http://www.22steps.io

Enabling Gmail API by Google API authentication with Oauth 2

This article is an attempt to systematically understand the prerequisites required to be carried out before we start implementing our app to send emails using Gmail API.

Contents

Objective :

Able to understand code, implement and test sending emails from java application using Gmail API.

Points to know :

- If Gmail is used as SMTP provider: When we use Gmail as SMTP provider, it will work only in case if you use an app in the same location where you sign in to the gmail account.
- Gmail API : Even with scenarios like unblocking the device under Google Account Settings, it works only for short period of time. After some time problem re-iterates and is not a fruitful act. In that case you should use Gmail API.

Gmail API :

Sending emails through Gmail API is better option and safer than using any other method because it solves above mentioned points and also it is less likely that emails will be treated as SPAM and Google won't block the application.

Now let's look into how to implement sending emails using apps. Google allows accessing its applications with Google API which opens the doors to access its bundle of services from our own apps. Few to mention are Google Drive, Gmail, Calendar, Blogger, Cloud Vision, Analytics and many more. Here are two sections which we need to follow. First is to setup Google project and create credentials to access APIs . Second to create a java App to use the API.
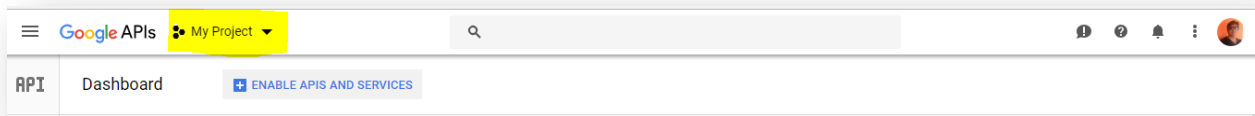
Prerequisites :

Create project

First you need to create a project in Google Developers Console.

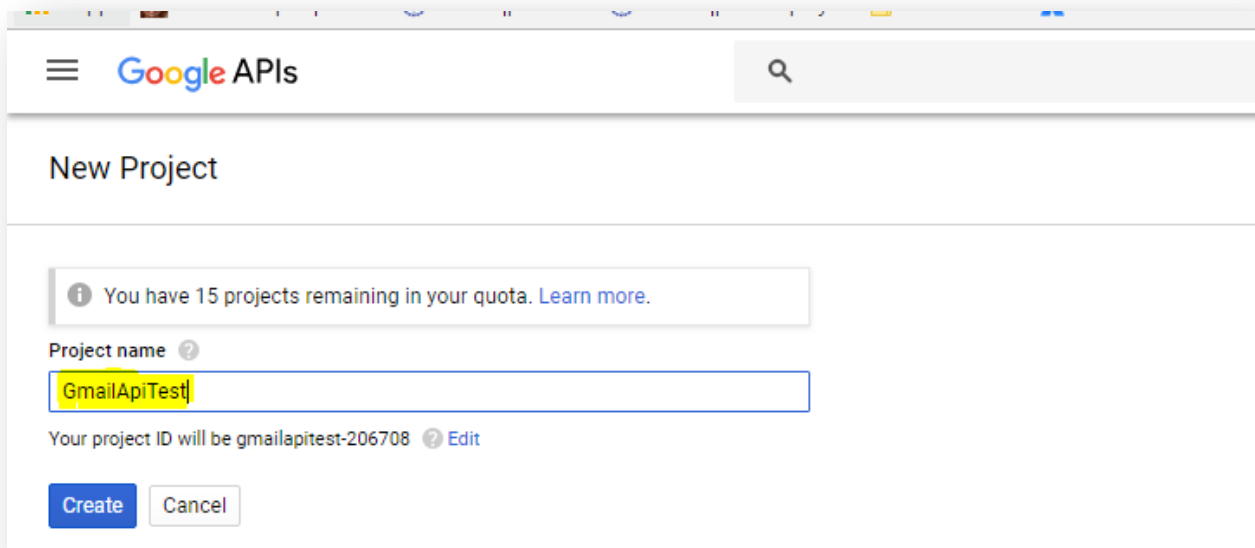https://console.developers.google.com/

Then you see the action bar on the top of the page:

Click on the list of projects:

