Fitbit Dev Guide

Overview

There are several steps for using Fitbit Oauth 2.0 to authenticate and fetch user information. Here I did some experiments using python, to fully understand how it works.

1) Create an application in the cloud, which has an **App ID and App secrete**, these two have uniquely define the application we used for fetching user's information.
2) Construct an url, that contains the above App ID information, so that we can login. In this example, we set client_id which is the AppID. Other useful information is the type of authentication(cloud-based, or application/client-based), what kind of information we want to fetch,the information is important because if mismatch there are several errors, the url is: https://www.fitbit.com/oauth2/authorize?response_type=code&client_id=227NCS&redirect_uri=http%3A%2F%2Fwww.cs.uml.edu&scope=activity%20nutrition%20heartrate%20location%20nutrition%20profile%20settings%20sleep%20social%20weight. After visiting the above url in browser, it will display a **login webpage**, in that page, enter the username and password, press login. After it verifies your account information, we will be redirected to the **redirect_url** that contained the necessary **authorization_code**, manually mark it down, we will use it later
3) In this step, we will use App ID, App secrete, authorization_code, to construct a HTTP request, with necessary header files, this will request to get the response which will contain the **access_token** and **refresh_token**(will be invalid if the time expires)
4) Parse the response and use this token, as well as the clientID, as well as the RESTful API request, and necessary header, to send request to server to fetch information.

The above description is what I have learned by the experiments with the python code, for implementation details, there is some code that stores the python code: https://github.com/csuml/research_by_chang_liu/tree/master/2016_06_FitbitApp_YYGA/prototype/Oauth2

NOTE: some of the steps like visiting the url is not done by python code(which has done manually), there are only two implementation which will send the request using python libs, and the parse of response should also be included in the code(now I manually cut&paste the authorization_code/token)

Progress

* review of easysocial package, get familiar with code (May 28th ~ May 30th)
* go through tutorial, write python code for fetching information (May 31th ~ June 1st)
* fix the issue of `scope`, when logging in we should use the “+” to concatenate multiple permission, instead of just using “,” (June 2nd)
* fix the log in issue, now we use **Implicit Grant Flow** method, which is 3-step, instead of our python code’s 4-step(**Authorization Code Flow** method) (June 3th)
* fix the issue of saving wrong access_token to the shared_preference (June 3th)
* fix JSONObject null exception that fail to set the display content. (June 6th)
* add all the health information button and listener, callback for HTTP GET response, fix the bugs for different responses/JSON parser, now we can fetch the information. (June 7th)

BugList

* NullPointerException when requesting the user information [fixed]
* should store the authorization code, now it seems that there is no such function. [Fixed, we don’t use authorization code method, so it’s not necessary, see the New Finding Section)
* one strange issue(when visiting login url, the PC browser gives back an **authorization_code**, but in my android code it directly gives back the **access_token**, which seems to omit one step, that will make me cannot get the authorization code, and my request json cannot be completed due to this issue) [urgent!!!] (Fixed, see Authorization Flow Difference (**))
* Json object is null(when parse response of fetch information, we can see the string, but after passing it to the onGetUserInfoFunction(), the passed json object is null(I have made sure that the constructed json object is not null before passing the parameter) [Fixed]

Remaining Problem

* ~~Get user information, still has null pointer exception, because there is some url left for requesting information. See step4.py as an example, but here we got a bit different implementation with using it directly. (June 3th)~~ [Fixed, see JSONException, June 6th]

Authorization Flow Difference

(**)What’s the difference of **Authorization Code Flow** and **Implicit Grant Flow**?

From the Fitbit Tutorial, we can see as the below shows:

Authorization Code Flow:

```
https://www.fitbit.com/oauth2/authorize?response_type=code&client_id=22942C&redirect_uri=http%3A%2F%2Fexample.com%2Ffitbit_auth&scope=activity%20nutrition%20heartrate%20location%20nutrition%20profile%20settings%20sleep%20social%20weight
```

Implicit Grant Flow:

```
https://www.fitbit.com/oauth2/authorize?response_type=token&client_id=22942C&redirect_uri=http%3A%2F%2Fexample.com%2Ffitbit_auth&scope=activity%20nutrition%20heartrate%20location%20nutrition%20profile%20settings%20sleep%20social%20weight&expires_in=604800
```

