● 브로드캐스트 리시버: 이벤트 모델로 수행되는 컴포넌트. 특히 시스템의 특정 상황을 알려야 할 때 많이 사용된다 (ex. SNS, bluetooth, wifi, screen on/off, etc). . 안드로이드 컴포넌트 중 엑티비티, 서비스, 브로드캐스트 리시버는 인텐트에 의해 실행된다.

자바 코드에서 인텐트를 준비하고 시스템에 의뢰하면 시스템에서 인텐트 정보에 맞는 컴포넌트를 실행하는 구조이다. 그런데 액티비티, 서비스, 브로드캐스트 리시버의 인텐트 동작 원리가 각각 다르다. 액티비티를 실행하는 인텐트에서는 시스템에 발생한 인텐트로 인해 실행할 액티비티가 하나일 때는 해당 액티비티가 정상적으로 실행되고, 맞는 액티비티가 하나도 없을 때는 인텐트가 발생한 곳에서 에러가 생기고, 인텐트를 발생하였는데 실행할 액티비티가 두 개 이상이라면 사용자 선택으로 하나만 실행된다.

반면, 브로드캐스트 리시버의 동작 원리는 다르다. 인텐트가 발생할 때 실행될 브로드캐스트 리시버가 없다고 하더라도 에러가 발생하지 않으며, 브로드캐스트 리시버가 여러 개라면 모두 실행된다. "없으면 말고, 있으면 모두 실행하고." 따라서 흔히 브로드캐스트 리시버를 이벤트 모델로 수행되는 컴포넌트라고 부른다 (시스템에서 이벤트 상황이 발생했을 때 없으면 말고, 있으면 모두 실행하는 브로드 캐스트 리시버).

예를 들면 스마트폰의 배터리가 부족한 상황이 되면 시스템에서 브로드캐스트 리시버를 실행하기 위한 인텐트가 발생하고, 이 인텐트로 인해 각 앱의 브로드캐스트 리시버가 실행되어 앱에 배터리가 얼마 남지 않았음을 알려 준다. 또한 외부 앱 연동 시 여러 앱에서 특정 인텐트에 반응할 브로드캐스트 리시버를 만들 수도 있는데, 대표적으로 부팅 완료 시점에 시스템에서 발생시키는 브로드캐스트 인텐트의 경우, 런처 앱의 브로드캐스트 리시버가 실행되고 그 리시버가 인텐트로 액티비티를 실행하여 런처 화면이 나옴과 동시에, 카톡처럼 앱을 실행하지 않아도 부팅 완료와 동시에 서버에 접속해야 채팅을 제공할 수 있는 서드파티 밴드의 앱들에도 부팅 완료 시점에 동작할 브로드캐스트 리시버를 만들어 놓으면 부팅이 완료될 때 실행된다. 이처럼 하나의 인텐트 발생으로 여러 앱이 함께 동작해야 하는 상황에서 브로드캐스트 리시버가 사용된다.

브로드캐스트 리시버는 특정한 목적의 업무를 담당하기 위해 정의된 컴포넌트가 아니다. 인텐트로 인 해 한 번 생성되어 실행된 브로드캐스트 리시버는 10초 이내에 업무 처리가 종료되어야 한다는 규칙이 있기 때문에, 시간 제약도 있고 메모리에 지속하지도 않으므로 대단한 업무를 진행할 수 없다. 단지 브로드캐스트 리시버는 이벤트 모델(없으면 말고, 있으면 모두 실행하고)로 수행되는 컴포넌트가 필요해서 만들어놓은 것이다. 대부분 간단한 업무만 처리하거나 아니면 브로드캐스트 리시버에서 다른 액티비티나 서비스를 실행하는 역할로만 사용된다.

▶ 브로드캐스트 리시버 작성 방법: 브로드캐스트 리시버는 BroadcastReceiver을 상속받아 작성하는 클래스이며, 이 클래스 내에 onReceive()란 함수만 정의해 주면 된다. onReceive()는 브로드캐스트 리시버가 인텐트로 인해 수행될 때 자동으로 호출되는 함수이다.

```
public class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Toast toast = Toast.makeText(context, "Im a BroadcastReceiver", Toast.Length_SHO
        toast.show();
    }
}
```

이렇게 작성된 브로드캐스트 리시버는 안드로이드 컴포넌트 클래스이므로, AndroidManifest.xml 파일에 <receiver> 태그로 등록해야 한다. <activity>와 마찬가지로 암시적 인텐트에 의해 실행되어야 한다면, <receiver> 태그 내에 <intent-filter>가 등록될 수도 있다.

```
<receiver
    android:name=".MyReceiver"
    android:enabled="true"
    android:exported="true">
</receiver>
```

이렇게 만들어진 브로드캐스트 리시버를 sendBroadcast() 함수로 인텐트를 발생시켜 실행한다. 또한, putExtra() 함수를 이용하여 데이터를 넘길 수도 있다.

```
// 브로드캐스트 리시버를 실행시킬 intent 준비
Intent intent = new Intent(this, MyReceiver.class);
// 브로드캐스트 리시버 인텐트를 발생시키는 함수 sendBroadcast
sendBroadcast(intent);
```

● 시스템 상태 파악: 브로드캐스트 리시버는 앱 내부에서 사용하려고 자주 만들지만, 시스템에서 실행하는 인텐트에 반응하여 각종 시스템 상황을 감지하려고도 자주 사용한다. 시스템에서는 많은 브로드캐스트 리시버 인텐트를 발생시킨다.

