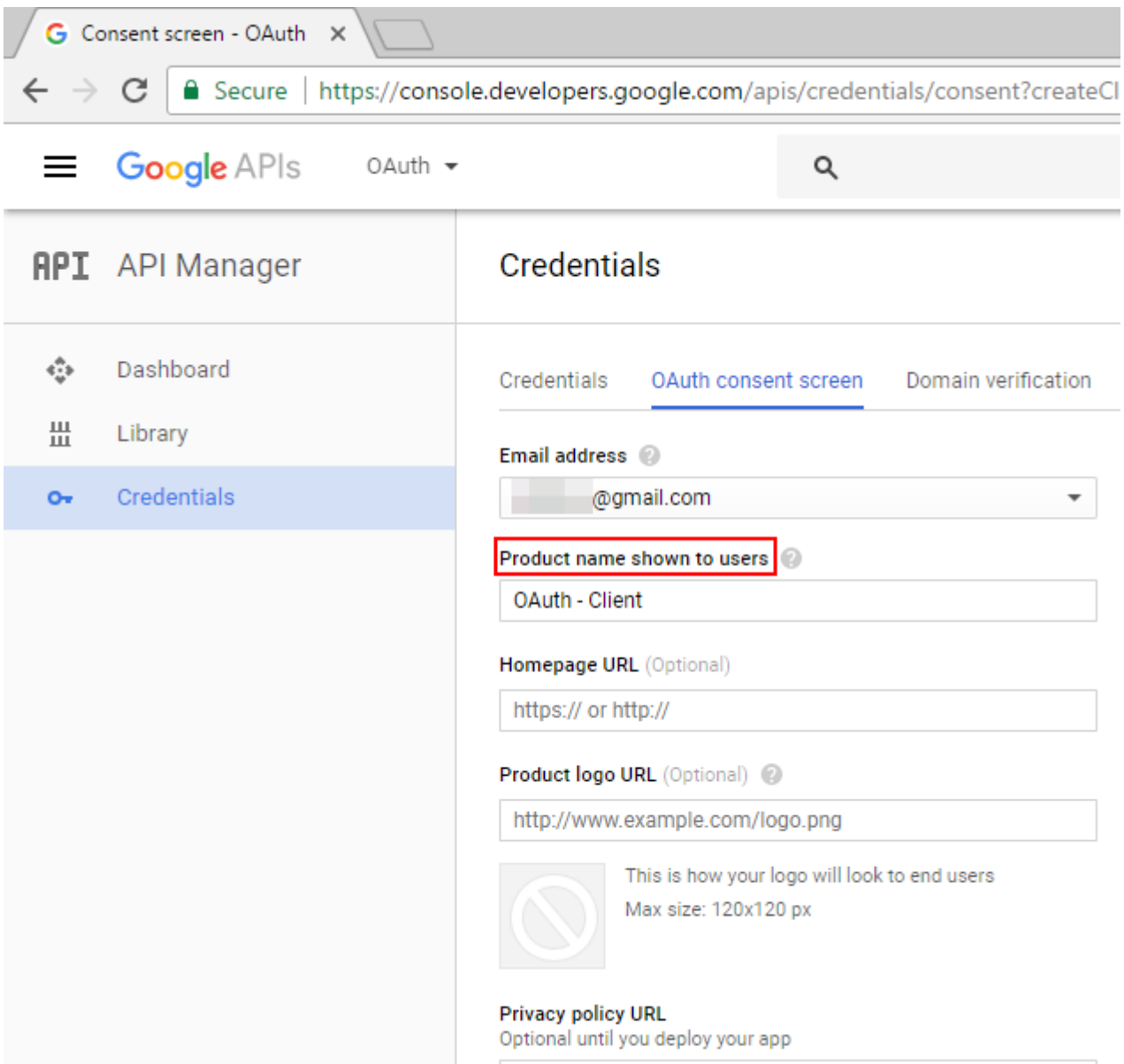
Project name can be anything but project id has to be unique