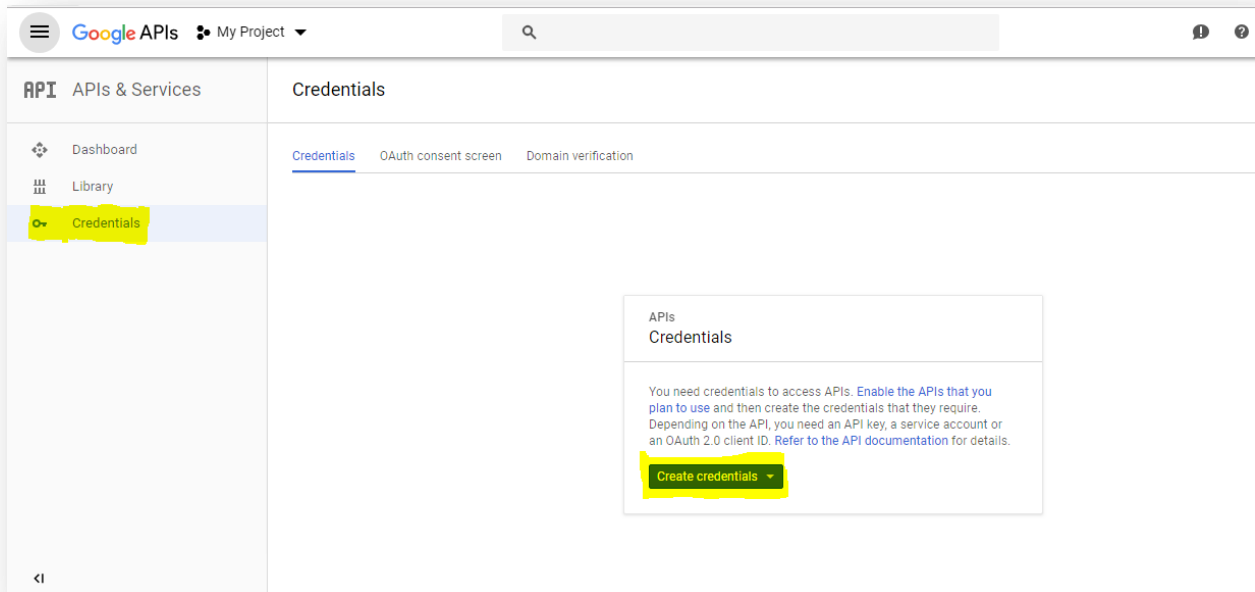and select "**New Project**" to add a new one.

Give your project a name. Let's name it "**GmailApiTest**".

Create project and go to **Credentials > OAuth consent screen**. Put the name of the product in the field "**Product name shown to users**". It is possible to use the same
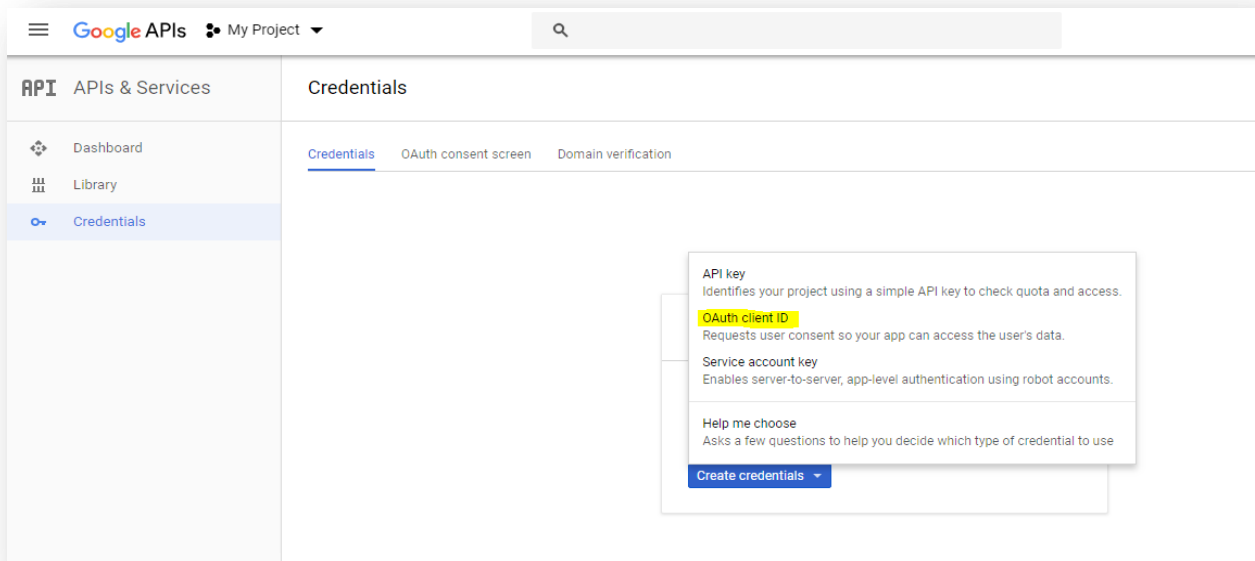
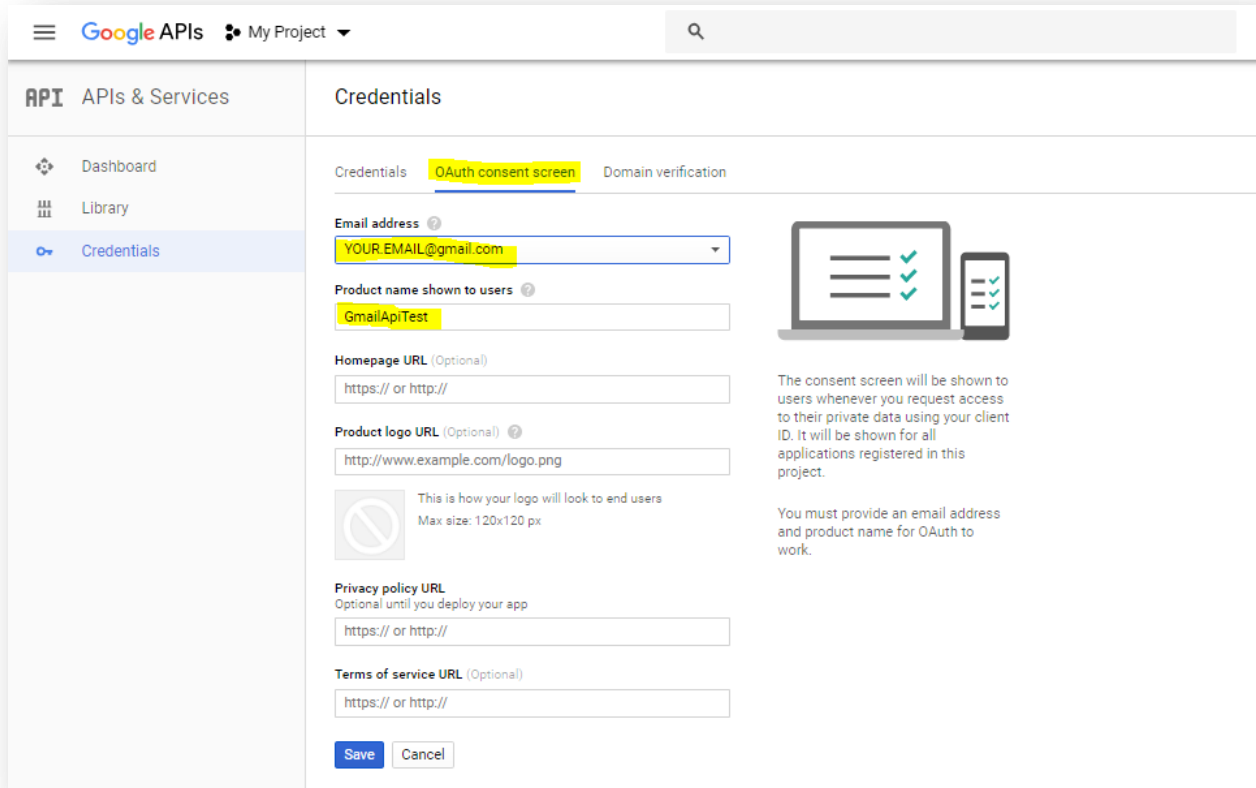name—"GmailApiTest". It will be displayed on the list of apps that have granted permissions to use google account.

