

세상의 속도를
따라잡고 싶다면

**Do
it!**

깡샘의
안드로이드
앱 프로그래밍
with **코틀린**

이지스퍼블리싱(주)

14

브로드캐스트 리시버 컴포넌트

- 14-1 브로드캐스트 리시버 이해하기
- 14-2 시스템 상태 파악하기
- 14-3 배터리 정보 앱 만들기

14-1 브로드캐스트 리시버 이해하기

- 이벤트 모델로 실행되는 컴포넌트

브로드캐스트 리시버 만들기

- BroadcastReceiver를 상속받는 클래스를 선언
- 생명주기 함수는 onReceive() 하나뿐
- 브로드캐스트 리시버도 컴포넌트이므로 매니페스트 파일에 등록

• 브로드캐스트 리시버 만들기

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

• 브로드캐스트 리시버 등록

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

14-1 브로드캐스트 리시버 이해하기

동적 등록과 해제

- 필요한 순간에 동적으로 등록
- registerReceiver()라는 함수를 이용해 시스템에 등록
- 필요 없으면 해제해 줘야 합니다. 이때 unregisterReceiver() 함수를 이용

• 리시버 객체 생성

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

• 동적 등록

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

• 브로드캐스트 리시버 해제 함수

```
unregisterReceiver(receiver)
```

14-1 브로드캐스트 리시버 이해하기

브로드캐스트 리시버 실행하기

- 브로드캐스트 리시버를 실행하려면 인텐트가 필요
- 리시버를 매니페스트 파일에 등록하고 <intent-filter> 태그를 선언했다면 암시적 인텐트로는 실행할 수 없습니다.
- 코드에서 registerReceiver() 함수로 등록한 리시버는 암시적 인텐트로도 잘 실행됩니다.
- 리시버를 실행하는 인텐트는 sendBroadcast() 함수로 시스템에 전달

• 리시버 실행

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

14-1 브로드캐스트 리시버 이해하기

구분	실행 대상 액티비티 수	동작
액티비티 인텐트	없음	오류 발생
	1개	정상 실행
	여러 개	사용자 선택으로 1개 실행
리시버 인텐트	없음	오류 발생하지 않음
	1개	정상 실행
	여러 개	모두 실행

14-2 시스템 상태 파악하기

부팅 완료

- 부팅이 완료되면 시스템에서는 android.intent.action.BOOT_COMPLETED라는 액션 문자열을 포함하는 인텐트가 발생

• 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>
```

• 권한 설정

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

14-2 시스템 상태 파악하기

화면 켜/끄

- 화면을 켜거나 끄는 상황을 감지하는 브로드캐스트 리시버는 매니페스트에 등록하면 실행되지 않습니다.
- android.intent.action.SCREEN_ON과 android.intent.action.SCREEN_OFF 액션 문자열

• 화면 켜/끄 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")  
        }  
    }  
}
```

• 리시버 등록

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


14-2 시스템 상태 파악하기

배터리 상태

- 안드로이드 시스템에서 배터리 상태가 변경되면 다음 액션 문자열로 인텐트가 발생
 - 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

14-2 시스템 상태 파악하기

• 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")  
        }  
    }  
}
```

14-2 시스템 상태 파악하기

- 배터리 정보

- 시스템 인텐트 없이 배터리 상태 파악하기

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

- 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")
}
```