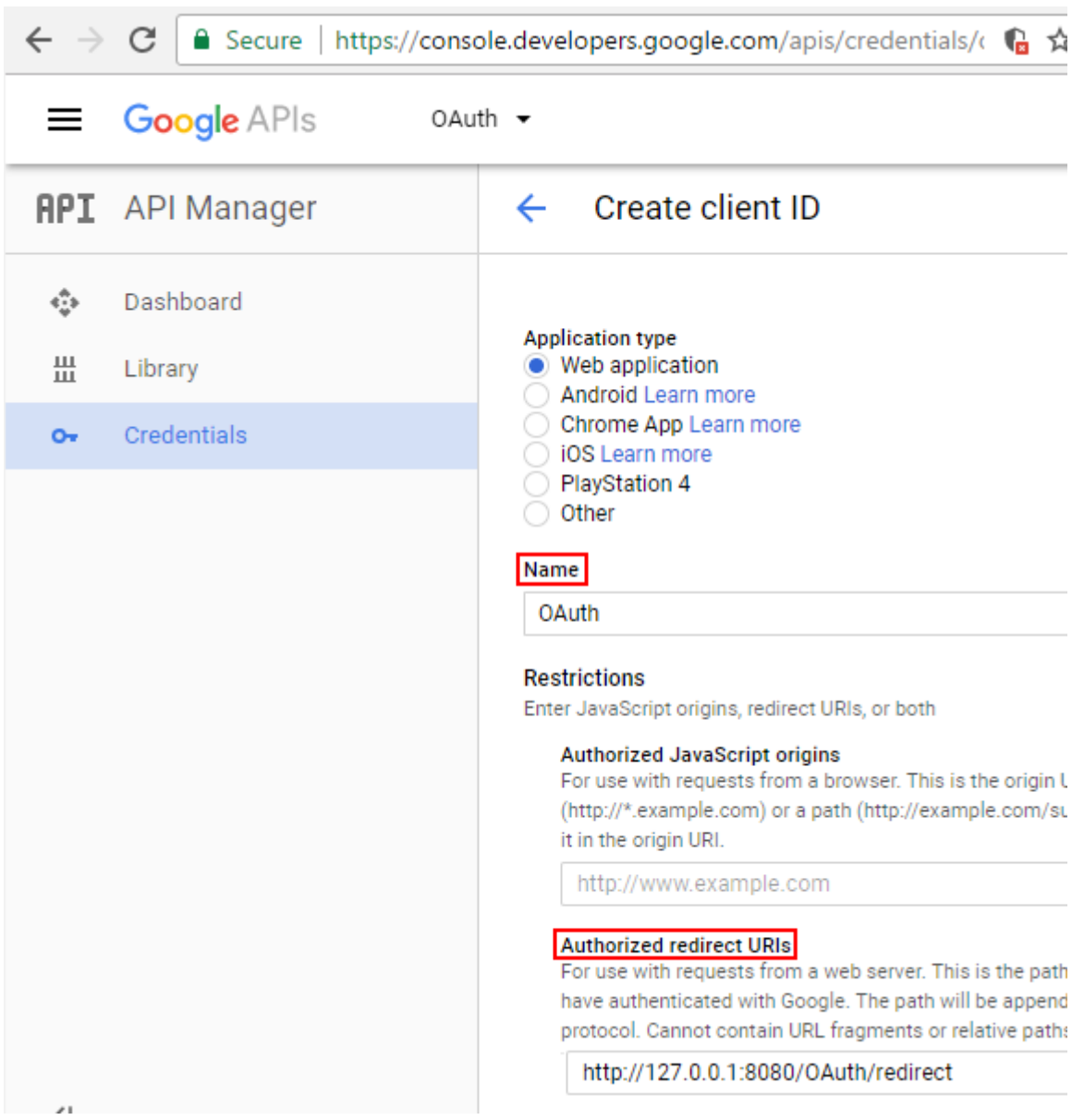


Configure the **consent screen** (screen presented to the user asking for permission to access his account information), **name of web application** and **redirect URI** (URI to which user will be redirected to after her/his authorization)

