

How to interfacing with IPS: Android Listener Programming

(IPS: Indoor Positioning System)

By Jung-Wook Moon

Pusan National University, Busan, South Korea

Nov 19, 2020

A decorative graphic consisting of a grid of blue squares in various shades, arranged in a stepped pattern on the left side of the slide.

Create a Listener

Create a Listener

- Declaring the event listener interface
 - It is okay to write parameters to be processed when an event is received in the function in the interface.
- Register event listener as a variable

```
public interface SampleEventListener{  
    void onReceivedEvent();  
}
```

These source code shows how to declare a general event listener interface.

You just need to include the formal declaration of the function in the interface.

```
private SampleEventListener mSampleEventListener;
```

This source code shows how to declare an event listener.

It is declared as a member variable in a specific class.

Create a Listener

- Create a set function so that it can be registered externally

```
public void setOnSampleReceivedEvent(  
    SampleEventListener listener){  
    mSampleEventListener = listener;  
}
```

These source code shows how to define and register callback function in listener.

- Event call

```
mSampleEventListener.onReceivedEvent()
```

This source code shows how to call an event using a member function of a listener.

Create a Listener

- Event reception

```
CustomView view = new CustomView(this);
```

```
view.setOnSampleReceivedEvent(  
    new CustomView.SampleEventListener(){  
        @Override  
        public void onReceivedEvent(){  
            // Codes for Event Reception  
        }  
    });
```

These source code defines what action the listener module should do when an event occurs.

You can use this code format to get information from the listener module.

Using Listener

Showing a Example: Using Coordispace Positioning Module

Using Listener Example

```
package com.coordspace.ips_sample_app;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

/* Import Code for Indoor Positioning System SDK : Start */
import com.coordspace.listener.PositionDetailListener;
import com.coordspace.listener.StatusListener;
import com.coordspace.main.PermissionActivity;
import com.coordspace.main.Status;
import com.coordspace.main.ServiceManager;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.widget.TextView;
/* Import Code for Indoor Positioning System SDK : End */

public class MainActivity extends AppCompatActivity {

    /* Variable for IPS SDK : Start */

    // IPS ServiceManager Variables
    private ServiceManager mServiceManager;
    private final int mRequestCode = 111;
    private Context mContext;

    // Server URL and Map(Service number) Setting Variables
    private static final String SERVICE_URL
        = "http://13.125.45.3:8080/poinsAnalytics_api/";
    private static final int[] SERVICE_NO = new int[] {
        168, 171, 173
    };
};
```

Activity class is a class that must be included in Android apps.

In this class, a listener variable is declared, and information about the status code and location sub is stored in member variables. mServiceManager stores listener's instance. mRequestCode is used to check the state of the positioning system.

SERVICE_URL and SERVICE_NO represent the URL and service area of the location server.

Using Listener Example

```
// Position Variables
int mStatusCode, mMapIdx;
float mX, mY;
int mLayer;
float mHeading, mPoinsSimilarity, mRollPitchYaw[];

// TextView Variable
TextView MapIDDData, LayerIDDData, PositionXData, PositionYData;
TextView HeadingData, YawAngleData, IPSStatusCode;
/* Variable for IPS SDK : End */

@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
}

/* IPS Service Start/Spot Control Code for IPS SDK : Start */
@Override
protected void onActivityResult(int requestCode, int resultCode,
    Intent data ) {
    ...
}

@Override
protected void onDestroy() {
    super.onDestroy();
    mServiceManager.stopService();
}
/* IPS Service Start/Spot Control Code for IPS SDK : End */
}
```

In the onCreate function, you define how to check the status of most positioning listeners, and how to handle positioning results. Therefore, onCreate function contains the core contents of Listener usage.

The onActivityResult function is related to this app's activity state.

onCreate()

Deliver the URL of the server system to download the fingerprint map and the ID of the service area.

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

    /* Interface Code for IPS SDK : Start */
    mContext = this;
    mServiceManager = ServiceManager.getInstance(this);
    mServiceManager.setBaseServiceInfo(SERVICE_URL, SERVICE_NO);
```

```
MapIDData = findViewById(R.id.MapIDData);
LayerIDData = findViewById(R.id.LayerIDData);
PositionXData = findViewById(R.id.PositionXData);
PositionYData = findViewById(R.id.PositionYData);
HeadingData = findViewById(R.id.HeadingData);
YawAngleData = findViewById(R.id.YawAngleData);
IPSStatusCode = findViewById(R.id.IPSStatusCode);
```

```
mServiceManager.setDetailListener(new PositionDetailListener() {
    @Override
    public void onPosition(int mapIdx, float x, float y, int layer,
        float heading, float pointsSimilarity,
        float[] rollPitchYaw) {
        mMapIdx = mapIdx;
        mX = x;
        mY = y;
        mLayer = layer;
        mHeading = heading;
        mPointsSimilarity = pointsSimilarity;
        mRollPitchYaw = rollPitchYaw;
        runOnUiThread(new Runnable() {
            @SuppressWarnings("DefaultLocale")
            @Override
            public void run() {
                // Using Positioning Data

                /* TextView Code 3 for Sample App : Start */
                MapIDData.setText(String.format("%d", mMapIdx));
```

2 Types of Listeners:

- **PositionDetailListener**
- **StatusListener**

Two types of callback functions must be registered with the listener.

