Nov 2nd, 10:00 AM - 10:15 AM

An Approach That Integrates Android Application With Google Scripts, Solves Different Problems Of Emergency Character

Tamara Luarasi
European University of Tirana

An Approach That Integrates Android Application With Google Scripts, Solves Different Problems Of Emergency Character

Tamara Luarasi
Information Technology, Mathematics, Statistics Department
European University of Tirana

Abstract The several Cloud technologies, and their services extends the space of problems that can be solved by them in different fields. The problems are different, but often they represent the same scenario from the perspective of their solution by the cloud technologies and their integrations in different applications. Many situations can be adapted to this scenario: some users send their data into a central Google spreadsheet and a manager follows this spreadsheet. There are specific situations where we are not much interested on the storage of the data, but on the momentarily evidence of them and getting some emergency conclusions.
An approach is adapted to this scenario. The approach essence is: an Android Application calls a Google script. The Google script plays the role of a web application, which provides the communication of the Android application with the Google server making possible to send some data on a spreadsheet from the Android user interface.

Keywords: Android applications, Google scripts, Cloud technologies, Google applications

1 Introduction

There are several technologies today that manage different aspects of the web applications and there are specific conditions and requests that determine which of them we need to use in a specific concrete problem.
Let assume that we need to create a work environment where many users are writing the same structure of data into the same Google spreadsheet, and the manager in the same time is following the spreadsheet results being able to judge about the rhythm of everybody work, without any operation from his own side. These kinds of problems are not conditioned by a high data security, because they have a character of a momentarily use in the sense that data are not aimed to be stored in one databas e. A big advantage is that there is not a cost for this communication, because the Google spreadsheet offers a free data storage.
There are different situations in the real life that would ask this kind of work environment.
Let mention some of them:
Emergency team sends systematically information of a seek person from far distances and this information is followed by qualified doctors in a Google hospital spreadsheet.
The teacher organizes the tests in distance and the answers are sent into the Goo gle spreadsheet.
The sales persons are selling the products and some info about their sales activity is sent into a central Google spreadsheet.
Monitoring the spreadsheet is very efficient in the sense that enriching it preliminarily with some graphs and macros, we can have an idea about the team work and different aspects of the work subject in every moment. Another advantage if the fact that this work is offered freely
This idea is achieved integrating an Android Application, and a Google script as a web application.

2 The scenario

There are different scenarios that we can use to be adapted to these situations and to solve them, but we have used the following one:

- A Google apps script is created in Google script console, which provides the writing of the information into a Google spreadsheet.

- An Android Application is created, which activates the script.
- The integration of the Google+ Sign-in with Android is established to make the application usable by the other users.

3 Google script

A Google script is a JavaScript in Cloud.[1] If we would like to manage a spreadsheet from a Google script we use the following format:

```
function doGet() {
  var start = new Date();
  . . .
  var doc = SpreadsheetApp.openById('the spreadsheet key ');
  var sheet = doc.getSheetByName('firstsheet');
  . . . .
  var end = new Date();
  Logger.log("time took: " + (end.getTime() - start.getTime()));
}
```

where 'the spreadsheet key ' is the part of a the Google URL spreadsheet between https://docs.google.com/spreadsheet/ccc?key and the first character &. The be able to deploy a Goggle script as a web application we have to include the function doGet() or doPost() inside this script. The process of the deployment into a web application generates a URL address for this application which can be used in different applications. Of course this URL address would be associated with some parameters and these parameters in our case are going inside a Google script. The idea is that the set of the parameters e that comes here are considered as elements of an object e.parameter and we can get a specific one by different syntaxes like e.parameter.parameter_name, or e.parameter[i], or e.parameter[‘parameter_name’].

```
function doGet(e) {
  . . . .
  var p = e.parameter.parameter_name;
  . . .
}
```

The can make some controls of the parameters before adding them into the spreadsheet. In the code bellow there are considered two parameters one integer and one a string.

```
function doGet(e) {
  . . . .
  var value1 = 0;
  if(e.parameter == undefined || e.parameter.param1 == undefined){
    value1 = -1;
  }else{
    value1 = parseInt(e.parameter. param1);
  }
  var value2 = 0;

  if(e.parameter == undefined || e.parameter.param2 == undefined){
    value2 = 1;
  }else{
    value2 = e.parameter.param2;
  }. . .
  sheet.appendRow([value1,value2 ,. . .]);
. . . }
```

What we need in our approach is to execute it from an Android application. In this case the script will execute under the identity of the active user who is accessing the script.