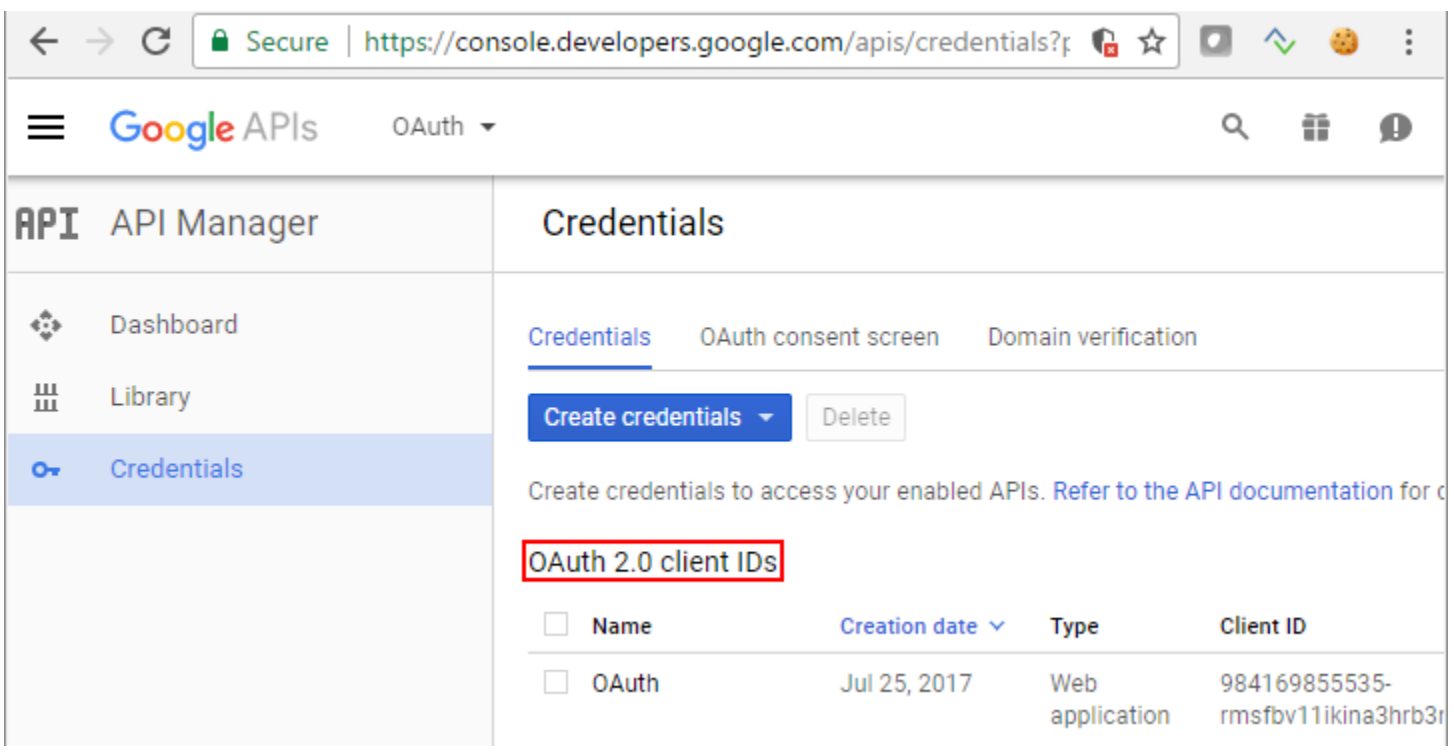


Configure the consent screen

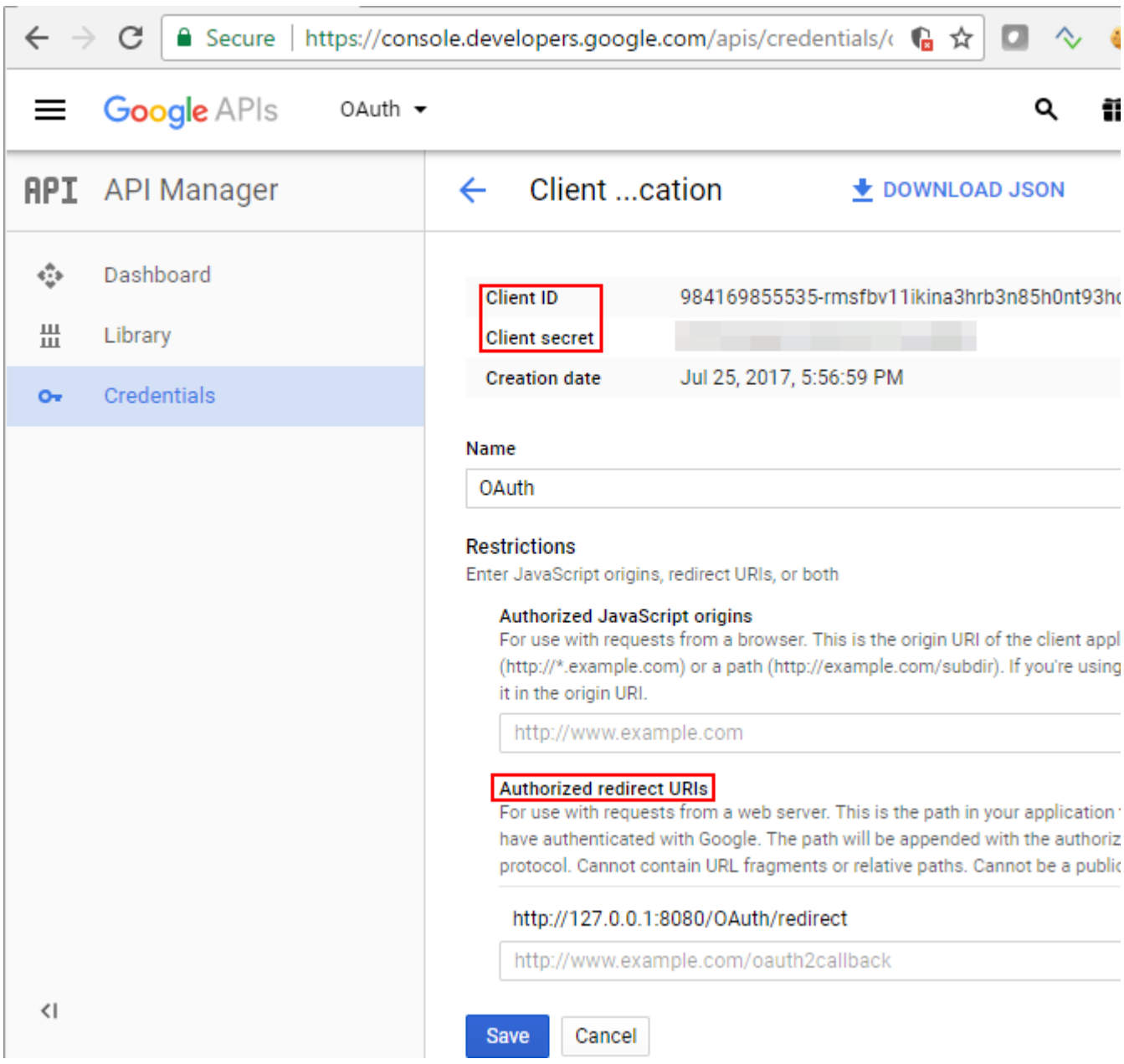




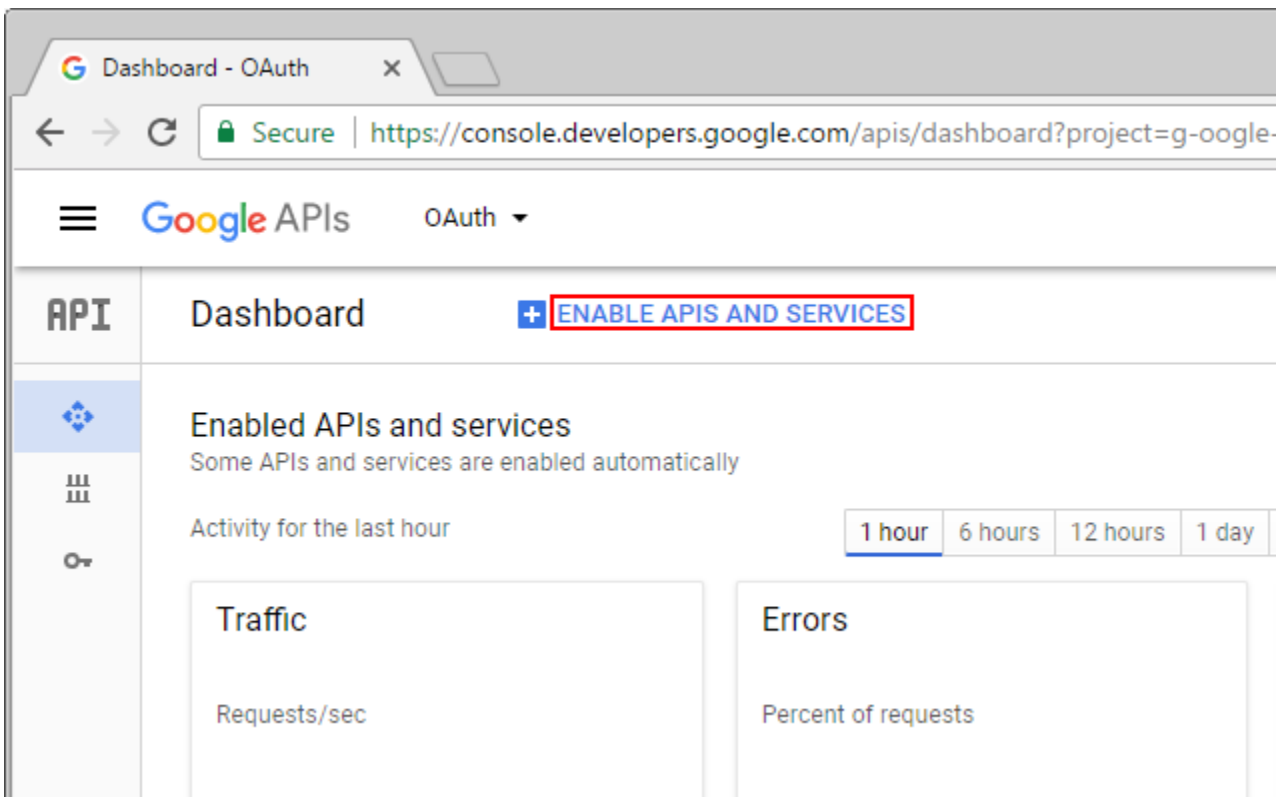
