Getting data from SAMI

SAMI Lecture 1
CONTENTS

Our class plan

Analyze sample codes

Assignment
Our class plan
Our class plan

- **5/9**
  Analyze sample codes. Getting data from SAMI (not real time)

- **5/16**
  Getting data from SAMI (real time)

- **5/23**
  Ordering devices or sensors to do any action.

- **5/30**
  Custom sensor, Theoretical knowledges

- **6/6**
  Final presentation
Before start

- SAMI system is renamed as "artik".
  User portal and developer portals are joined.
  New portal address: https://artik.cloud/my

- Use Samsung account as before. This is not changed.

- You can self-study with tutorial or samples at
  https://developer.artik.cloud/documentation/introduction

- Some other changes will be touched in lecture.
Analyzing sample code
Make sample application

- Get tutorial project from comnet homepage. Or you can use the project which you used at previous class (4/18).

- This sample project is almost made with preset library. I recommend you use this project as base for your final project. Or you should set some configure for SAMI SDK.
Made with two activities

• **Main Activity**
  - This activity works for the login system.
  - Almost the login process is run on the webserver. So this activity just shows a webview with an authentication page.
  - By the authentication process, we can get an access token.
  - Load the message activity with that access token.

• **Message Activity**
  - This activity works for sending data and getting data with the SAMI platform.
  - This app sends data as the SAMI Gear Fit device.
  - And we receive data when we want.
Main Activity

```
button.setOnClickListener((v) -> {
  try {
    Log.v(TAG, "button is clicked.");
    loadWebView();
  } catch (Exception e) {
    Log.v(TAG, "Run into Exception");
    e.printStackTrace();
  }
});
```

1. When button is clicked, load authentication page in webview.
Main Activity

```java
String url = getAuthorizationRequestUri();
Log.v(TAG, "webview loading url: " + url);
mWebView.loadUrl(url);
}

public String getAuthorizationRequestUri() {
  return SAMI_AUTH_BASE_URL + "/authorize?client=mobile&response_type=token&" + 
      "client_id=" + CLIENT_ID + "&redirect_uri=" + REDIRECT_URL;
}
```

2. Load authorization webpage.
Main Activity

```java
mWebView.setWebViewClient(new WebViewClient() {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String uri) {
        if (uri.startsWith(REDIRECT_URL)) {
            // Extract OAuth2 access_token in URL
            String[] sArray = uri.split("&");
            for (String paramVal : sArray) {
                if (paramVal.indexOf("access_token=") != -1) {
                    String[] paramValArray = paramVal.split("access_token=");
                    String accessToken = paramValArray[1];
                    startActivity(accessToken);
                    break;
                }
            }
        }
        return true;
    }
    // Load the web page from URL (login and grant access)
    return super.shouldOverrideUrlLoading(view, uri);
});
```

3. Set `shouldOverrideUrlLoading()` method to `webView`

**shouldOverrideUrlLoading()**

Give the host application a chance to take over the control when a new url is about to be loaded in the current WebView.
Main Activity

mWebView.setWebViewClient(new WebViewClient() { 
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String uri) {
        if (uri.startsWith("REDIRECT_URL")) {
            // Redirect URL has format http://localhost:8000/samidemo/index.php?expirea_in=1209600&token_type=bearer&access_token=xxxx
            // Extract OAuth2 access_token in URL
            String[] sArray = uri.split("&");
            for (String paramVal : sArray) {
                if (paramVal.indexOf("access_token=") != -1) {
                    String[] paramValArray = paramVal.split("access_token=");
                    String accessToken = paramValArray[1];
                    startMainActivity(accessToken);
                    break;
                }
            }
        }
        return true;
    }
    return super.shouldOverrideUrlLoading(view, uri);
});

3. Set shouldOverrideUrlLoading() method to webview

AccessToken
OAuth 2.0 is a protocol that allows external applications to request access to private data in a user’s ARTIK Cloud account without getting their passwords.
1. Setup basic configure for SAMI API.
Message Activity

```java
class CallUsersApiInBackground extends AsyncTask<UsersApi, Void, UserEnvelope> {
    final static String TAG = "CallUsersApiInBackground";

    @Override
    protected UserEnvelope doInBackground(UsersApi... apis) {
        UserEnvelope retVal = null;
        try {
            retVal = apis[0].self();
        } catch (Exception e) {
            Log.v(TAG, "::doInBackground run into Exception");
            e.printStackTrace();
        }

        return retVal;
    }

    @Override
    protected void onPostExecute(UserEnvelope result) {
        Log.v(TAG, "::setupUserHubApi self name = " + result.getData().getFullName());
        mWelcome.setText("Welcome " + result.getData().getFullName());
    }
}
```

2. Get user data in background and setup view with that data.
3. Send data with Message API. Tags and values are already defined.
4. Receive last data of any device.
Run your tutorial application

HelloSAMI

Please login to SAMI

login

Play with SAMI

Welcome your full name

send a message

Response: class MessageID {
mid: e62f873400784bcbd7e83d136e9afc24
}

Get the latest message

id:e62f873400784bcbd7e83d136e9afc24
data:{state=1, description=xllktrih, stepCount=4393, heartRate=110, activity=0}
Process of getting sensor data from SAMI

센서 → SAMI

센서 → 인식한 데이터

SAMI → getLastNormalizedMessages() → 디바이스
Assignment
Assignment

- We learned about getting data from SAMI in not real time way. But sometimes we need to get data in real time. Although we will learn about it next class, let’s make it in very dumb way.

- Real time event can be made with two ways. One is ‘interrupt’ and the other is ‘polling’. We will learn interrupt method in next class. So our assignment is making with polling method.
Polling

- Polling, or polled operation, in computer science, refers to actively sampling the status of an external device by a client program as a synchronous activity. Polling is most often used in terms of input/output (I/O), and is also referred to as polled I/O or software-driven I/O.

- Because we cannot make well-made polling system, we will just imitate polling system with sleep operation.
Assignment details

• Use TA's Samsung account for use real arrival sensor.
  ID: kilyongshin@gmail.com       PW: se3.skku

• There are 'SmartThings Arrival Sensor'. At each 30sec, your app should get data of arrival sensor from SAMI.

• The data is formed with one tag 'presence'.

• When you get data, keep it to use later.

• If the data is changed (= if arrival sensor recognize presence or not presence), write that data historically with timestamp.
Assignment details

Presence history
2016.05.09 13:47 present
2016.05.09 21:12 not present
2016.05.09 22:03 present
2016.05.10 02:17 not present
2016.05.10 09:08 present
2016.05.10 11:59 not present
2016.05.10 14:16 present
2016.05.10 15:22 not present
2016.05.11 10:47 present
Submit

• Submit your assignment at Icampus.

• Submission date: Until 16.05.15 23:59

• Zip your project folder and submit it.

• Rename your zip file as 홍길동_2014012345.zip
THANK YOU!!!