Generate credentials

Credentials > credentials page allows to generate a key. Choose **OAuth client ID**

Create OAuth client ID

To create an OAuth client ID, you must first set a product name on the consent screen.
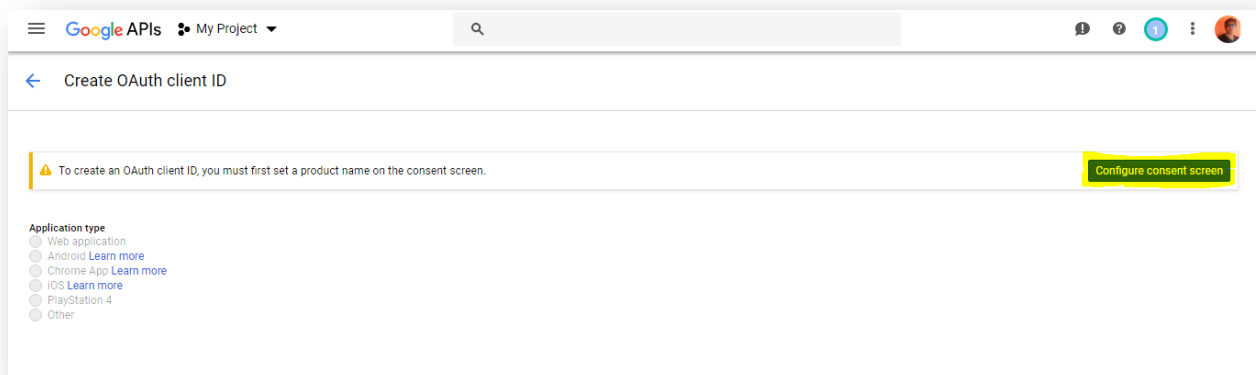

Enter details and click save. Optional fields can be skipped.

Choose application type. I prefer to select "**Other**". Put the name for the client.

If you choose different option, for example, *Web application* you have to insert one of the *Authorized redirect URIs*. For the purpose of this tutorial I suggest to put there

