Software Practice 3

- Today’s lecture
- Today’s Task
- Porting Android App. in real device

Prof. Hwansoo Han
T.A. Jeonghwan Park 43
ARTIK, cloud platform for IoT

- Device-agnostic and scale to all IoT applications
Unfortunately...

- **We cannot purchase Artik devices**
  Sorry ;(
  - No one imports Samsung SmartThings device which is one of famous Artik devices family anymore

- **So, for your project assignment,**
  - you can use the IoT simulator serviced by Artik cloud
  - it allows to plan scenario that data should be generated
Communication to Artik

- Artik cloud supports two versions of communication
  - Realtime communication (TODAY)
    - Receive data actively while devices are alive
  - Interactive communication (AFTER MIDTERM)
    - Receive data when user requests some data
    - Send data/action also when user requests to do so

- Both above types use special socket named as WebSocket
Traditional HTTP

- Previous web services process HTTP requests in client that responses them

- Interactive system cannot be made easily because intermediate data between requests are independent

- Single directional communication results the system too complicated to buildup interactive web service
WebSocket

- Newly published standard named as WebSocket enables interactive communication.
- Maintains the main concept of HTTP but the protocol using socket makes them totally different.
WebSockets in Artik

- Artik cloud supports two WebSocket named as “Firehose” and “Device Channel”
- To use WebSocket, server and clients both should support WebSocket protocol
- By using WebSockets, you can set up a connection between Artik cloud and compatible devices or applications to receive and/or send messages in realtime
WebSockets in Artik

- **Firehose WebSocket (TODAY)**
  - To listen to messages sent by the source devices that the application monitors

- **Device channel WebSocket (AFTER MIDTERM)**
  - To receive messages targeted to your applications or devices
  - Allows the applications or devices to send messages back to Artik
SETUP ARTIK
Two important websites

- **Artik cloud developer portal**
  https://developer.artik.cloud
  - for creating Artik user application or device
  - check client id for generating access token

- **Artik cloud user portal**
  https://my.artik.cloud
  - for creating and managing device
  - check device id for generating access token
Sign in Artik cloud webpage

- Open developer portal
- Find “SIGN UP” button in topside of site
- Register account as you want

- After sign up, login both developer and user websites
Organize Artik team
Create your first device type

Create IoT Application in 5 Minutes
Build applications that read historical and real-time data from any connected device. ARTIK cloud services is a platform rich in APIs for managing user and device connectivity while you focus on your app.

CREATE YOUR FIRST APPLICATION
Tip: Creating an application

Integrate Device Hardware
Develop private device types on ARTIK cloud services with security built in as standard. Any data, any message, sent and received in real-time.

CREATE YOUR FIRST DEVICE TYPE
Tip: Creating a device type
Create your first device type
Set up device
Set up device (manifest)
Set up device (manifest)

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>BROWSE STANDARD FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>onFire</td>
<td></td>
</tr>
</tbody>
</table>

- Is Collection (if the field contains an array)

<table>
<thead>
<tr>
<th>DATA TYPE</th>
<th>UNIT OF MEASUREMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Type unit symbol</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

| TAGS (COMMA SEPARATED) | |
|------------------------| |

![Device setup screen](https://developer.artik.cloud/dashboard/device-types/dt714d1353a4c848a89ce657a90b6e7777/setup/manifests...)
Set up device (manifest)
Set up device (manifest)
Set up device (manifest)
Move on the user portal
Add device using device type

![Add a Device](https://my.artik.cloud/devices/new/edu.skku.swp3.fire_sensor_sim)

**Give Your Device a Name**

- **Fire Sensor on First Floor**

**Choose a Different Type**

- **Fire detection sensor (sim)**

**Unique Name**

- `edu.skku.swp3.fire_sensor_sim`
Note DEVICE_ID
Note DEVICE_ID
Move on to developer portal
Create your first application

Most important!
Create your first application

AUTH METHODS

- Client credentials
  - Server Side Application (without user interaction)
- Authorization Code
  - With Secret
    - Web Server
  - Without secret, PKCE mandatory
    - Installed Applications (Mobile, Desktop)
- Implicit
  - Client Side Application running in a browser
- Limited Input
  - Application running on limited input platform like TV, watch, etc.