Credentials to access user information

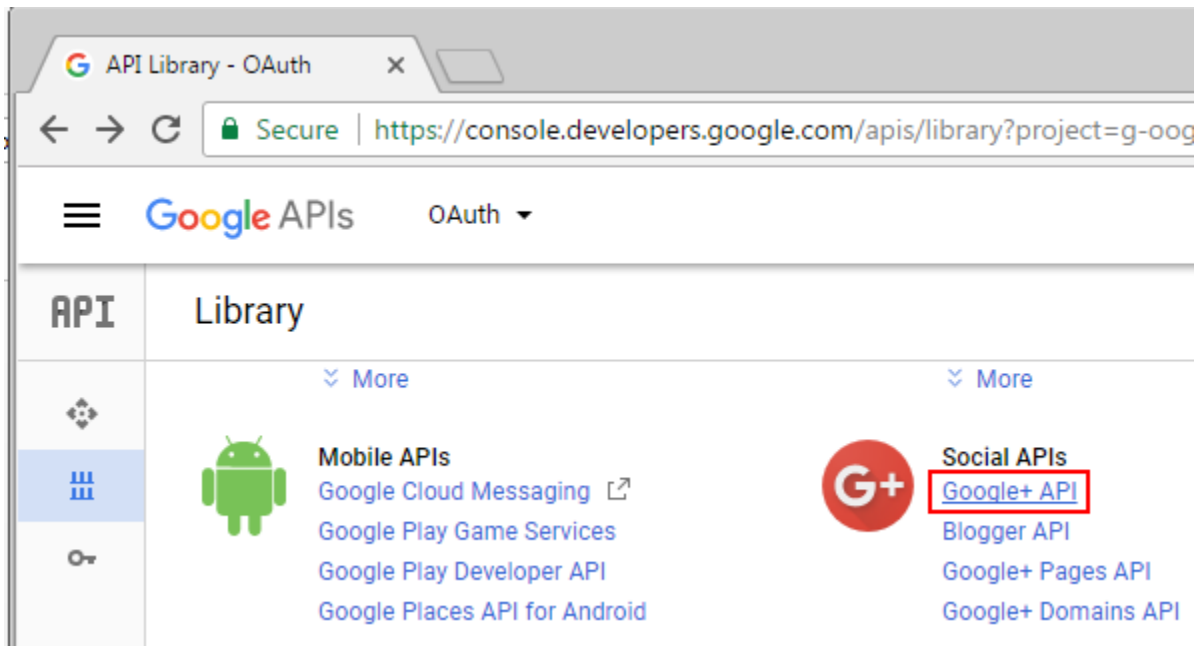


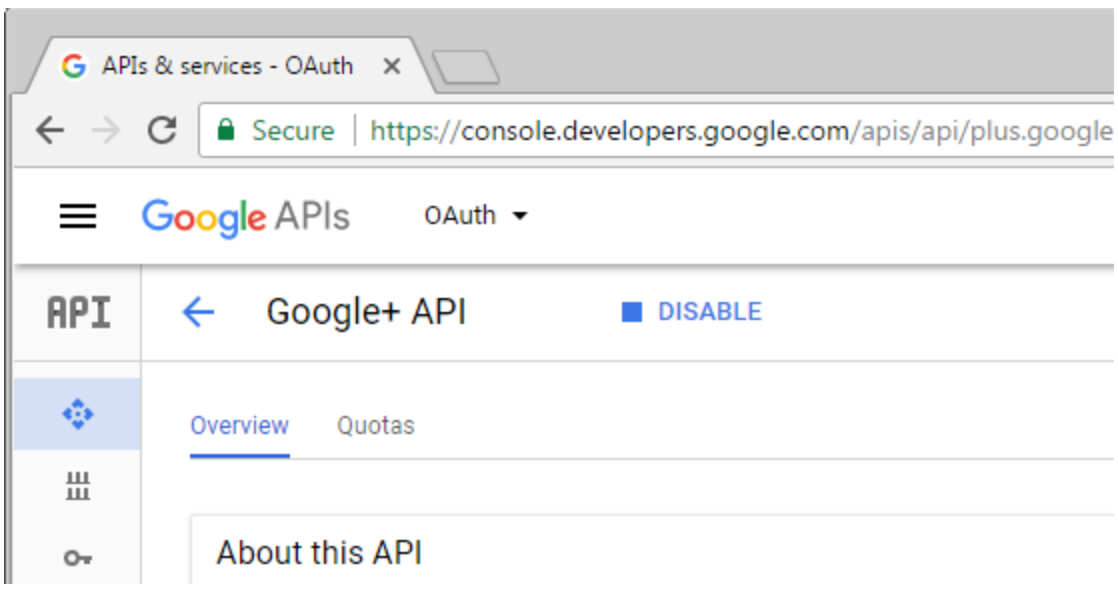
Once done with above steps you can see your client id, client secret and redirect URI. Make sure to keep your client secret confidential else someone can impersonate your application.