http://localhost

Save your OAuth client. Thanks to that you receive **clientID** and **clientSecret**.
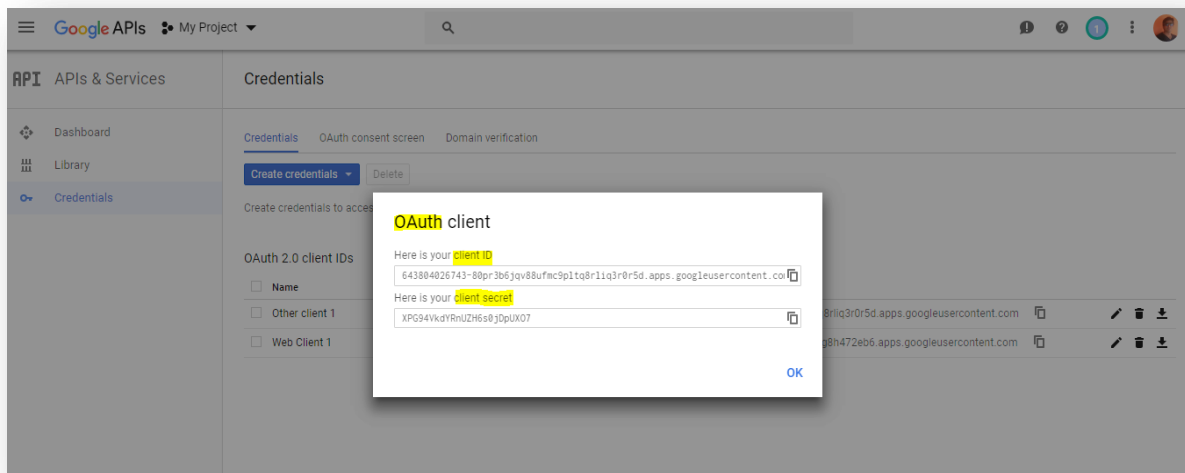
Get authorization tokens

There are 2 steps to get authorization tokens for your project. First is a request for an authorization **code** and then there is the code confirmation. After the process you will have an **access_token** and **refresh_token**. You need both of them if your app is offline app, in other case access_token is enough. Refresh_token is useful to renew access_token after it expires.

Full documentation of this process is available in the google developers website but you fortunately don't need it with my tutorial.

Get code for permissions request
Use your browser to request permissions for your app. You need to call

https://accounts.google.com/o/oauth2/v2/auth

with GET parameters.

- **client_id**

- **response_type**, put here **code**, it is the simplest way to authenticate by requesting the **code**

- **scope**—each API has its own scope. E.g. Gmail API has scope to send messages, compose messages etc. Auth scopes express the permissions you request users to authorize for your app. Choose [one of the scopes you need for Gmail](#). We'll choose

https://www.googleapis.com/auth/gmail.send

- **redirect_uri**—put here:

http://localhost

- **access_type**—use it with value *offline*, only if you need **refresh_token.**

Our request URI looks like that:

```
https://accounts.google.com/o/oauth2/v2/auth?client_id=643804026743-
80pr3b6jqv88ufmc9pltq8rliq3r0r5d.apps.googleusercontent.com&response_type=c
ode&scope=https://www.googleapis.com/auth/gmail.send&redirect_uri=http://localh
ost&access_type=offline
```

