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.

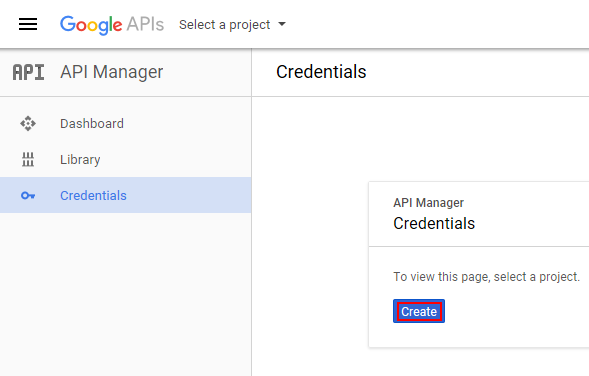
Before a third-party application can access user’s data, it must be registered with the service provider.

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 from Google account.

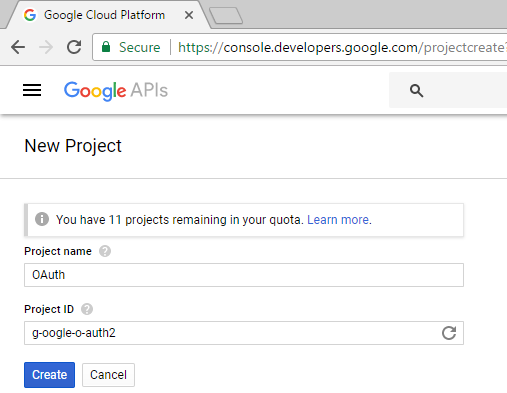
**Step 1** – Register third-party application “OAuth client” with google.

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

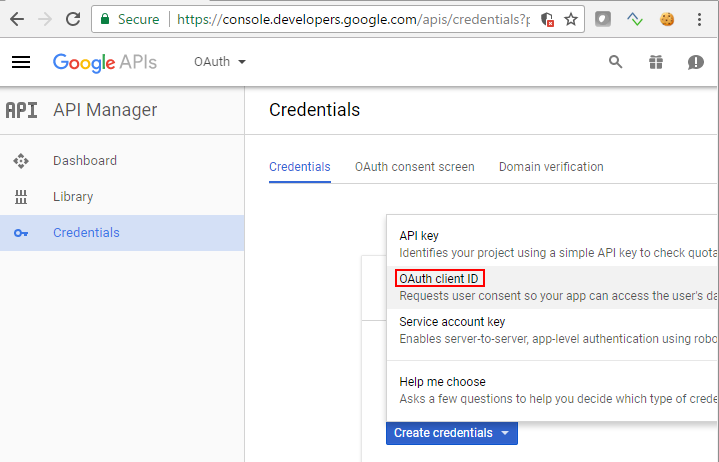
<https://console.developers.google.com/projectselector/apis/dashboard?authuser=1&organizationId=0>



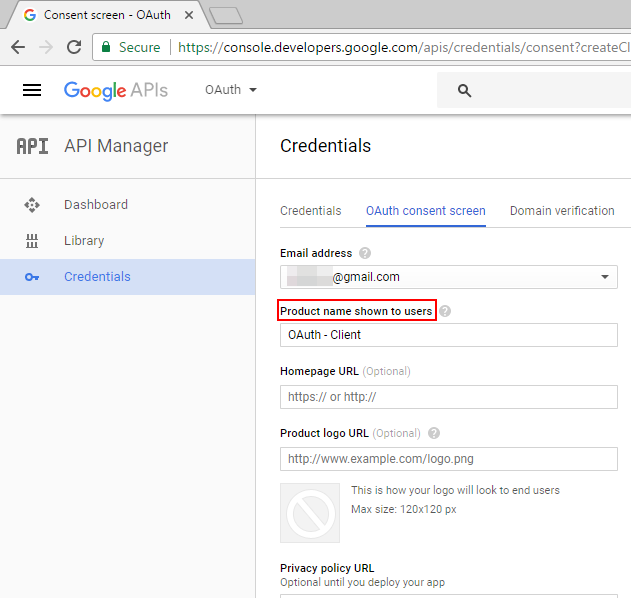
Project name can be anything but project id has to be unique

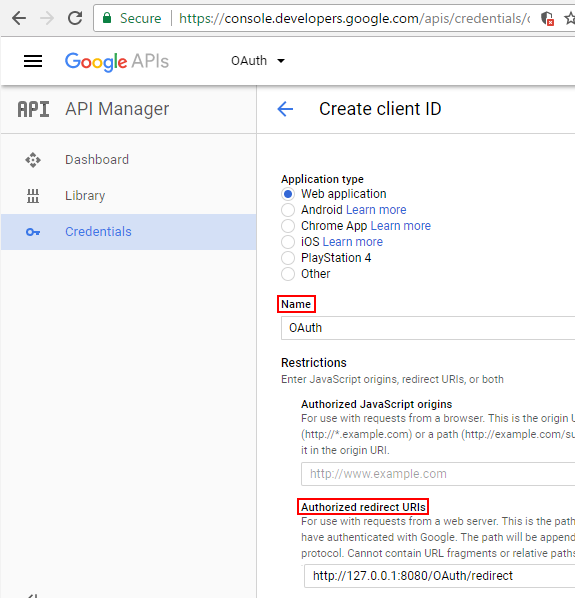


Configure the consent screen (screen shown to the user asking for permission to access his account information), name of web application and redirect uri (uri to which user will be redirected to after her/his authorization)