AUTH REDIRECT URL

cloud.artik.example.hellocloud://oauth2callback
Set permissions

User Profile

What level of access does your app need to a user’s account?

- No Access Required
- Read Only
- Read and Write

What will your users see?

Authorize Fire Fighter to use your account?

This application will be able to

- View your basic profile information
- View your email address
- Write permissions
Set permissions

Devices
Select the Device Type your app connects to, and the level of access.

<table>
<thead>
<tr>
<th>DEVICE TYPE</th>
<th>PERMISSIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>fire detection</td>
<td>fire_detection_sensor (sim) eduaskku.awsp.3.fire_sensor_sim</td>
</tr>
</tbody>
</table>

Device Type vs Device?
A Device Type defines a category of Devices in ARTIK cloud services. Learn more

Your users will need to accept and authorize the permission you requested for every Device of selected Device Type(s) in this section.
Set permissions

Devices

Select the Device Type your app connects to, and the level of access.

<table>
<thead>
<tr>
<th>DEVICE TYPE</th>
<th>PERMISSIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fire detection sensor (sim)</td>
<td>Read and Write</td>
</tr>
</tbody>
</table>

Device Type vs Device?

A Device Type defines a category of Devices in ARTIK cloud services. Learn more.

Your users will need to accept and authorize the permission you requested for every Device of selected Device Type(s) in this section.
Move on to overview page
Note CLIENT ID
HOW TO GET MESSAGE FROM ARTIK IN ANDROID
Get tutorial application

- Get tutorial project from course webpage
- This sample project is made with Artik library. I recommend you to use this project as base for your final project if you plan to use IoT devices
Customize your application

```java
class Config {
    static final String CLIENT_ID = "your client id";
    static final String DEVICE_ID_FIRE_SENSOR = "your device id";

    // MUST be consistent with "AUTH REDIRECT URL" of your application
    // set up at the developer.artik.cloud
    static final String REDIRECT_URI = "cloud.artik.example.hellocloud://oauth2callback";
}
```

- Configure custom information in Config.java
- CLIENT_ID & DEVICE_ID must be set to your own configurations
- REDIRECT_URI must be fixed to given value
Add some configurations

```java
private static final String TAG = "MessageActivity";

private UsersApi mUsersApi = null;

private String mAccessToken;

private String userId;

private Button openFirehoseButton;

private TextView fireSensorText;
```

- Add some components to use in Message.java
- Button and TextView is already defined in layout xml file
Set listener to added button

```java
openFirehoseButton = (Button) findViewById(R.id.listen_button);
fireSensorText = (TextView) findViewById(R.id.fire_sensor_response);

openFirehoseButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Log.v(TAG, "listen button is clicked.");
        try {
            connectFirehoseWebSocket(DEVICE_ID_FIRE_SENSOR, R.id.fire_sensor_response);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});

openFirehoseButton.setClickable(false);
```

- For stability of application, we set connectFirehose button not-clickable for now
- Because FirehoseWebSocket needs user information
- We will define this function later
Get user identification

private void getUserInfo() {
    final String tag = TAG + " getSelfAsync";
    try {
        mUsersApi.getSelfAsync(new ApiCallback<UserEnvelope>() {
            @Override
            public void onFailure(ApiException exc, int statusCode, Map<String, List<String>> map) {
                processFailure(tag, exc);
            }
            
            @Override
            public void onSuccess(UserEnvelope result, int statusCode, Map<String, List<String>> map) {
                Log.v(TAG, "getSelfAsync::setupArtikCloudApi self name = " + result.getData().getFullName());
                updateWelcomeViewOnUiThread("Welcome " + result.getData().getFullName());
                userId = result.getData().getId();
            }
        });
    }
    // ...

    onSuccess() function is called when API gets data normally
    When getting user data, we store userId for initializing FirehoseWebSocket
Set connect button clickable

```java
private void updateWelcomeViewOnUiThread(final String text) {
    this.runOnUiThread(new Runnable() {
        @Override
        public void run() {
            fireSensorText.setText(text);
            openFirehoseButton.setClickable(true);
        }
    });
}
```