4 Android Application

The Android application, will activate the Google script, managing in this way a background process, which is the connection with the Internet and sending data into a Spreadsheet. For that reason we are using the class AsyncTask. The AsyncTask class encapsulates the creation of a background process and the synchronization with the main thread, which is user interface. [2]

The following lines create an object of the class AsyncTask and the method execute in the line task.execute((Object) s); run the method doInBackground() and where s or param[0] is the script URL associated with the parameters.

```
AsyncTask task = new AsyncTask() {
@Override
 protected Object doInBackground(Object... params) {
  . . .
  try {
          . . .
          DefaultHttpClient clientscript = new DefaultHttpClient();
          HttpGet httpget = new HttpGet(params[0].toString());
          clientscript.execute(httpget);
  . . .
  } catch (IOException e) {
           e.printStackTrace();
          }
          task.execute((Object) s);
```

4.1 Android Authentication

The aim of the work presented here after all, is to make the Android application available for the other users. This would need the integration of the Google+ features in our application. There are these feature that makes the other users with their Google credentials to be able to use our application . [3][4]

Two sides are involved in this process. One is getting started with the Google+ Platform for Android.

Here are two steps to start with the Google+ Platform: [5]

- **Enable the Google+ API**: a **Google project** is created on Google API Console, the service **Google+API** in service pane is activated and an **OAuth 2.0 client ID** is created on API Access pane. The creation of OAuth 2.0 client ID, some info is required like: a **product name**, **Installed option** as the aplication type and the **package name**, which is the package name of the Android application that we develop in Eclipse.

The Android system requires that all installed applications be digitally signed with a certificate whose private key is held by the application's developer. The Android system uses the certificate as a means of identifying the author of an application and establishing trust relationships between applications. [6] For this reason we run the following command in a terminal:

keytool -exportcert -alias androiddebugkey -keystore *~/.android/debug.keystore.*

To get the SHA-1 fingerprint of the certificate. For the debug.keystore, the password is android.

- **Configure the Eclipse project**

In Eclipse we have to import and provide the reference to Google Play services library project and set up the Java build path and libraries. The following are needed:

1. Launch Eclipse.
2. Select **File** > **Import** > **Android** > **Existing Android Code Into Workspace** and click **Next**.
3. Select **Browse...**. Enter <android-sdk-folder>/extras/google/google_play_services/libproject.

The **other side** of the Android authentication is the management by code of the Google+ features .

The life cycle of an activity is managed via the following *call-back* methods, defined in the Activity base class:

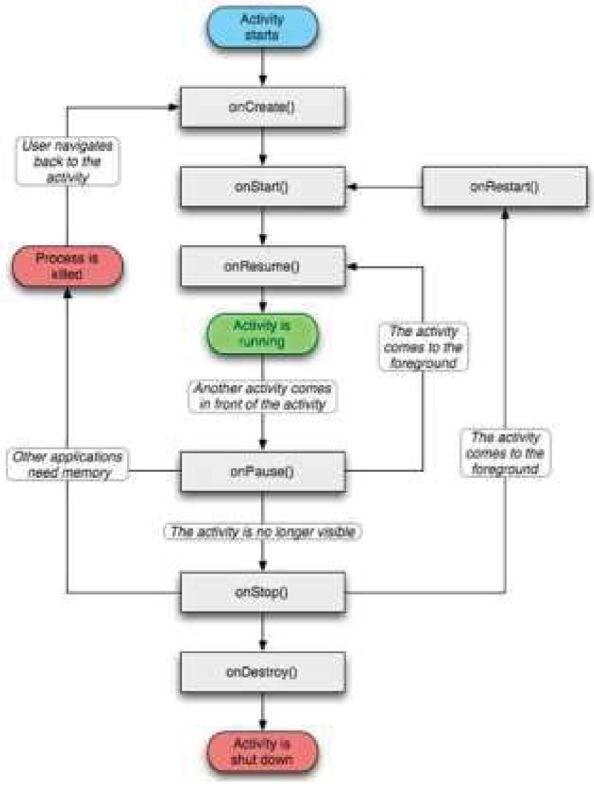
Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Fig. 1 The life cycle of an activity