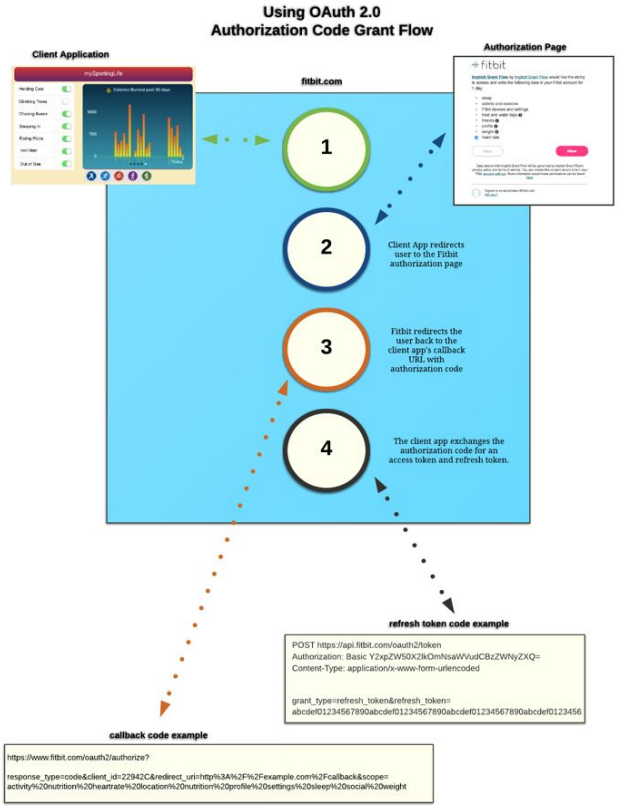
In my Python code, I used this url to login:

https://www.fitbit.com/oauth2/authorize?**response_type=code**&client_id=227NCS&redirect_uri=http%3A%2F%2Fwww.cs.uml.edu&scope=activity%20nutrition%20heartrate%20location%20nutrition%20profile%20settings%20sleep%20social%20weight

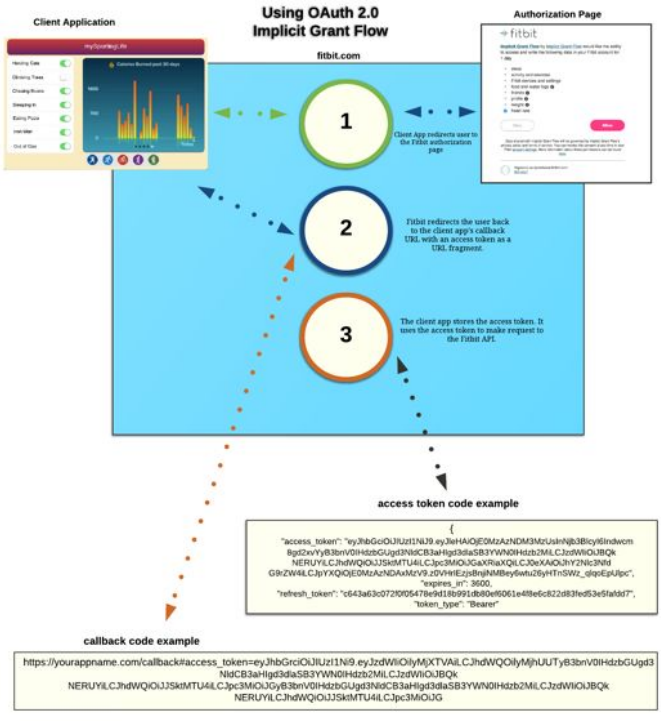
In my android code, i used this url to login: 06-03 16:02:16.751 https://www.fitbit.com/oauth2/authorize?**response_type=token**&client_id=227NCS&redirect_uri=http://www.cs.uml.edu&scope=activity+nutrition+heartrate+location+sleep+profile+settings+social+weight

See the difference? We used two different ways of authentication!

Let's see the difference of two methods:



And the **Implicit Grant Flow**:



There is less steps for using Implicit Grant Flow(3 steps instead of 4 steps), we don't need the authorization code, that's why we get another url instead of the python-code's 4-step code!

So, which one we should use for Android Application?