Grant Permission for App via Google Account

After the call, browser shows question about granting permissions to the app :
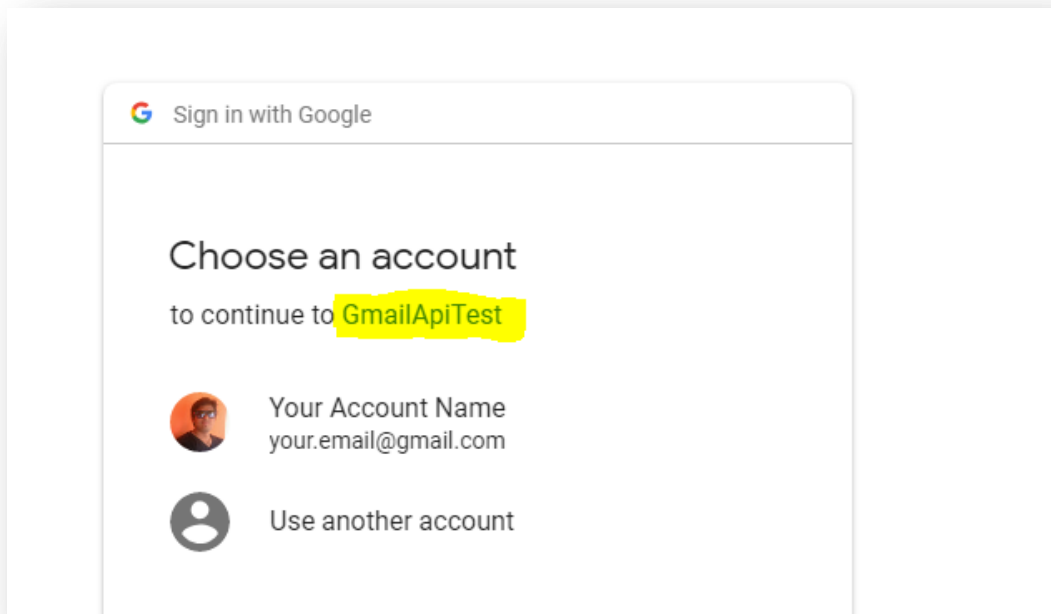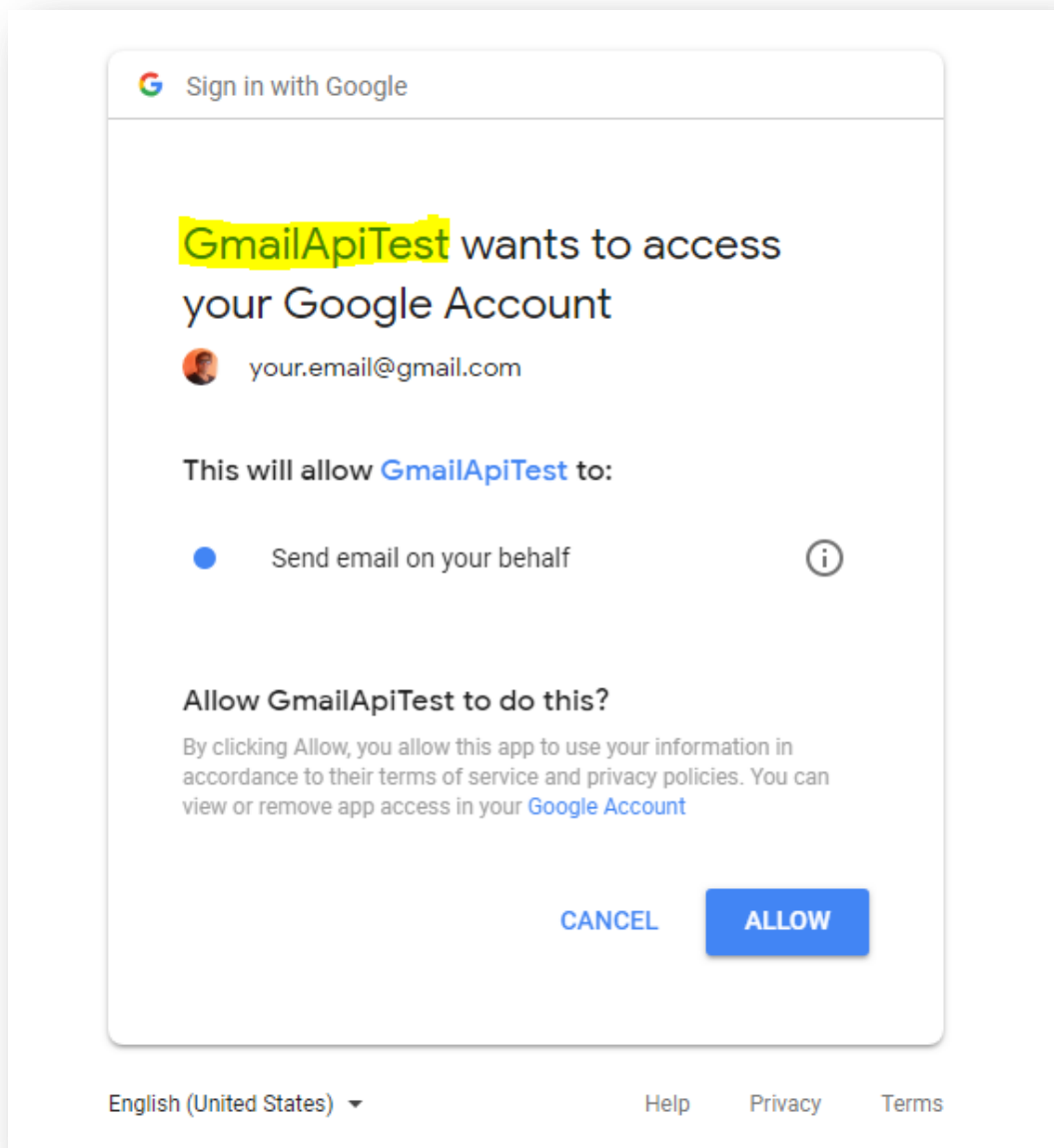
Turn off any service that is running on your computer that is available on port 80 with URL *http://localhost.* Click **"Allow"** and then we will be redirected into *localhost*.

Redirected URI looks as follows:

Confirm permissions request

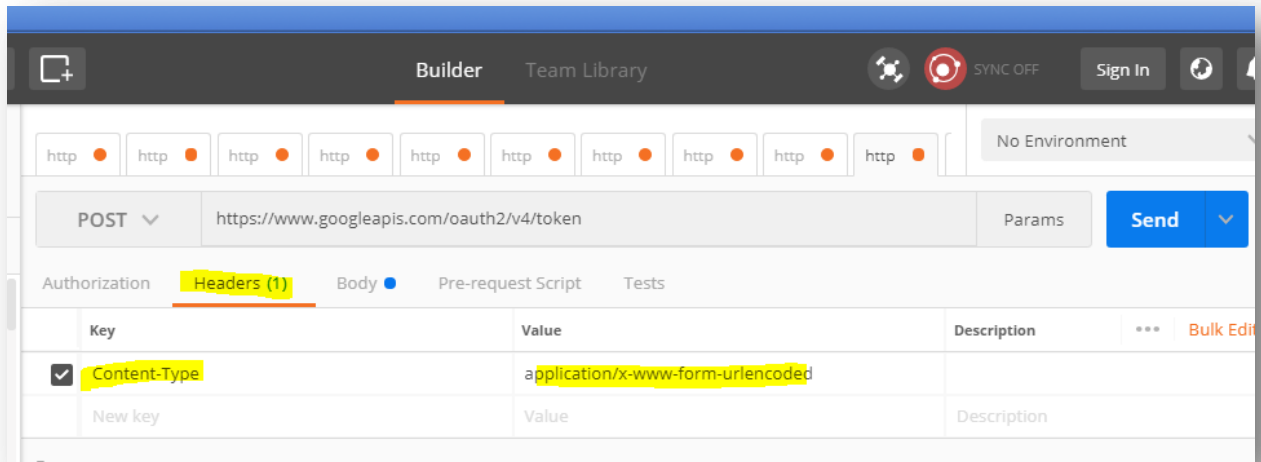We need to POST **code** with some other parameters on:

https://www.googleapis.com/oauth2/v4/token

http://www.22steps.io

Let's use rest client of your choice: I have chosen **Postman** here

Headers:

