14 브로드캐스트 리시버 컴포넌트

이 것도 시스템에서 생명주기를 관리하는 컴포넌트 클래스 4개 중 하나다. 핵심 로직 담당하지도 않고 API의 도움도 별로 없지만 독특한 동작 방식 때문에 자주 사용한다.

1. 브로드캐스트 리시버 이해하기
이는 흔히 이벤트 모델로 실행되는 컴포넌트라고 정의한다. 여기서 이벤트는 부팅이 완료되는 것과 같은 시스템의 특정 상황이다.

브로드캐스트 리시버를 줄여서 그냥 리시버라고 한다. 이를 직역하면 방송 수신기이다. 즉, 시스템에서 특정한 상황을 알리는 방송을 할 때(이벤트 발생) 이를 받아서 처리하는 수신기를 앱에 장착한다고 생각하면 쉽다. 이 수신기도 컴포넌트라서 인텐트로 시스템에 전달하여 실행한다.

- 브로드캐스트 리시버 만들기

브로드케스트 리시버를 만들려면 BroadcastReceiver를 상속받는 클래스를 선언해야 한다. 브로드캐스트 리시버의 생명주기 함수는 onReceive() 하나뿐이다. 어디선가 이 리시버를 실행하려고 인텐트를 시작하면 onReceive() 함수가 자동으로 호출된다. 그리고 자신을 호출한 인텐트 객체를 매개변수롤 전달받는다.

• 브로드캐스트 리시버 만들기

```
class MyReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
    }
}
```

• 브로드캐스트 리시버 등록

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

필수 속성

● 동적 등록과 해제

안드로이드 컴포넌트는 매니페스트에 등록해야 시스템에서 인지하고 어디선가 인텐트가 발생할 때 컴포넌트를 실행한다. 브로드캐스트 리시버도 마찬가지지만 등록하지 않고 필요한 순간 동적으로 등록할 수 있다.

다음 코드는 액티비티나 서비스 컴포넌트에서 작성한 코드입니다.

• 리시버 객체 생성

```
receiver = object : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
    }
}
```

브로드캐스트 리시버 객체는 액티비티나 서비스에서 생성할 수 있습니다. 그리고 필요한 순간에 다음처럼 registerReceiver()라는 함수를 이용해 시스템에 등록합니다.

• 동적 등록

```
val filter = IntentFilter("ACTION_RECEIVER")
registerReceiver(receiver, filter)
```

● 브로드캐스트 리시버 실행하기

이를 실행하려면 인텐트가 필요하다. 리시버의 클래스명만 등록했으면 클래스 타입 레퍼런슬ㄹ 이용해 명시적 인텐트로 실행하고, 인텐트 필터를 등록했으면 암시적 인텐트로 실행한다. 하지만 리시버는 이렇게 안된다.

브로드캐스트 리시버를 실행하는 인텐트는 sendBroadcast() 함수로 시스템에 전달합니다. 그러면 시스템은 브로드캐스트 리시버 객체를 생성하여 실행해 줍니다.

• 리시버 실행

```
val intent = Intent(this, MyReceiver::class.java)
sendBroadcast(intent)
```

그런데 브로드캐스트 리시버는 시스템에 해당 인텐트로 실행될 리시버가 없으면 아무런 일도 일어나지 않습니다. 즉, 인텐트를 시작한 곳에서 오류가 발생하지 않습니다. 또한 실행될 리시버가 여러 개 있으면 모두 실행됩니다. 즉, '없으면 말고 있으면 모두 실행'하는 형태입니다.

2. 시스템 상태 파악하기

시스템에서 발생하는 인텐트는 여러 종류이며 부팅 완료, 화면 켬/끔, 배터리 상태 등이 대표적이다.

- 부팅 완료

부팅 완료 후 무엇을 실행시키고 싶으면 브로드캐스트 리시버를 만들고 매니페스트 파일에 인텐트 필터에 구성해서 등록한다.

부팅이 완료되면 시스템에서는 android.intent.action.BOOT_COMPLETED라는 액션 문자열을 포함하는 인텐트가 발생합니다. 이때 실행할 리시버에는 <action>의 name 속성에 똑같은 액션 문자열을 똑같이 등록합니다.

• 브로드캐스트 리시버와 인텐트 필터 등록

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

그런데 리시버를 실행하려면 권한이 필요하므로 매니페스트에 다음처럼 퍼미션을 추가해야 합니다.

• 권한 설정

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

- 화면 켬/끔

여기서 매니페스트에 등록해도 안된다. registerReceiver() 함수를 이용해 동적으로 등록해야 한다.

• 화면 켬/끔 리시버

```
receiver = object : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        when (intent?.action) {
            Intent.ACTION_SCREEN_ON -> Log.d("kkang", "screen on")
            Intent.ACTION_SCREEN_OFF -> Log.d("kkang", "screen off")
        }
    }
}
```

이렇게 하면 onReceive() 함수의 매개변수인 Intent 객체에서 자신을 호출한 인텐트의 액션 문자열을 가져와 화면을 켜거나 끄는 상황을 판단할 수 있습니다.