Step 2 – Go to API dashboard and enable API access

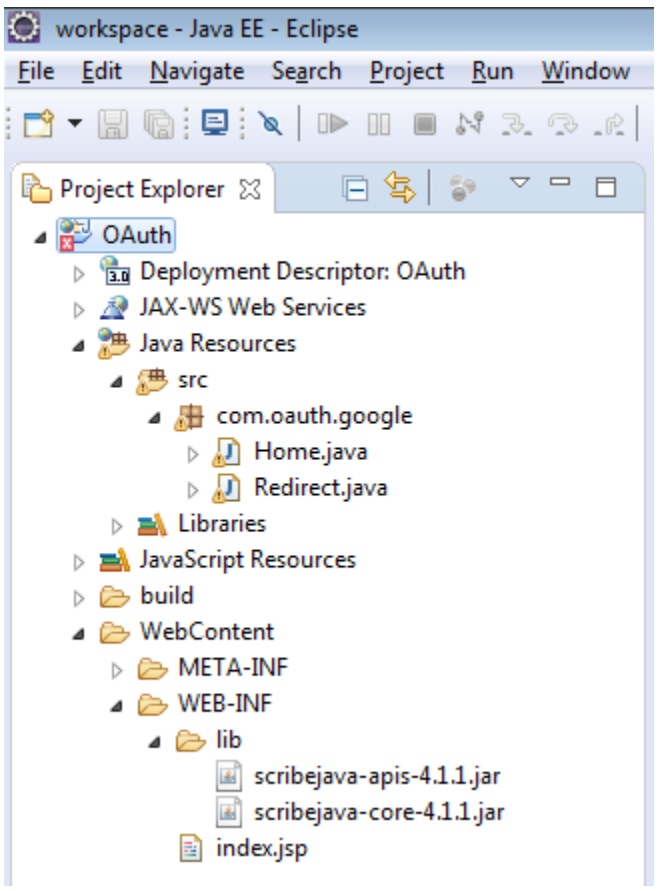





APIs & services - OAuth
Secure
https://console.developers.google.com/apis/api/plus.google
Google APIs
OAuth
API
Google+ API
DISABLE
Overview
Quotas
About this API

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

```
Home.java
1  package com.oauth.google;
2
3⊕ import java.io.IOException;
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)
34                .state(secretState).callback(redirectUri).build(GoogleApi20.instance());
35
36          // Add additional parameters (--NOT MANDATORY--)
37          final Map<String, String> additionalParams = new HashMap<>();
38          additionalParams.put("access_type", "offline");
39          additionalParams.put("prompt", "consent");
40
41          // Obtain authorization URL
42          String authorizationUrl = authUrl.getAuthorizationUrl(additionalParams);
43          System.out.println("Authorization Url - "+authorizationUrl + "\n");
44
45          //Add authUrl to session variable to be used in Redirect.java
46          HttpSession session = request.getSession();
47          session.setAttribute("authUrl", authUrl);
48          response.sendRedirect(authorizationUrl);
49      }
50
```

Explanation

clinetId – id that we got at the time of registering our application with the service provider

clinetSecret – 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%2FOAuth%2Fredirect&**scope**=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.e

mail%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%2FOAuth%2Fredirect&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fplus.login&state=ou0af6lnfdnhu1ssd7na1onvqm";

response.sendRedirect(authorizationURL);