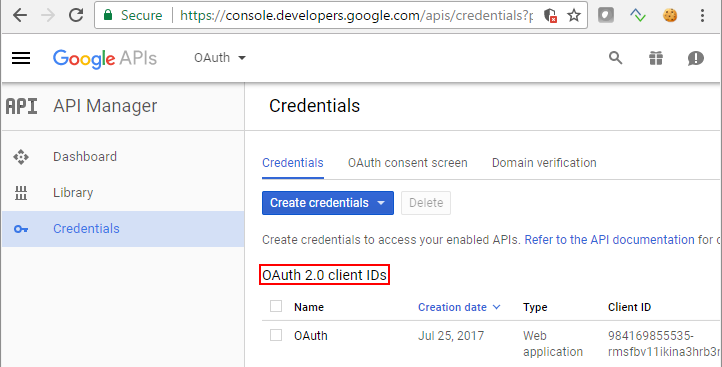


Configure the consent screen

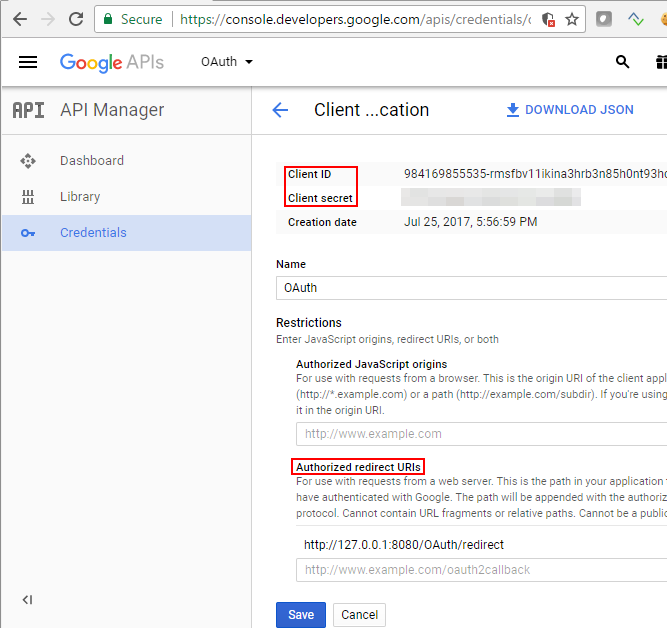




Credentials to access user information

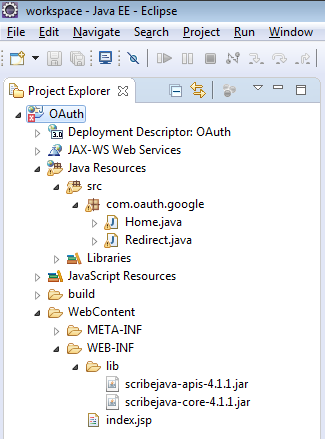


Once done with above steps you can see your client id, client secret and redirect uri. Make sure to keep your client secret confidential else someone can impersonate your application.