Content-Type: application/x-www-form-urlencoded

Body:

code=4/AACZoszhUIIh0Gqrv0g51U09mfuCucGd4zGZW1Qw2i-
LsORKKg99GJbZ5oZTa73_0XAWDqAKqzklH0QMWx-PrYA#&client_id=643804026743-
80pr3b6jqv88ufmc9pltq8rliq3r0r5d.apps.googleusercontent.com&client_secret=XPG94Vk
dYRnUZH6s0jDpUXO7&grant_type=authorization_code&redirect_uri=http://localhost

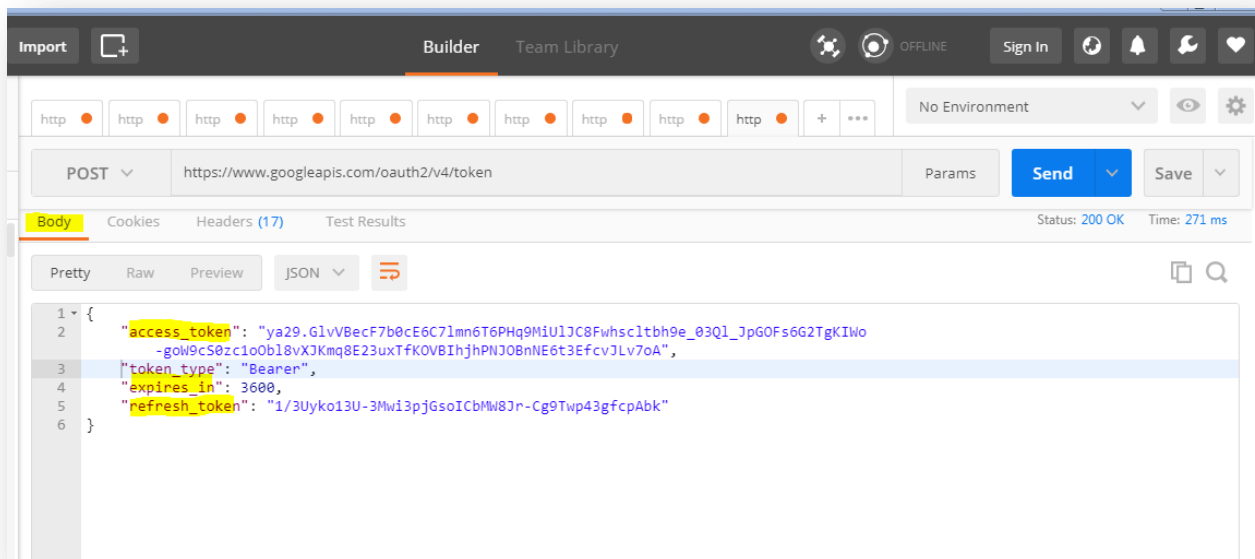The answer gives us **access_token** and **refresh_token**:

Parameters that we need:

- **code**
- **client_id**
- **client_secret**
- **grant_type**—use *authorization_code*
- **redirect_uri**—use *http://localhost*