According to the tutorial:
"The Authorization Code Grant flow is recommended for applications that have a web service. This flow requires **server-to-server** communication using an application's client secret.
Note: Never put your client secret in distributed code, such as apps downloaded through an app store or client-side JavaScript.

Applications that do not have a web service should use the Implicit Grant flow."

"Unlike the Authorization Code Grant Flow, the refresh tokens are not issued with the Implicit Grant flow. Refreshing a token requires use of the client secret, which cannot safely be stored in distributed application code. When the access token expires, users will need to re-authorize your app.
Access tokens from the Implicit Grant Flow are longer lived than tokens from the Authorization Code Grant flow. Users may specify the lifetime of the access token from the authorization page when an application uses the Implicit Grant flow. The access token lifetime options are 1 day, 1 week, and 30 days. Applications can pre-select a token lifetime option, but the user ultimately decides.

If an application using the Implicit Grant Flow sends a user to the authorization page before the previously issued access token has expired, the user will not be prompted unless the scope has increased. The user will be redirected immediately to the application with an access token.

Access tokens obtained via the Implicit Grant Flow should only be stored on the device used to obtain the authorization. If your application has a web server component, your application should use the Authorization Code Grant Flow."

HTTPURLConnection

Error information:

06-05 22:21:13.066 15694-15694/com.uwanttolearn.easysocialfacebooktesting E/EasySocialFitbit: callback line null
06-05 22:21:13.074 15694-15694/com.uwanttolearn.easysocialfacebooktesting E/AndroidRuntime: FATAL EXCEPTION: main
Process: com.uwanttolearn.easysocialfacebooktesting, PID: 15694
java.lang.NullPointerException: Attempt to invoke virtual method 'int java.lang.String.length()' on a null object reference
at org.json.JSONTokener.nextCleanInternal(JSONTokener.java:116)
at org.json.JSONTokener.nextValue(JSONTokener.java:94)
at org.json.JSONObject.<init>(JSONObject.java:156)
at org.json.JSONObject.<init>(JSONObject.java:173)
at com.uml.deercoderi.fitbitmodule.EasySocialFitbit$1.requestComplete(EasySocialFitbit.java:116)
at com.uwanttolearn.easysocial.webrequests.GetWebRequest$LocalAsyncTask.onPostExecute(GetWebRequest.java:83)
at com.uwanttolearn.easysocial.webrequests.GetWebRequest$LocalAsyncTask.onPostExecute(GetWebRequest.java:48)
at android.os.AsyncTask.finish(AsyncTask.java:651)
at android.os.AsyncTask.access$500(AsyncTask.java:180)
at android.os.AsyncTask$InternalHandler.handleMessage(AsyncTask.java:668)
at android.os.Handler.dispatchMessage(Handler.java:102)
at android.os.Looper.loop(Looper.java:148)
at android.app.ActivityThread.main(ActivityThread.java:5438)
at java.lang.reflect.Method.invoke(Native Method)
at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:739)
at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:629)

Solution:

You need to set it to true if you want to send (*output*) a request body, for example with POST or PUT requests. **With GET, you do not usually send a body, so you do not need it.**

Sending the request body itself is done via the connection's output stream:

```
conn.getOutputStream().write(someBytes);
```

JSON Exception

When I finished the parsing of HTTP GET request, I find that the response String is correct, but after I construct a JSON Object in my Fitbit Module, then call the MainFragment's function to set the content of user information.

```
GetWebRequest getWebRequest = new GetWebRequest(new WebRequest.Callback() {
   @Override
   public void requestComplete(String line) {
       try {
           // Chang, to be fixed here, it's null!!
           Log.e("EasySocialFitbit", "callback line " + line);
           JSONObject jsonObject = new JSONObject(line); // Chang, if I use "aaa" value, it
will pass, but wrong value
           if (jsonObject == null) {
               Log.e("EasySocialFitbit", "null json object");
           }
           userInfoCallback.onComplete(jsonObject);
       } catch (JSONException e) {
           e.printStackTrace();
           Log.e("EasySocialFitbit", "Exception when consturction JSON Object");
           userInfoCallback.onComplete(null);
       }
   }
});
```

In the above code snippets, the red code is called, as the **onComplete() function** is called, but here the `Log.e("EasySocialFitbit", "null json object");` log is not printed, which shows that the **json object is not null**.