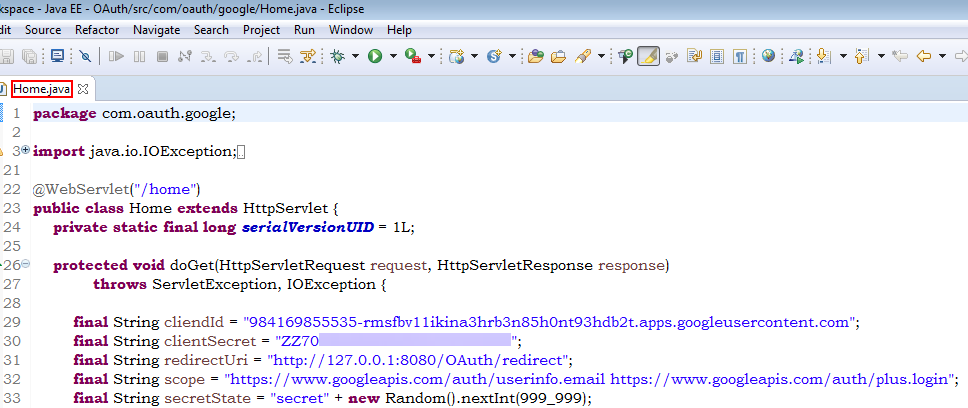


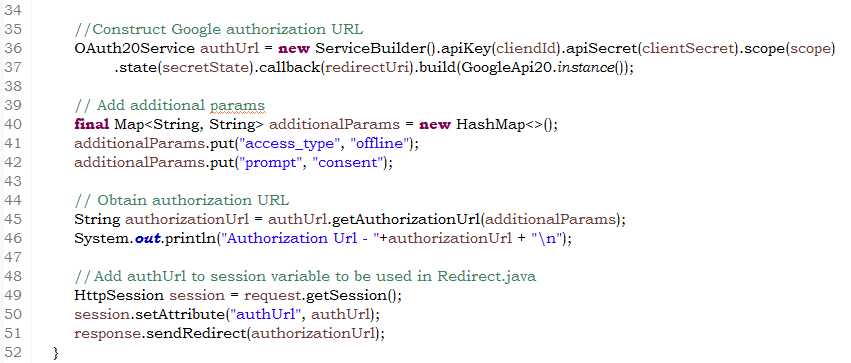
**Step 2** – Create a web application to access a user’s basic info (name and email address) and display it back to the end user. I am using scribe java library [scribe java library](https://github.com/scribejava/scribejava) to implement OAuth.

Project structure



Home.java

****

****

Explanation

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

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

[redirectUri](#clinetId) – 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](https://developers.google.com/oauthplayground/) to learn more about scopes

secretState – a random string to prevent [CSRF attack](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)), it will be verified later

authUrl – construct the Google authorization URL to redirect to when the user loads 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%2FOAuthRedirect&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fplus.login&state=secret898652> (URL encoded)

authUrl session variable – create session variable to be accessed in the redirect page.

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

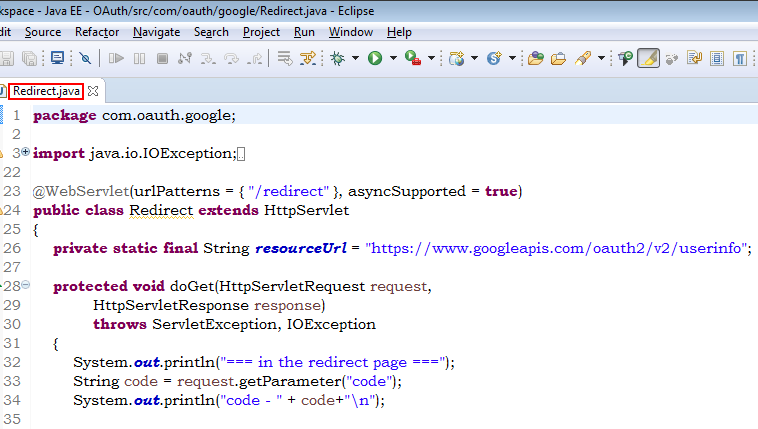
String authorizationUrl = “<https://accounts.google.com/o/oauth2/auth?access_type=offline&prompt=consent&response_type=code&client_id=984169855535-rmsfbv11ikina3hrb3n85h0nt93hdb2t.apps.googleusercontent.com&redirect_uri=http%3A%2F%2F127.0.0.1%3A8080%2FOAuth%2FOAuthRedirect&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fplus.login&state=secret898652>”;

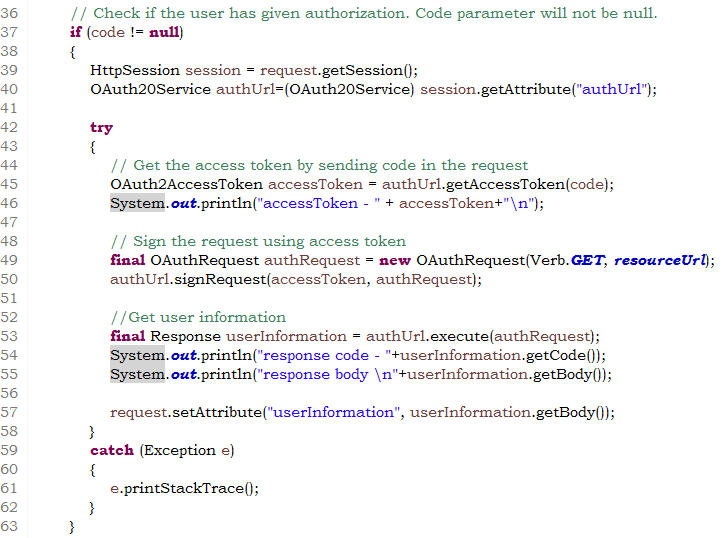
response.sendRedirect(authorizationUrl);

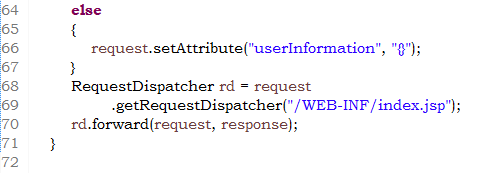
Put SS from browser dev tool for showing redirection

Consent SS

Redirct.java







Explanation

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

e.g. - Paste code value

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. - Paste code value

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

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