▶ 부팅 완료: 스마트폰의 부팅 완료 시점에 동작하여 간단한 업무를 수행하거나, 서비스 등을 구동해야 할 때, 부팅 완료 시점에 시스템에서 발생시키는 브로드캐스트 인텐트에 반응할 브로드캐스트 리시버만 정의해주면 된다. AndroidManifest.xml 파일에 등록할 때 시스템에서 띄우는 인텐트의 Action 문자열과 같은 문자여로 intent-filter을 구성하여 등록하면 된다. 부팅 완료 시점에 띄우는 인텐트의 action 문자열은 android.intent.action.BOOT_COMPLETED이다.

```
<receiver
    android:name=".MyReceiver"
    android:enabled="true"
    android:exported="true"
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
```

부팅 완료 시점에 브로드캐스트 리시버가 동작하게 하려면 Androidmanifest.xml에 uses-퍼미션이 등록되어 있어야 한다.

```
// AndroidManifest.xml
<uses-permission android:nmae="android.permission.RECEIVE_BOOT_COMPLETED"/>
```

▶ 화면 On/Off: 사용자 스마트폰의 화면에 on/off 되는 상황에 맞게 특정 로직을 수행하거나 정지하도록 구현해야 하는 경우는 많다. 화면에 On/Off되는 상황에 시스템에서 브로드캐스트 인텐트를 발생해 주며, 앱에서 브로드캐스트 리시버를 이용하여 이 상황을 감지할 수 있다. 화면 On/Off를 위한 브로드캐스트 리시버는 다든 브로드캐스트 리시버와는 다르게 AndroidManifest.xml 파일에 <receiver> 태그로 등록(정적 등록)하면 실행되지 않는다. 액티비티 또는 서비스 등의 코드에서 동적으로 등록해야만 실행된다 (동적 등록). (이처럼 시스템에서 강제하는 사항이 아닐 때도 자주 코드에서 동적 등록/해제를 하는데, 특정 액티비티나 서비스가 구동 중일 때만 브로드캐스트 인텐트로 인해 실행되어야 할 때이다. AndroidManifest.xml에 등록하면 인텐트가 발생하기만 하면 항상 실행되지만, 액티비티나 서비스가 구동되어 있지 않다면 의미 없을 때는 액티비티 또는 서비스의 생명주기 함수를 이용하여 등록하고 해제하는 방법을 이용한다).

```
// 액티비티나 서비스 클래스 내부의 코드. 자바 코드에서 브로드캐스트 리시버를 정의한다.
BroadcastReceiver brOn = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d("kkang", "screen on.....");
    }
};
```

자바 코드에서 브로드캐스트 리시버를 정의한 후, registerReceiver() 함수로 시스템에 등록한다. 화면 On/Off 때 시스템에서 띄우는 인텐트의 Action 문자열은 android.intent.action.SREEN_ON과 android.intent.action.SREEN_OFF이며, 이 문자열을 가지는 intentFilter 객체를 만들어 registerReceiver() 함수로 동적 등록한다.

```
registerReceiver(brOn, new IntentFilter(Intent.ACTION_SCREEN_ON));
```

이렇게 코드에서 동적으로 등록한 브로드캐스트 리시버는 또 코드에서 동적으로 등록이 해제되는데, 해제는 unregisterReceiver() 함수를 이용한다.

```
unregisterReceiver(brOn);
```

▶ 배터리: 스마트폰에 USB 케이블 등으로 전원이 공급되고 있는 상황과, 전원 공급이 끊어진 상황 등 배터리와 관련된 각종 상황을 앱에서 인지해야 할 때가 있다. 이때 시스템에서는 앱에서 이런 상황을 인지할 수 있도록 브로드캐스트 인텐트를 발생해 준다. 배터리와 관련된 브로드캐스트 인텐트의 action 문자열은 여러 가지이다.

android.intent.action.BATTERY_LOW	스마트폰의 배터리가 낮은 상태가 되었을 때
android.intent.action.BATTERY_OKAY	스마트폰의 배터리가 낮은 상태에서 벗어날 때
android.intent.action.BATTERY_CHANGED	스마트폰 배터리의 충전 상태가 변경되었을 때
android.intent.action.BATTERY_POWER_CONNECTED	케이블 등으로 외부 전원공급이 연결되었을 때
android.intent.action.BATTERY_POWER_DISCONNECTED	외부 전원공급이 끊어질 때

```
registerReceiver(batteryReceiver, new IntentFilter(Intent.ACTION_POWER_CONNECTED));
registerReceiver(batteryReceiver, new IntentFilter(Intent.ACTION_POWER_DISCONNECTED));
```