In the next function, however, we are sure that **onComplete() is called** because of the log output, but it goes into the `if(jsonObject == null)` direction.

```
            mEasySocialFitbit.getUserInfo(getActivity(),
                    new EasySocialFitbit.UserInfoCallback() {
                        @Override
                        public void onComplete(JSONObject jsonObject) {
                            Log.e("MainFragment", "onGetUserInfoClick's onComplete is
clicked!");
                            if(jsonObject == null){
                                Log.e("MainFragment", "Here!!!!!");
                                mResponseTextView.setText("UserInfo null");
                            }else{
                                mResponseTextView.setText(jsonObject.toString());
                                mUserId = jsonObject.optString("id");
                                if(mUserId != null){
                                    enableButton(mGetFriendsButton);
                                    enableButton(mPostMessageButton);
                                }
                            }
                            progressDialog.dismiss();
                        }
                    });
```

Reasons:

It takes some time to locate this error. I haved tried to find if there is some errors in the AsynTask, or the Callback interface as it's not implemented but created a new interface, however all are in vain.

Finally I go back to the root call, and find there is an Exception handler, to my surprise, there is a call of null, but my first guess is still try to add the logs in the Exception to see if it really goes into it. And the log is as follows:

```
06-06 10:13:31.923 7457-7457/com.uwanttolearn.easysocialfacebooktesting
E/MainFragment: GetUserInfo is clicked!
06-06 10:13:31.955 7457-8654/com.uwanttolearn.easysocialfacebooktesting
E/GetWebRequest: GetWebRequest's doInBackground
https://api.fitbit.com/1/user/-/activities/frequent.json
06-06 10:13:32.896 7457-8654/com.uwanttolearn.easysocialfacebooktesting E/WebRequest:
WebRequest's inputStreamParse [{"activityId":90013,"calories":0,"description":"less than 2
mph, strolling very
slowly","distance":0,"duration":1024000,"name":"Walk"},{"activityId":15000,"calories":0,"descri
ption":"","distance":0,"duration":1229000,"name":"Sport"},{"activityId":3001,"calories":0,"descri
ption":"","distance":0,"duration":1282000,"name":"Aerobic
Workout"},{"activityId":1071,"calories":0,"description":"","distance":0,"duration":976000,"name"
:"Outdoor Bike"},{"activityId":90009,"calories":0,"description":"5 mph (12
```

min/mile)","distance":0,"duration":1382000,"name":"Run"},{"activityId":20047,"calories":0,"description":"","distance":0,"duration":974000,"name":"Elliptical"}]
06-06 10:13:32.905 7457-7457/com.uwanttolearn.easysocialfacebooktesting E/EasySocialFitbit: callback line [{"activityId":90013,"calories":0,"description":"less than 2 mph, strolling very slowly","distance":0,"duration":1024000,"name":"Walk"},{"activityId":15000,"calories":0,"description":"","distance":0,"duration":1229000,"name":"Sport"},{"activityId":3001,"calories":0,"description":"","distance":0,"duration":1282000,"name":"Aerobic Workout"},{"activityId":1071,"calories":0,"description":"","distance":0,"duration":976000,"name":"Outdoor Bike"},{"activityId":90009,"calories":0,"description":"5 mph (12 min/mile)","distance":0,"duration":1382000,"name":"Run"},{"activityId":20047,"calories":0,"description":"","distance":0,"duration":974000,"name":"Elliptical"}]
06-06 10:13:32.906 7457-7457/com.uwanttolearn.easysocialfacebooktesting E/EasySocialFitbit: Exception when consturction JSON Object
06-06 10:13:32.906 7457-7457/com.uwanttolearn.easysocialfacebooktesting E/MainFragment: onGetUserInfoClick's onComplete is clicked!
06-06 10:13:32.906 7457-7457/com.uwanttolearn.easysocialfacebooktesting E/MainFragment: Here!!!!!

Which clearly shows that the response string is correct and correspond to our GET request, but the JSON object is not null as it encounter an exception, so it goes into the loop and call the null parameter method of onComplete().

Solution:

Find the reason why there is an JSONException, as the constructor supports using the string to construct. It's caused by the "[" and "]" of the parse string, just get the substring of the whole string.

Reference

1) Repo: https://github.com/csuml/research_by_chang_liu/tree/master/2016_06_FitbitApp_YYGA

Chang Liu

June 1st - Jun XX