- In Android, UI related works should be processed on UI thread which is also called as main thread
- runOnUiThread allows to process such a works in main thread
Connect Firehose WebSocket

- OkHttp is very useful library for establishing HTTP connection in JAVA

```java
private void connectFirehoseWebSocket(final String device_id, final int textId)
    throws Exception {
    OkHttpClient client = new OkHttpClient();
    client.retryOnConnectionFailure();
    FirehoseWebSocket ws = new FirehoseWebSocket(client, mAccessToken, device_id,
        null, null, userId, new ArtikCloudWebSocketCallback() {
            @Override
            public void onOpen(int httpStatus, String httpStatusMessage) {
                Log.d(TAG, "onOpen");
            }
        });
}
```

- User id and others are used to initialize FirehoseWebSocket
Connect Firehsot WebSocket

```java
@Override
public void onMessage(MessageOut message) {
    Log.d(TAG, "onMessage");
    Map<String, Object> data = message.getData();
    StringBuilder sb = new StringBuilder();
    for (String key : data.keySet()) {
        sb.append(key);
        sb.append(" : ");
        sb.append(data.get(key));
        sb.append("\n");
    }
    //TextView text = (TextView) MessageActivity.this.findViewById(textId);
    //text.setText("Listened data : " + sb.toString());
    updateListenedResponseOnUIThread(sb.toString(), textId);
}
```

- When monitoring device send message, `onMessage()` method is called
- We can get data by `getData()` method
- `updateListenedResponseOnUIThread` will be defined later
public void onAction(ActionOut action) {
    Log.d(TAG, "onAction");
}

public void onAck(Acknowledgement ack) {
    Log.d(TAG, "onAck");
}

public void onClose(int code, String reason, boolean remote) {
    Log.d(TAG, "onClose");
}

public void onError(WebSocketError error) {
    Log.d(TAG, "onError : " + error.getMessage());
}
Connect Firehose WebSocket

- Artik server send ping to application in every 30 seconds
- When your application receive ping, `onPing()` method will be called

```java
@Override
public void onPing(long timestamp) {
    Log.d(TAG, "onPing");
}
ws.connect();
```

- `connect()` method of WebSocket instance will establish connection between mobile application and Artik cloud
private void updateListenedResponseOnUiThread(final String response, final int textId) {
    this.runOnUiThread(new Runnable() {
        @Override
        public void run() {
            TextView listenedText = (TextView) findViewById(textId);
            listenedText.setText("Listened Data : " + response);
        }
    });
}

- As mentioned in previous slide, every UI process should be processed in main thread
- Also updating the content in TextView component should be done in the same way
Simulate Artik devices

- You have to start simulator before running your application
Sample screen on mobile phone
Develop Artik application & mobile application

- Consists of Artik application
  - Fire detection and water spray system
  - 2 Fire detection sensors – virtually on 1st, 2nd floor
    - (Field) onFire – boolean; set as true while detecting fire
  - 2 Water spray machine – virtually on 1st, 2nd floor
    - (Field) status – String; set “On” while spraying water else “Off”
[Lab – Practice #5]

- Group fire detectors and water spray machines on the same floor respectively
  - if status of any device changed on the same floor, update TextView like given examples represented in next slide
[Lab – Practice #5]

Play with ARTIK Cloud

Get message from Firehose

Listened message:
1st floor: in flame and water spray does not work
2nd floor: no fire and water spray is now working

Play with ARTIK Cloud

Get message from Firehose

Listened message:
1st floor: no fire and water spray does not work
2nd floor: no fire and water spray is now working
Current version of Artik application

- FirehoseWebSocket establish new connect even though connection already exist
  - Make application use existing connection, which means that resolve this problem without disable clickable
  - when button clicked again, print “connection already established” with Toast message
    - Toast.makeText(getApplication(), “connection already established”, Toast.LENGTH_LONG).show();
  - Don’t copy this code, \“ and \” character cannot be compiled!