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 his account information. Eg. Google, Facebook, Github etc.
2. Third party application (Client application) – application which wants to access a part of users account information
3. User - who authorizes a third-party application to access a part of his account information

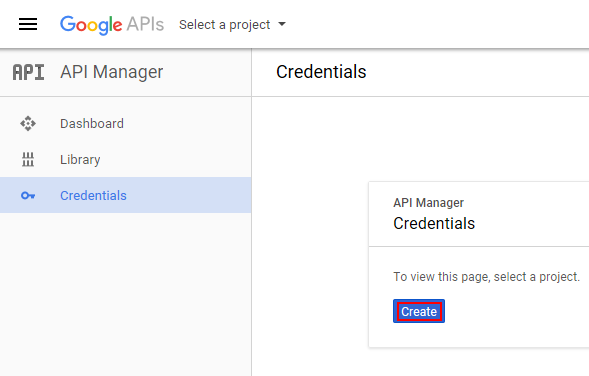
Before a third-party application can access user’s data it must be registered with the service provider.

Lets take an example to see the end-to-end process where a third party application (client application) - “OAuth client” wants to access basic users account information from google.

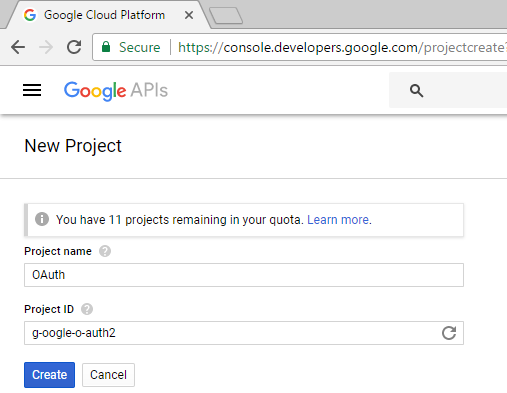
**Step 1** – Register third-party application “OAuth client” with google.

To register an application we need to create a project 1st in google developer console

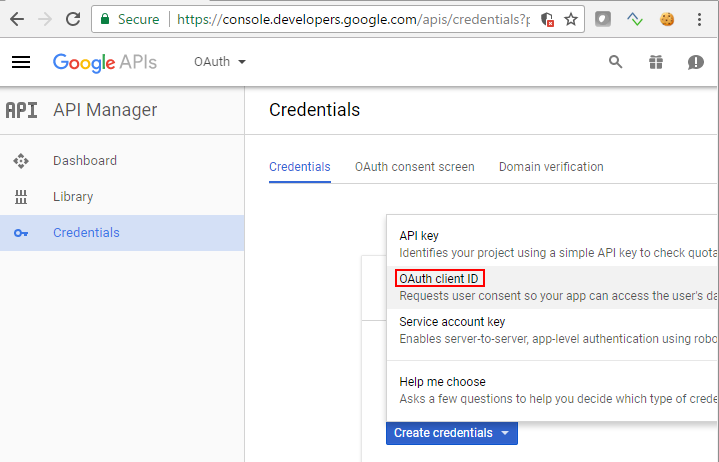
<https://console.developers.google.com/projectselector/apis/dashboard?authuser=1&organizationId=0>



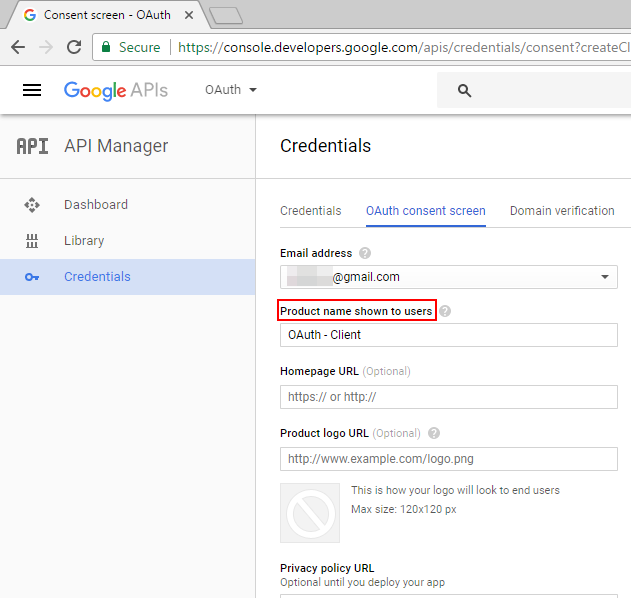
Project name can be anything but project id has to be unique