The essence of the Google+ features is the management of a PlusClient object , which is used to communicate with the Google+ service and becomes functional after an asynchronous connection has been established with the service.

For this the MainActivity in Android application implements some interfaces like below and some of the properties are:

```
public class MainActivity extends Activity implements
  ConnectionCallbacks, OnConnectionFailedListener, OnClickListener,
OnAccessRevokedListener{
  . . .
    private static final int OUR_REQUEST_CODE = 49404;
        private PlusClient client;
```

```
            private  ConnectionResult connectionresult;
            private  boolean  flag;
      . . .
}
```

where the property flag is used to stop multiple dialogues appearing for the user.

The basic life cycle of the PlusClient looks a bit like this: [7]

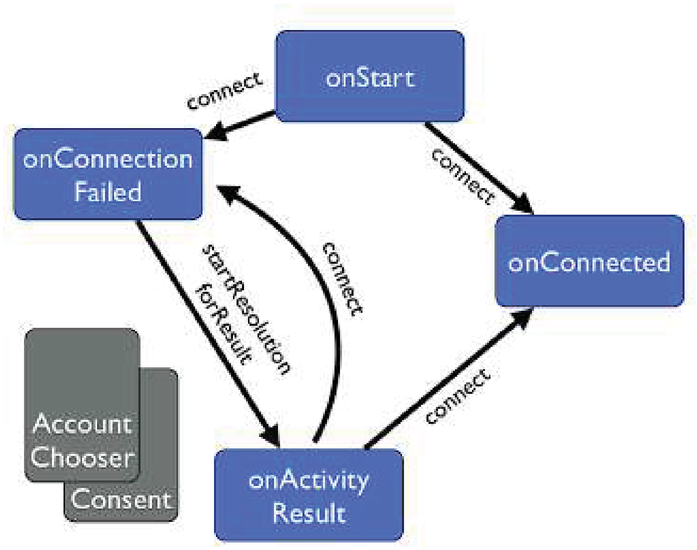

Fig. 2 The basic life cycle of the PlusClient.

Considering two pictures the code that manages the PlusClient object in MainActivity for some of the boxes is as below[8]:

onCreate() - the object of PlusClient is created and initialized. The user interface is managed too and one of buttons is so called the Google+ Sign-In button, that we add in application layout.[5]

```
<com.google.android.gms.common.SignInButton
                                                  android:id="@+id/sign_in_button"
                                              android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```
@Override

protected  void  onCreate(Bundle  savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
            . . .

  client = new  PlusClient.Builder(this,   this, this).build();
  flag = false;

  findViewById(R.id.sign_in_button).setOnClickListener(this);
  findViewById(R.id.logout).setOnClickListener(this);
  findViewById(R.id.revoke).setOnClickListener(this);
}
```

onStart() runs after onCreate()to establish the user connection

```
@Override
protected void onStart() {
            super.onStart();
            client.connect();
}
```