http://www.22steps.io
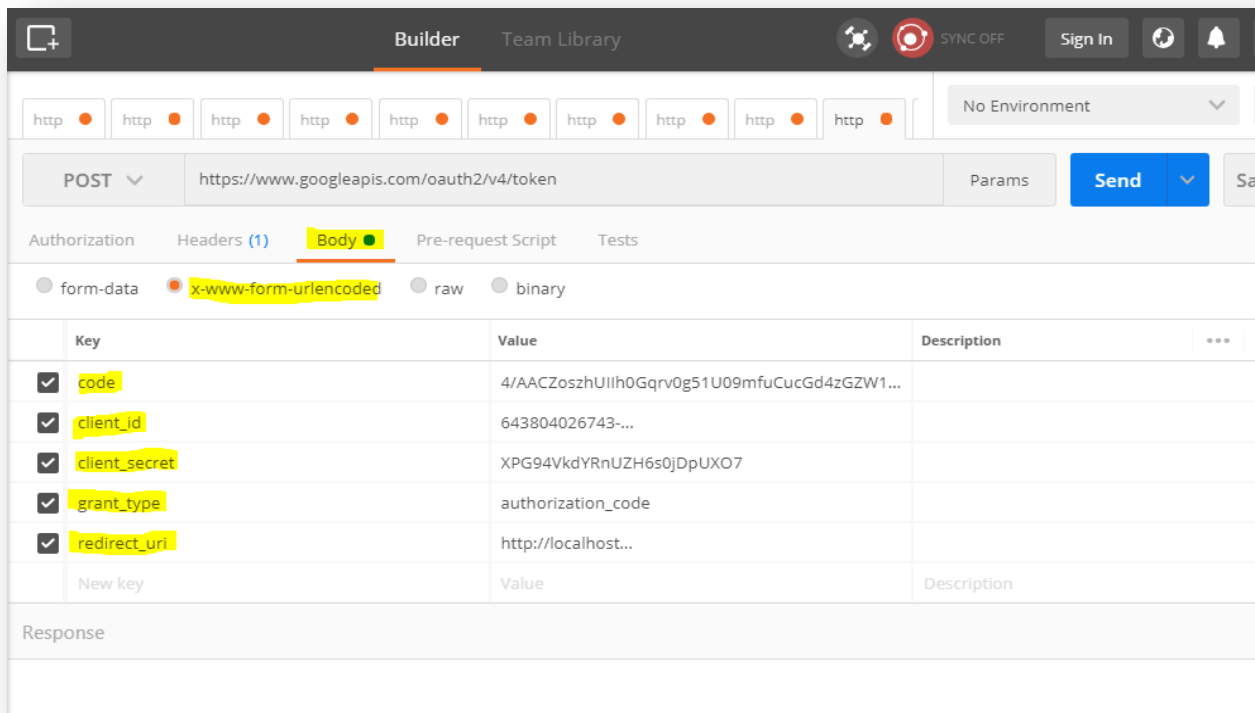
Generate Access and Refresh Tokens

```
{

    "access_token":
"ya29.GlvVBecF7b0cE6C7lmn6T6PHq9MiUlJC8Fwhscltbh9e_03QI_JpGOFs6G2T
gKIWo-goW9cS0zc1oObl8vXJKmq8E23uxTfKOVBlhjhPNJOBnNE6t3EfcvJLv7oA",

    "token_type": "Bearer",

    "expires_in": 3600,

    "refresh_token": "1/3Uyko13U-3Mwi3pjGsoICbMW8Jr-Cg9Twp43gfcpAbk"

}
```

As you can see in the picture, application "GmailApiTest" has access to sending emails through Gmail.
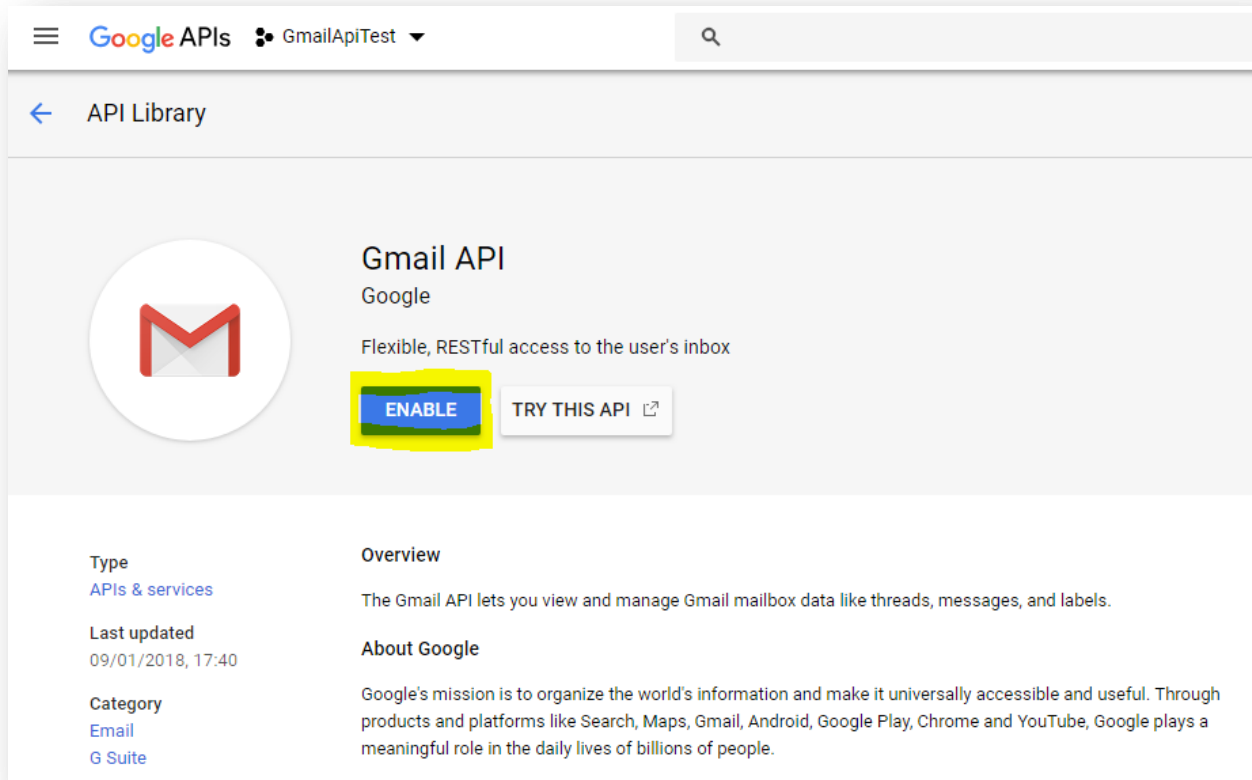
http://www.22steps.io

Enable API

In the end enable API you need. It is not enough to have a permission to some service. API need to be enabled.