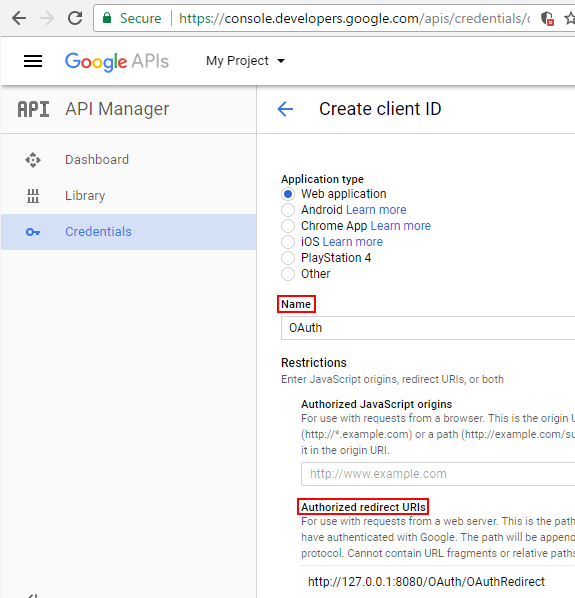


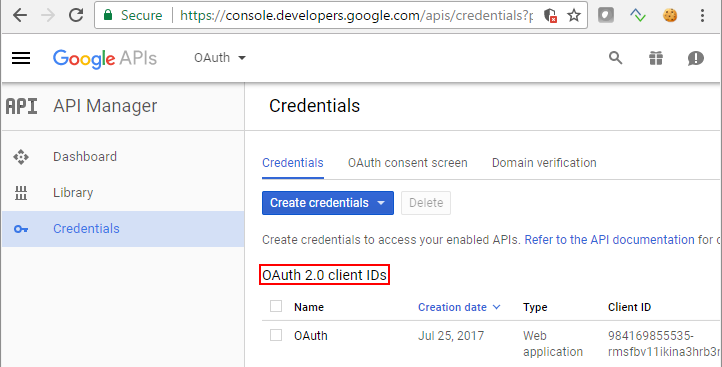
Configure the consent screen (screen that will be shown to the user asking for permission to access his account information), name of the web application and redirect uri (uri to which the user will be redirected after his authorization)



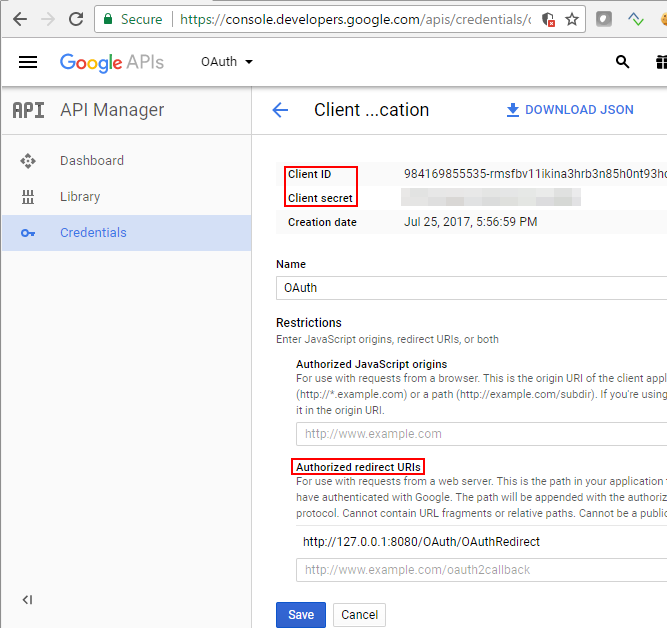
Configure the consent screen 1st







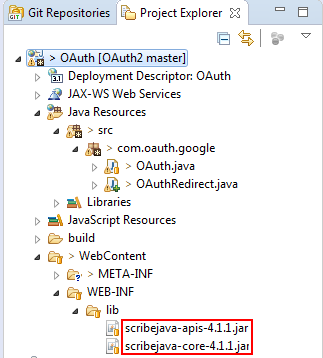
Once done with above steps you can see your client id, client secret and redirect uri. Make sure to keep your client secret confidential else someone can impersonate your application.



Details everything here

**Step 2** – 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 to implement oauth.

Project structure



**SS of home.java**

Explanation

clientId – id that we got at the time of registering our application with the service provider (Provide link to SS)

clientSecret – secret that we got at the time of registering our application with the service provider (Provide link to SS)

redirectUri – redirect uri (Provide link to SS) that we have given at the time of registering our application

scope – scope defines which part of the account information we want to access

secretState – a random string to prevent csrf attacks, it will be verified later

authUrl – Prepare the google authorization url to redirect when the user loads the home page of the application

authUrl session varaibale

Here is what an authorization URL looks like

<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%2FOAuthRedirect&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fplus.login&state=secret898652> (URL encoded)

Put SS from browser dev tool for showing redirection

Consent SS

**Explain redirect.java**

**SS here**

Code – a unique code that Google has sent, take the code from the request parameter

Paste code value

authUrl session varaibale

Till this point we haven’t accessed the users information yet we have just got his consent to access the information. ☺

accessToken -pass the code to get the access token which will be used to get

authRequest - create openauth request to access users information and specify the protected resource url and the method. Pass the accesstoken to get the users information.

Try to access username and email address and display the same back to the user.