이렇게 정의한 브로드캐스트 리시버 객체는 액션 문자열 정보로 구성한 인텐트 필터와 함께 registerReceiver() 함수로 시스템에 등록합니다.

• 리시버 등록

```
val filter = IntentFilter(Intent.ACTION_SCREEN_ON).apply {
    addAction(Intent.ACTION_SCREEN_OFF)
}
registerReceiver(receiver, filter)
```

registerReceiver() 함수로 등록한 브로드캐스트 리시버는 필요 없는 순간이 되면 unregisterReceiver() 함수로 등록을 해제해 줘야 합니다.

● 배터리 상태

이것도 인텐트 필터 안에 쓰면 암시적 사용이 안됨.

- BATTERY_LOW: 배터리가 낮은 상태로 변경되는 순간
- BATTERY_OKAY: 배터리가 정상 상태로 변경되는 순간
- BATTERY_CHANGED: 충전 상태가 변경되는 순간
- ACTION_POWER_CONNECTED: 전원이 공급되기 시작한 순간
- ACTION_POWER_DISCONNECTED: 전원 공급을 끊은 순간

• 배터리 상태 리시버

```
receiver = object : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        when (intent?.action) {
            Intent.ACTION_BATTERY_OKAY -> Log.d("kkang", "ACTION_BATTERY_OKAY")
            Intent.ACTION_BATTERY_LOW -> Log.d("kkang", "ACTION_BATTERY_LOW")
            Intent.ACTION_BATTERY_CHANGED -> Log.d("kkang", "ACTION_BATTERY_CHANGED")
            Intent.ACTION_POWER_CONNECTED -> Log.d("kkang", "ACTION_POWER_CONNECTED")
            Intent.ACTION_POWER_DISCONNECTED -> Log.d("kkang", "ACTION_POWER_DISCONNECTED")
        }
    }
}
```

• 시스템 인텐트 없이 배터리 상태 파악하기

```
val intentFilter = IntentFilter(Intent.ACTION_BATTERY_CHANGED)
val batteryStatus = registerReceiver(null, intentFilter)
```

배터리 상태를 알려면 액션 문자열을 Intent.ACTION_BATTERY_CHANGED로 지정한 IntentFilter 객체를 만들고, 이 IntentFilter 정보를 이용해서 registerReceiver() 함수로 브로드캐스트 리시버를 등록해야 합니다. 이때 첫 번째 매개변수를 null로 지정합니다. 결국 특정 브로드캐스트 리시버를 등록하는 코드가 아니라 배터리와 관련된 정보를 registerReceiver() 함수의 반환값인 인텐트 객체의 엑스트라 정보로 등록하는 코드입니다.

• 인텐트의 엑스트라를 이용해 배터리 상태 파악하기

```
val status = batteryStatus!!.getIntExtra(BatteryManager.EXTRA_STATUS, -1)
if (status == BatteryManager.BATTERY_STATUS_CHARGING) {
    // 전원이 공급되고 있다면
    val chargePlug = batteryStatus.getIntExtra(BatteryManager.EXTRA_PLUGGED, -1)
    when (chargePlug) {
        BatteryManager.BATTERY_PLUGGED_USB -> Log.d("kkang", "usb charge")
        BatteryManager.BATTERY_PLUGGED_AC -> Log.d("kkang", "ac charge")
    }
```

```
} else {
    // 전원이 공급되고 있지 않다면
    Log.d("kkang", "not charging")
}
```

현재 전원이 공급되는지는 BatteryManager.EXTRA_STATUS로 얻는 엑스트라값으로 알아낼 수 있습니다. 이 값이 BatteryManager.BATTERY_STATUS_CHARGING이면 전원이 공급되는 상태입니다. 또한 전원이 공급된다면 BatteryManager.EXTRA_PLUGGED로 엑스트라값을 얻을 수 있으며, 이때 엑스트라값이 BatteryManager.BATTERY_PLUGGED_USB이면 저속 충전 상태를, BatteryManager.BATTERY_PLUGGED_AC이면 고속 충전 상태를 의미합니다.

앞에서는 현재 사용자 기기에 전원이 공급되는지를 알아봤는데 때로는 배터리가 얼마나 충전되었는지 알고 싶은 경우도 있습니다. 이때는 다음처럼 작성합니다.

• 배터리 충전량을 퍼센트로 출력

```
val level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1)
val scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1)
val batteryPct = level / scale.toFloat() * 100
Log.d("kkang", "batteryPct : $batteryPct")
```

3. 배터리 정보 앱 만들기

- 정리

1. 브로드캐스트 리시버는 이벤트 모델로 실행되는 컴포넌트이다.
2. 리시버는 BroadcastReceiver를 상속받아 onReceiver() 함수를 재정의해서 작성한다.
3. 리시버를 실행하려면 sendBroadcast() 함수를 이용해 인텐트를 시스템에 전달해야 한다.
4. 시스템 상태 변화를 감지할 수 있다.