```
LayerIDData.setText(String.format("%d", mLayer));
PositionXData.setText(String.format("%5.2f", mX));
PositionYData.setText(String.format("%5.2f", mY));
HeadingData.setText(String.format("%5.2f", mHeading));
YawAngleData.setText(String.format("%5.2f",
    mRollPitchYaw[2]*180f/Math.PI));
/* TextView Code 3 for Sample App : End */

});
});

mServiceManager.setStatusListener(new StatusListener() {
    @Override
    public void onStatus(int statusCode) {
        mStatusCode = statusCode;
        runOnUiThread(new Runnable() {
            @SuppressWarnings("DefaultLocale")
            @Override
            public void run() {
                String IpsEngineStatus =
                    mStatusCode + " : " +
                    Status.sdkStatus.get(mStatusCode);
                IPSStatusCode.setText(IpsEngineStatus);
            }
        });
    }
});

Intent intent = new Intent(mContext, PermissionActivity.class);
startActivityForResult(intent, mRequestCode);
/* Interface Code for IPS SDK : End */
}
```

onCreate()

```

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

    /* Interface Code for IPS SDK : Start */
    mContext = this;
    mServiceManager = ServiceManager.getInstance(this);
    mServiceManager.setBaseServiceInfo(SERVICE_URL, SERVICE_NO);

    MapIDData = findViewById(R.id.MapIDData);
    LayerIDData = findViewById(R.id.LayerIDData);
    PositionXData = findViewById(R.id.PositionXData);
    PositionYData = findViewById(R.id.PositionYData);
    HeadingData = findViewById(R.id.HeadingData);
    YawAngleData = findViewById(R.id.YawAngleData);
    IPSStatusCode = findViewById(R.id.IPSStatusCode);

    mServiceManager.setDetailListener(new PositionDetailListener() {
        @Override
        public void onPosition(int mapIdx, float x, float y, int layer,
            float heading, float pointsSimilarity,
            float[] rollPitchYaw) {
            mMapIdx = mapIdx;
            mX = x;
            mY = y;
            mLayer = layer;
            mHeading = heading;
            mPointsSimilarity = pointsSimilarity;
            mRollPitchYaw = rollPitchYaw;
            runOnUiThread(new Runnable() {
                @SuppressWarnings("DefaultLocale")
                @Override
                public void run() {
                    // Using Positioning Data

                    /* TextView Code 3 for Sample App : Start */
                    MapIDData.setText(String.format("%d", mMapIdx));
                }
            });
        }
    });
}

```

If you look at the structure of onCreate function, it performs the following functions.

1. Initialize mServiceManger. This allows the listener to operate in earnest by allocating and storing a new instance through the ServiceManager class.
2. The listener's first task is to connect to the positioning server, download the fingerprint map for the region (area) we want, and prepare for positioning.
3. By defining the onPoosition callback function, the detailed operation contents of how to process and process the location information for the listener is determined.
4. By defining the onStatus callback function, the location status of the listener is reported every moment and detailed action details to cope with it are determined.

onCreate()

```

        LayerIDData.setText(String.format("%d", mLayer));
        PositionXData.setText(String.format("%.2f", mX));
        PositionYData.setText(String.format("%.2f", mY));
        HeadingData.setText(String.format("%.2f", mHeading));
        YawAngleData.setText(String.format("%.2f",
            mRollPitchYaw[2]*180f/Math.PI));
        /* TextView Code 3 for Sample App : End */
    }
});
});
});

mServiceManager.setStatusListener(new StatusListener() {
    @Override
    public void onStatus(int statusCode) {
        mStatusCode = statusCode;
        runOnUiThread(new Runnable() {
            @SuppressWarnings("DefaultLocale")
            @Override
            public void run() {
                String IpsEngineStatus =
                    mStatusCode + " : " +
                    Status.sdkStatus.get(mStatusCode);
                IPSSStatusCode.setText(IpsEngineStatus);
            }
        });
    }
});
});

Intent intent = new Intent(mContext, PermissionActivity.class);
startActivityForResult(intent, mRequestCode);
/* Interface Code for IPS SDK : End */
}

```


If you look at the structure of onCreate function, it performs the following functions.

1. Initialize mServiceManger. This allows the listener to operate in earnest by allocating and storing a new instance through the ServiceManager class.
2. The listener's first task is to connect to the positioning server, download the fingerprint map for the region (area) we want, and prepare for positioning.
3. By defining the onPoision callback function, the detailed operation contents of how to process and process the location information for the listener is determined.
4. By defining the onStatus callback function, the location status of the listener is reported every moment and detailed action details to cope with it are determined.

onActivityResult()

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
    Intent data ) {
    super.onActivityResult(requestCode, resultCode, data);

    if( requestCode == mRequestCode) {
        if( resultCode == RESULT_OK ) {
            mServiceManager.startService();
        }
        else {
            finish();
        }
    }
}
```



This source code is the code for the Android App Activity. This code can be used as it is without any modification.

To understand this part, it is necessary to understand the activity status of the android application, and if you have an understanding of it, you can understand it without a separate explanation.