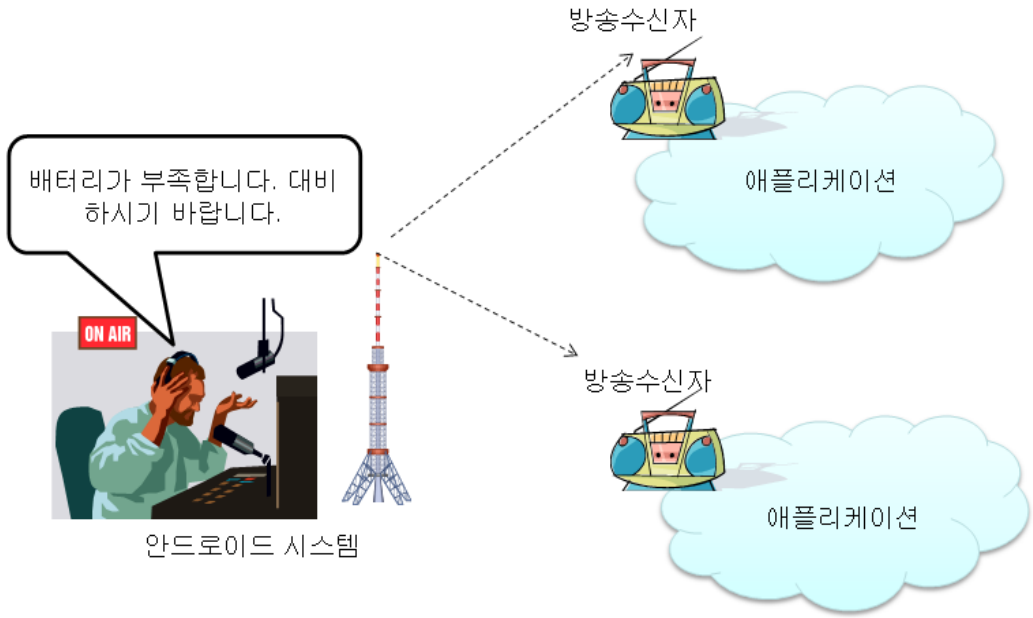
브로드캐스트 수신자 (BroadcastReceiver)

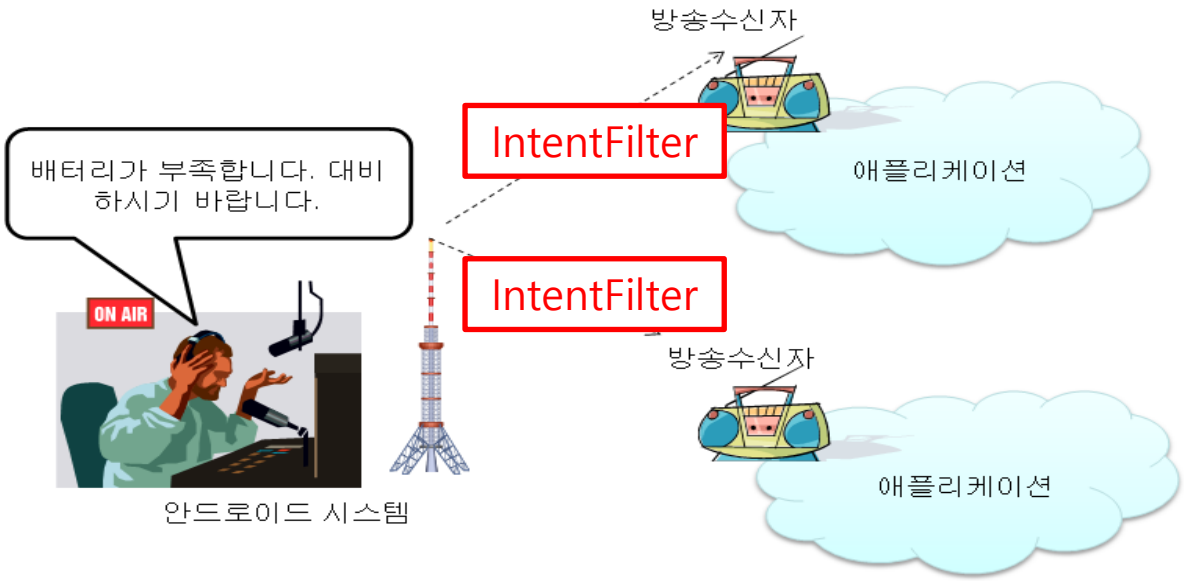
브로드캐스트 수신자

- 안드로이드 장치에서 발생하는 많은 브로드캐스트들을 수신하는 컴포넌트
  - https://developer.android.com/guide/components/broadcasts



브로드캐스트 수신자

- 브로드캐스트 메시지 송수신
  - 안드로이드 시스템이나 다른 앱에서의 메시지
- 앱에서 특정 브로드캐스트를 수신
  - 인텐트 필터에 특정 브로드캐스트를 등록
  - 해당 브로드캐스트가 발생하면 시스템은 등록된 앱들에게 전달



브로드캐스트 수신자

- Context-Registered Receivers (Foreground 수신)
  - 수신자를 컨텍스트에 등록해서 사용
  - 등록 컨텍스트가 유효한 동안 브로드캐스트를 수신