여러 action 문자열을 등록하여 실행되는 브로드캐스트 리시버에서는 onReceive() 함수에서 action 문자열을 추출하여 자신이 어떤 인텐트 정보로 인해 실행된 것인지 구분할 수 있다.

```
BroadcastReceiver batteryReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (action.equals(Intent.ACTION_POWER_CONNECTED)) {
            addListItem("ON CONNECTED....");
        } else if (action.equals(Intent.ACTION_POWER_DISCONNECTED)) {
            addListItem("ON DISCONNECTED...");
        }
    }
};
```

→ 배터리 상태 파악: 어떨 때는 특정 코드 (액티비티, 서비스)가 실행되면서 배터리 상황을 파악해야 할 때도 있다.

```
IntentFilter ifilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
Intent batteryStatus = registerReceiver(null, ifilter);
```

위의 코드도 registerReceiver() 함수로 브로드캐스트 리시버를 등록하는 구문이지만, 브로드캐스트 리시버 객체 부분이 null로 대입되었다. 실제 개발자가 준비한 브로드캐스트 리시버를 등록하는 구문이 아니라, 시스템의 배터리 상태의 정보값만 얻기 위한 구문이다. 이렇게 얻은 인텐트 객체에 다양한 배터리 정보가 담기게 된다. 가장 많이 얻는 정보가 스마트폰에 전원이 공급되고 있는지에 대한 정보이다. 다음은 현재 스마트폰에 전원이 공급되고 있는지 파악하기 위한 구문이다. isCharging 값이 true이면 전원이 공급되고 있는 상황이다.

```
int status = batteryStatus.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
boolean isCharged = status == BatteryManager.BATTERY_STATUS_CHARGING;
```

또한, 전원이 공급되고 있을 때 USB를 이용하는 건지, 아니면 AC를 이용하는 건지를 파악해야 할 때도 있다. 이처럼 전원 공급의 유형을 파악하는 구문은 다음과 같다.

```
int chargePlug = batteryStatus.getIntExtra(BatteryManager.EXTRA_PLUGGED, -1);
boolean usbCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_USB;
BOOLEAN acCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_AC;
```

이번에는 스마트폰의 배터리가 몇 퍼센트 충전된 상황인지 파악하는 구문이다.

```
int level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
int scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1);

float batteryPct = (level / (float) scale) * 100;
```

```
// new > Other > Broadcast Receiver
```

```java
// MyReceiver.java

package com.android.practicebroadcastreceiver;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "I am receiver", Toast.LENGTH_SHORT).show();
    }
}
```

```java
// MainActivity.java

package com.android.practicebroadcastreceiver;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.IntentFilter;
import android.os.BatteryManager;
import android.os.Bundle;

import com.android.practicebroadcastreceiver.databinding.ActivityMainBinding;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityMainBinding binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        // 버튼을 클릭하면 인텐트를 발생시켜 브로드케스트 리시버를 실행함
        binding.lab1Button.setOnClickListener(view -> {
            // 브로드캐스트 리시버를 발생시킬 인텐트 준비
            Intent intent = new Intent(this, MyReceiver.class);
            // 브로드캐스트 리시버 인텐트를 발생시키는 함수 sendBroadcast
            sendBroadcast(intent);
        });

        // 배터리 상태 파악
        // registerReceiver() 함수로 시스템의 배터리 상태의 정보값을 얻기 위한 구문
        IntentFilter ifilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
        Intent batteryStatus = registerReceiver(null, ifilter);

        // 스마트폰에 전원이 공급되고 있는지 파악하기 위한 구문. isCharging 값이 true이면 전원
        int status = batteryStatus.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
        boolean isCharging = status == BatteryManager.BATTERY_STATUS_CHARGING;

        // 만약 공급되고 있다면,
        if (isCharging) {
            // 전원이 공급되고 있을 때 USB를 이용하는 건지, AC를 이용하는 건지
```

```
            int chargePlug = batteryStatus.getIntExtra(BatteryManager.EXTRA_PLUGGED, -1)
            boolean usbCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_USB;
            boolean acCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_AC;

            // 만약 usb로 charge되고 있으면 (usbCharge가 true이면)
            if (usbCharge) {
                binding.lab1PluggedView.setText("USB Charging");
            // 만약 ac로 charge되고 있으면 (acCharge가 true 이면)
            } else if (acCharge) {
                binding.lab1PluggedView.setText("AC Charging");
            }
        // 전원이 공급되고 있지 않다면
        } else {
            binding.lab1PluggedView.setText("unPlugged");
        }

        // 배터리가 몇 퍼센트 충전된 상황인지 파악하는 구문
        int level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
        int scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1);
        float batteryPct = (level / (float) scale) * 100;
        binding.lab1LevelView.setText("현재 배터리 : " + batteryPct + "%");
    }
}
```

Resources: 깡쌤의 안드로이드 프로그래밍