If the connection is successful the method onConnected() is activated. In this method beside other tasks, like sending our data into the spreadsheet we can think to get a token from the server.
A token is a packet of data created by server, and contains information to identify a particular user and token validity. The token will contain the user's information , as well as a special token code that user can pass to the server with every method that supports authentication, instead of passing a username and password directly.
Token-based authentication is a security technique that authenticate the user who attempts to log in to a server, a network, or some other secure system, using a security token provided by the server.[9]
An authentication is successful if a user can prove to a server that he or she is a valid user by passing a security token. The service validates the security token and processes the user request, but this aspect is not treated in this paper.

```
@Override
public void onConnected(Bundle bundle) {
  . . .
  flag = false;
  final Context context = this.getApplicationContext();
  . . .
  AsyncTask task = new AsyncTask() {
  @Override
    protected Object doInBackground(Object... params) {
     String scope = "oauth2:" + Scopes.PLUS_LOGIN;
     try {
            String token = GoogleAuthUtil.getToken(context,
                                        client.getAccountName(), scope);
            . . .
     } catch (UserRecoverableAuthException e) {
             e.printStackTrace();
     } catch (GoogleAuthException e) {
             e.printStackTrace();
     }
     return null;
    }
};
  task.execute();
}
```

If the connection is unsuccessful the method onConnectionFailed () is activated with a parameter result, which represents a connection status. In this method it is an opportunity to resolve any connection. We save this connection status in a member variable and invoke it by calling ConnectionResult.startResolutionForResult() when the user presses the sign-in button or +1 button.

```
@Override
public void onConnectionFailed(ConnectionResult result) {
  if (result.hasResolution()) {
    connectionresult = result;
    if (flag) {
            startResolution();
    }
  }
}

private  void  startResolution() {
  try {
    flag = false;
    connectionresult.startResolutionForResult(this, OUR_REQUEST_CODE);
  } catch  (SendIntentException  e) {
    client.connect();
  }
}
```

startResolutionForResult() activates onActivityResult()as we see it in fig. 2, which establishes the connection.

```
protected  void  onActivityResult(int requestCode, int
    responseCode,Intent intent) {
  setResult("ActivityResult: " + requestCode);
  if (requestCode == OUR_REQUEST_CODE  && responseCode == RESULT_OK) {
    flag = true;
    client.connect();
  } else if (requestCode == OUR_REQUEST_CODE  &&
                              responseCode != RESULT_OK) {
     setResult("ActivityResult: " + requestCode);
  }
}
```

The user interface of the Android application is managed by some buttons. Being associated by listeners the method onClick() is available and we can do different tasks.

```
@Override
public  void  onClick(View   view) {
  switch (view.getId())  {
  case R.id. sign_in_button:
    if (!client.isConnected()) {
            flag = true;
            if (connectionresult != null) {
              startResolution();
    } else {
            client.connect();
    }
. . . .
```

When client is now disconnected, and access has been revoked (and this can provoked by a button), we should now delete any data we need to comply with the developer properties. To reset ourselves to the original state, we should now connect again. This is provided by the method onAccessRevoked()

```
. . .
client.clearDefaultAccount();
client.revokeAccessAndDisconnect(this);
. . .

@Override
public void onAccessRevoked(ConnectionResult status) {
 client.connect();
                          . . .
}
```

5 Conclusions:

The wide development of the new technology makes us to think how to integrate them in an efficient way to solve specific problems in different fields.
There are situations where the data storage is not much important as would be an emergent evidence of the data that comes from users in distance.
An approach is proposed in this paper. The users send the data from their smart-phones into a central spreadsheet. This data immediately can be elaborated by the spreadsheet tools like charts, or various macros. Some advantages justify this approach and one of them is the cost. The Google spreadsheet use is free.
The integration of an Android application and a Google script implements this approach.
The two sides of the process of how to make this application usable by the other users is described, both from Google side and development side of the application.

References:

1. https://developers.google.com/apps-script/
2. Lars Vogel, 2013 Android Background Processing with Handlers and AsyncTask and Loaders - Tutorial
3. https://developers.google.com/+/
4. https://developers.google.com/+/mobile/android/sign-in
5. https://developers.google.com/+/mobile/android/getting-started
6. https://developer.android.com/tools/publishing/app-signing.html
7. http://www.riskcompletefailure.com/2013/03/common-problems-with-google-sign-in-on.html
8. https://gist.github.com/ianbarber/5170508
9. Rajkumar Singh, Abhinav Sonker,Authentication 2012, Protocol in Different Scenarios