14-2 시스템 상태 파악하기

- 배터리가 얼마나 충전되었는지 알고 싶은 경우

• 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")
```

배터리 정보 앱 만들기

1단계. 모듈 생성과 빌드 그래들 설정하기

- Ch14_Receiver라는 이름으로 새로운 모듈
- 그래들 파일에 뷰 바인딩을 사용하도록 설정

2단계. 브로드캐스트 리시버 만들기

- [New → Other → Broadcast Receiver]
- 클래스 이름에 MyReceiver라고 입력

3단계. 리소스 & 소스 파일 복사하기

- res 디렉터리 아래의 drawable, layout 디렉터리를 현재 모듈의 res 디렉터리로 복사
- 소스가 든 디렉터리에서 MainActivity.kt, MyReceiver.kt 파일을 현재 모듈의 소스 영역에 복사

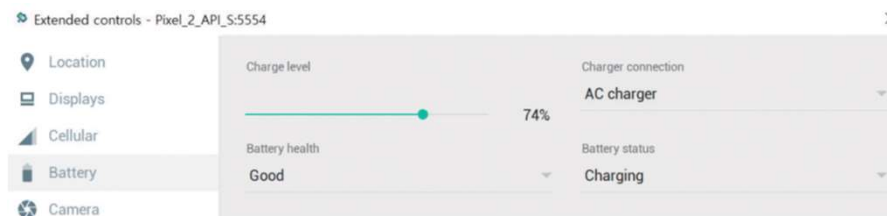
시계 앱의 스톱워치 기능 만들기

4단계. 메인 액티비티 작성하기

- MainActivity.kt 파일을 열고 내용을 추가합니다

5단계. 에뮬레이터에서 배터리 상태 변경하기

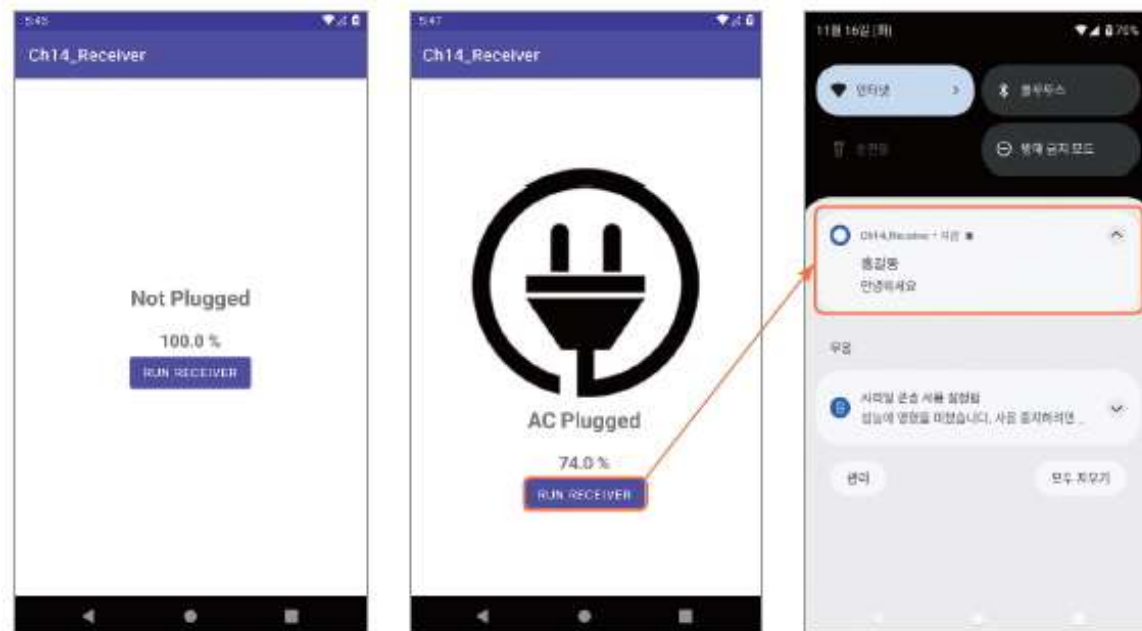
- 에뮬레이터 오른쪽에 있는 확장 메뉴에서 More 아이콘을 클릭
- 왼쪽에서 [Battery]를 선택
- 충전량, 충전기, 배터리 상태 등을 조절



Do it! 실습

시계 앱의 스톱워치 기능 만들기

6단계. 앱 실행하기





감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.



윌리엄 셰익스피어
William Shakespeare