Redirct.java

```java
Redirect.java
1  package com.oauth.google;
2
3⊕ import java.io.IOException;
22
23 @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, resourceUrl);
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.getCode());
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

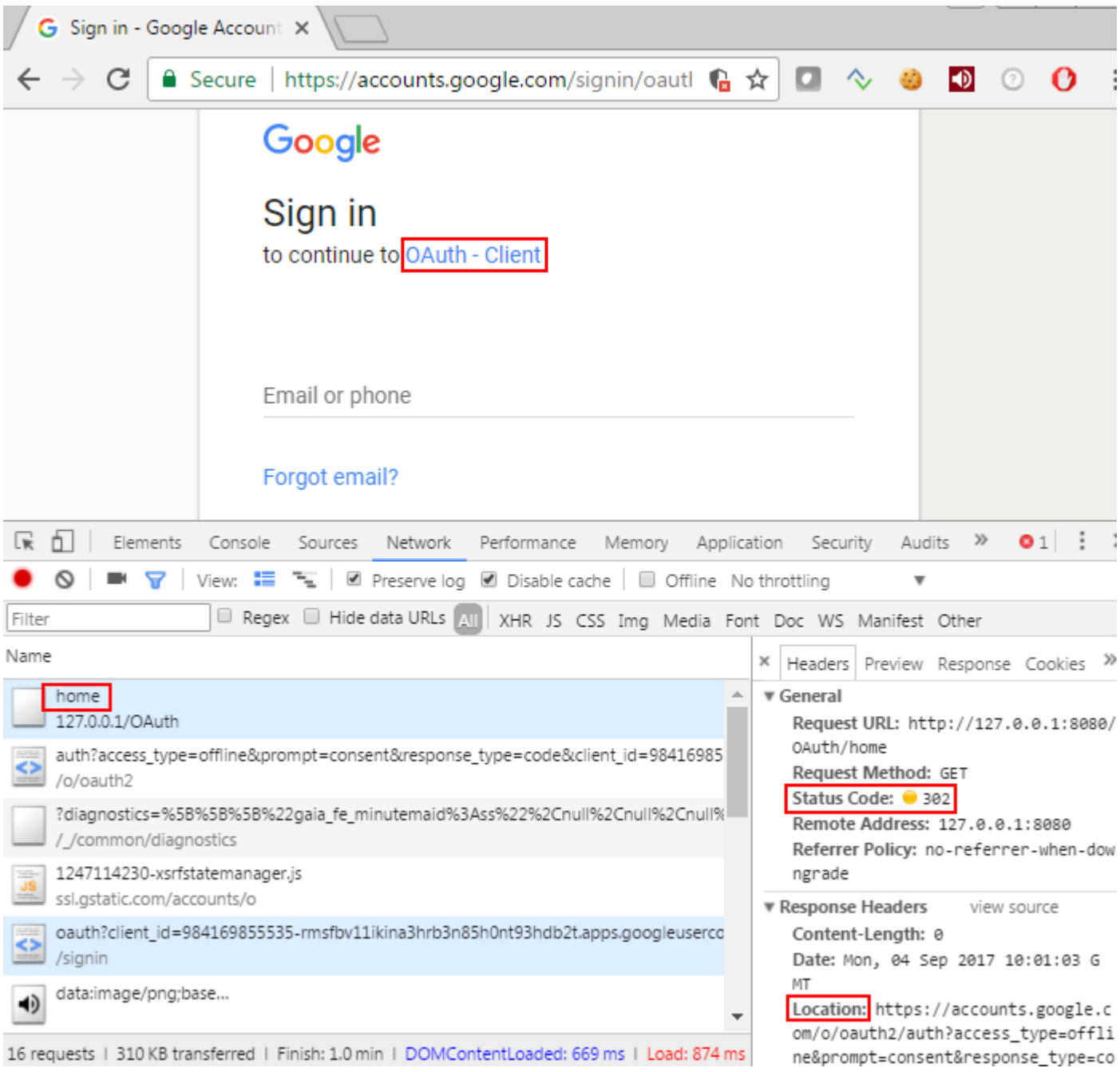
e.g. - GoogleToken{**access_token**=ya29.Glu8BAJrFxW-xGEaKG66sdfsq3lgozYUQsZwSoaYn5mPPEsdfor04eOnCzr3rR3fgIdfkwD2EtRhqYMRVq-2DVB6puJNAlW3yicxZDe2uKsbpFBwBVCRFSlVCQH02, **token_type**=Bearer, **expires_in**=3600, **refresh_token**=1/BQB10JcUU440TMJAUk4L8rTt07V-RQdfRCVXKtkEi1Y, scope=null, open_id_token=eyJhbGciOiJSUzI1NiIsImtpZCI6ImM3OTc2ZTVmYTk0MjM5ZmFlOWI4NjY3MTgxMDIxMWNlZWQ0NjhkMTYifQ.eyJhenAiOiI5OsfDQxNjk4NTU1MzUtcm1zZmJ2MTFpa2luYTNocmIzbjg1aDBudDkzaGRiMnQuYXBwcy5nb29nbGV1c2VyY29udGVudC5jb20iLCJhdWQiOiI5ODQxNjk4NTU1MzUtcm1zZmJ2MTFpa2luYTNocmIzbjg1aDBudDkzaGRiMnQuYXBwcy5nb29nbGV1c2VyY29udGVudC5jb20iLCJzdWIiOiIxMDk}

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.

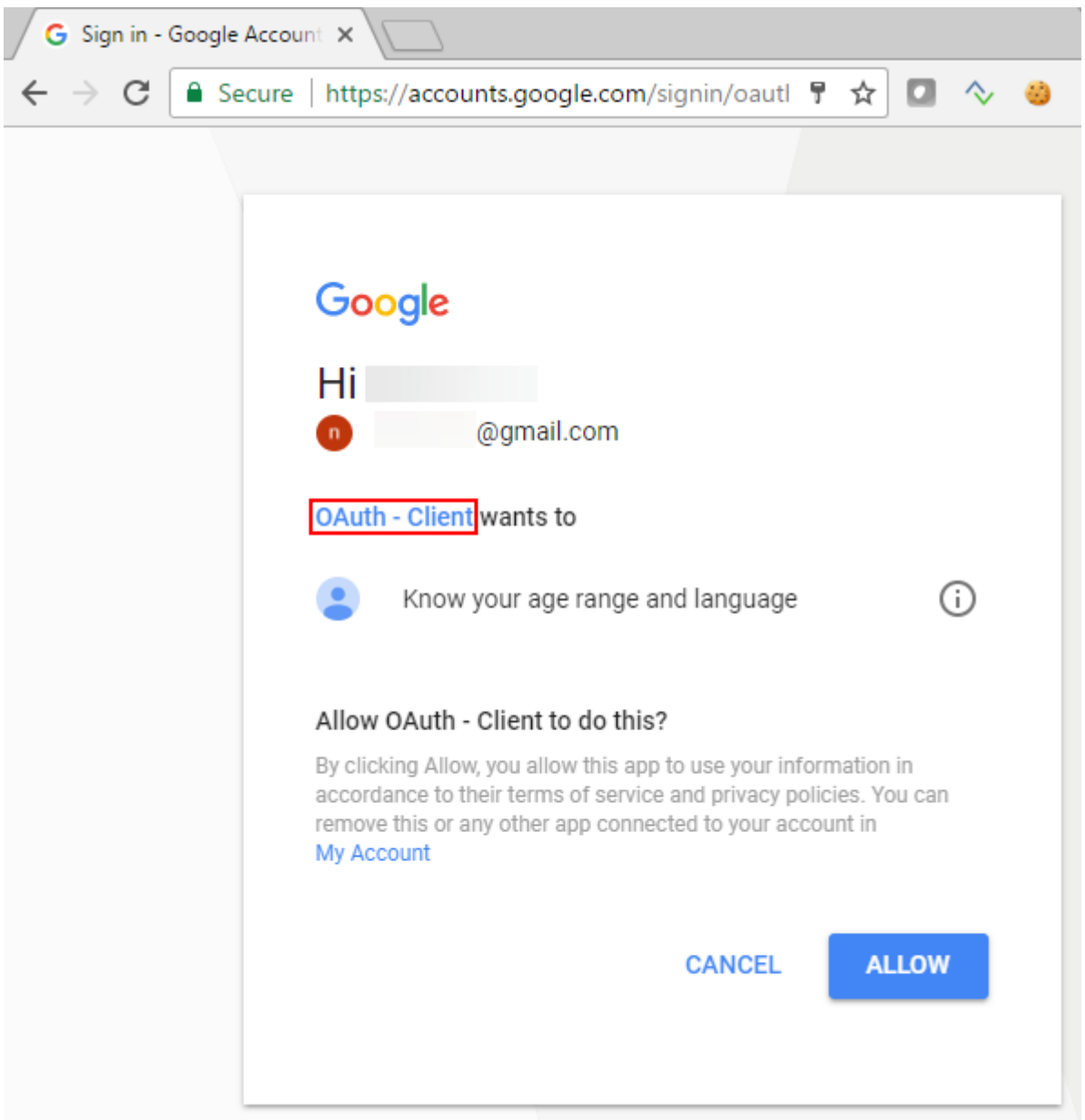
All the hard work done, it's time to access user's information

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.



Login with Gmail credentials and provide/deny authorization below.



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