Do it by clicking "**Enable**" for specific API. In case of Gmail, go to https://console.developers.google.com/apis/library/gmail.googleapis.com/?project=gmailapitest-206708
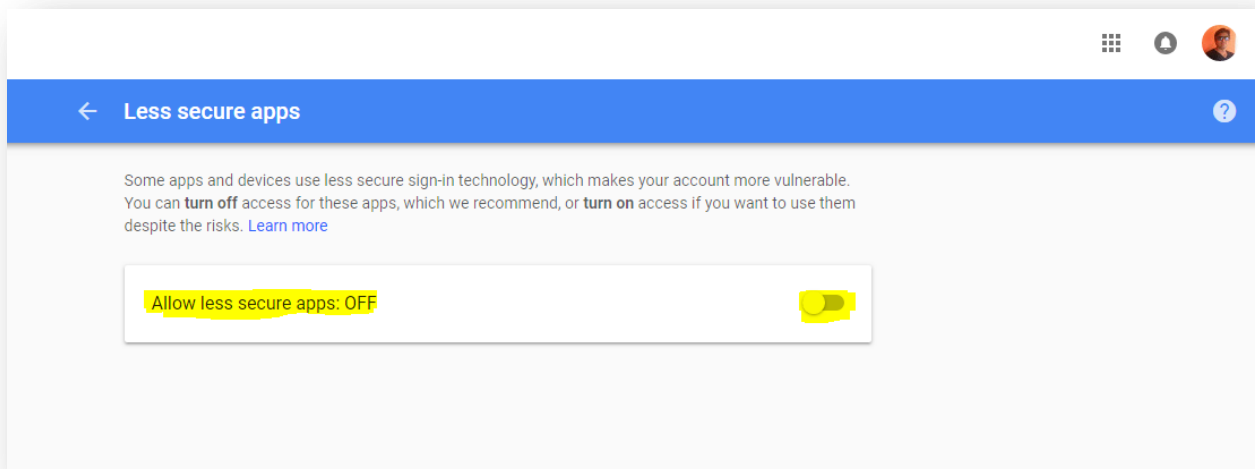

Allow less Secure Apps

In some case, there are issues with authentication. Acoording to Security and privacy. Google may block sign-in attempts from some apps that do not use modern security standards, such as OAuth 2.0.

For javamail, the solution prevent this error is : Go to the "Less secure apps" section in Google Account. https://myaccount.google.com/lesssecureapps

Next to "Access for less secure apps," select Turn on.
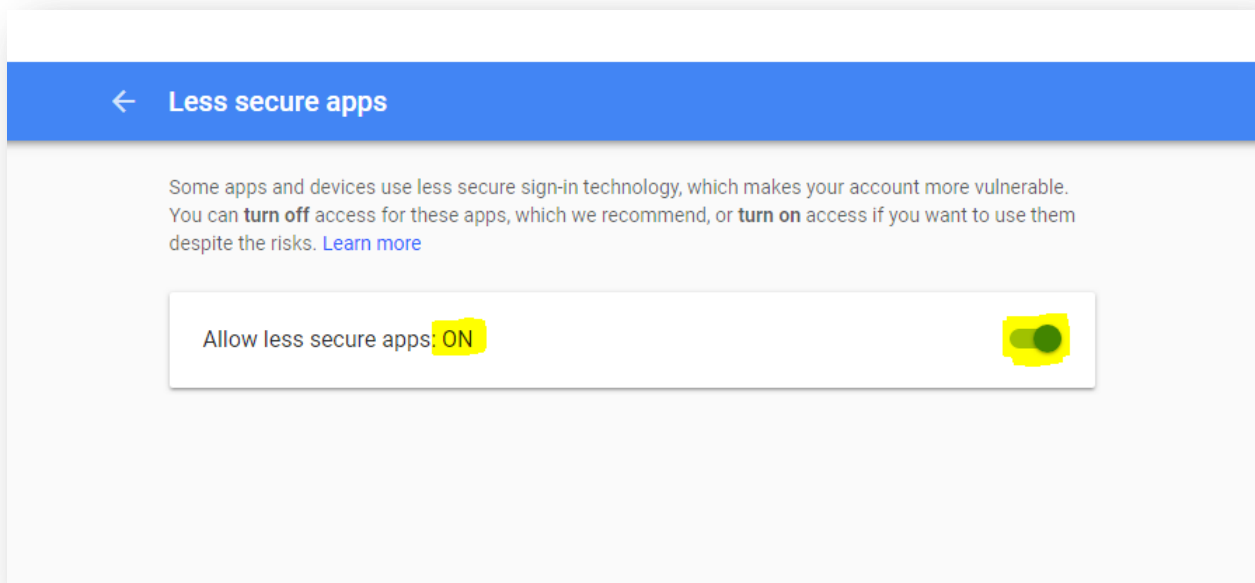
Summary

I hope you've learned how to authenticate to Google APIs and you will be able to use the power of Google Services.

After this tutorial you can sending emails by Gmail API from your Java app.

If you have any questions or comments. Please let me know. Constructive feedback is appreciated.