- Manifest-declared Receivers (Background 수신)
  - API레벨 26 이상부터 묵시적 인텐트를 수신할 수 없음
  - 일부 제한된 브로드캐스트만 수신할 수 있음
  - 앱이 실행 중이 아니라면 알림을 통해 앱을 실행

브로드캐스트 수신자

- 브로드캐스트 수신 클래스 선언

```
class MyBroadcastReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {

        // 브로드캐스트가 수신되면 할일
        if(intent.action.equals( 액션 )){

        }
    }
}
```

브로드캐스트 수신자

- 컨텍스트에 등록하여 수신하기

```
val br: BroadcastReceiver = MyBroadcastReceiver()

val filter = IntentFilter()
filter.addAction(Intent.ACTION_POWER_CONNECTED)
context.registerReceiver(br, filter)
```

- 해제하기 context.unregisterReceiver(br)
- 수신자의 등록과 해제 위치
  - onCreate / onDestroy , onResume / onPause
  - Composable 함수 : DisposableEffect
- 브로드캐스트 중단
  - abortBroadcast()

예제. 컨텍스트에 등록하여 수신하기

```
@Composable
fun SystemBroadcastReceiver(
    systemAction: String,
    onSystemEvent: (intent: Intent?) -> Unit
) {
    // Grab the current context in this part of the UI tree
    val context = LocalContext.current

    // Safely use the latest onSystemEvent lambda passed to the function
    val currentOnSystemEvent by rememberUpdatedState(onSystemEvent)

    // If either context or systemAction changes, unregister and register again
    DisposableEffect(context, systemAction) {
        val intentFilter = IntentFilter(systemAction)
        val broadcast = object : BroadcastReceiver() {
            override fun onReceive(context: Context?, intent: Intent?) {
                currentOnSystemEvent(intent)
            }
        }
        context.registerReceiver(broadcast, intentFilter)

        // When the effect leaves the Composition, remove the callback
        onDispose {
            context.unregisterReceiver(broadcast)
        }
    }
}
```

외부에서 전달된 값이
변경될때만 다시 랜더링
(최신상태 유지)

브로드캐스트 수신자

- Background 수신(Manifest-declared Receivers)
  - 사용이 제한되어 있지만, 예외가 존재함

    https://developer.android.com/develop/background-work/background-tasks/broadcasts/broadcast-exceptions
  - 안드로이드 manifest 파일에 등록하여 사용
    - 시스템에서 자동으로 호출 해 줌

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

-. enabled : 시스템에서 브로드캐스트 수신자를 인스턴스화 할 수 있는지 여부
-. exported : 애플리케이션 외부 소스로부터 메시지 수신할 수 있는지 여부

브로드캐스트 수신자

```
class MyBroadcastReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {

        // 브로드캐스트가 수신되면 할일
        if(intent.action.equals( 액션 )){

        }
    }
}
```

- Manifest에 선언된 수신자는 우선 순위가 낮음
  - onReceive 함수에서 수행해야 되는 일이 많다면, goAsync()를 사용해야 함
    - pendingResult가 finish()되기 전까지 Receiver가 남아 있게 됨
      - » 실행시간이 긴 Thread를 동작시키면 안됨

```
val pendingResult = goAsync()
//
pendingResult.finish()
```

ANR(Application Not Responding)

*포그라운드
13이전 : 10초 / 14이후 : 10~20초
* 백그라운드
13이전 : 60초 / 14이후 : 60~120초

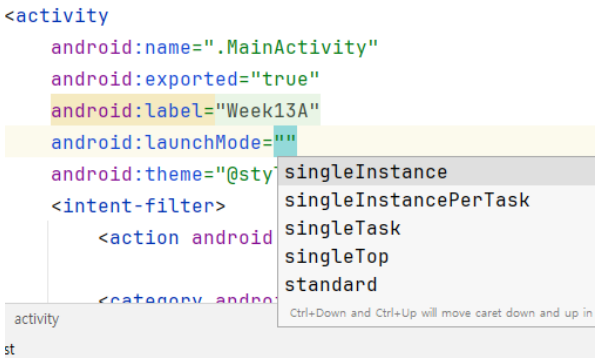
브로드캐스트 수신자

- Background 수신(Manifest-declared Receivers)
  - Background에서 앱 실행 금지
    - Background에서 앱을 직접 실행하는 것은 금지
    - 알림을 통해 앱을 실행시켜야 함
      - 권한 요청 및 허용

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

Activity의 Launch Mode

- Activity의 launch mode
  - Activity의 실행을 제어 및 관리하기 위한 모드
  - https://developer.android.com/guide/components/activities/tasks-and-back-stack
- Standard
  - 디폴트 모드
  - 매번 새로운 액티비티 인스턴스를 생성
    - Activity stack : A → B → C → D
    - Activity stack : A → B → C → D → B (B가 실행)



- SingleTop
  - 현재 태스크의 Top에 액티비티 인스턴스가 존재하면, 새로 생성하지 않음
    - onNewIntent()를 통해 intent 정보를 전달
    - Activity stack : A → B → C → D
    - Activity stack : A → B → C → D → C (C가 실행)
    - Activity stack : A → B → C → D → C (C가 실행, 새로 생성하지 않음)

Activity Launch Mode

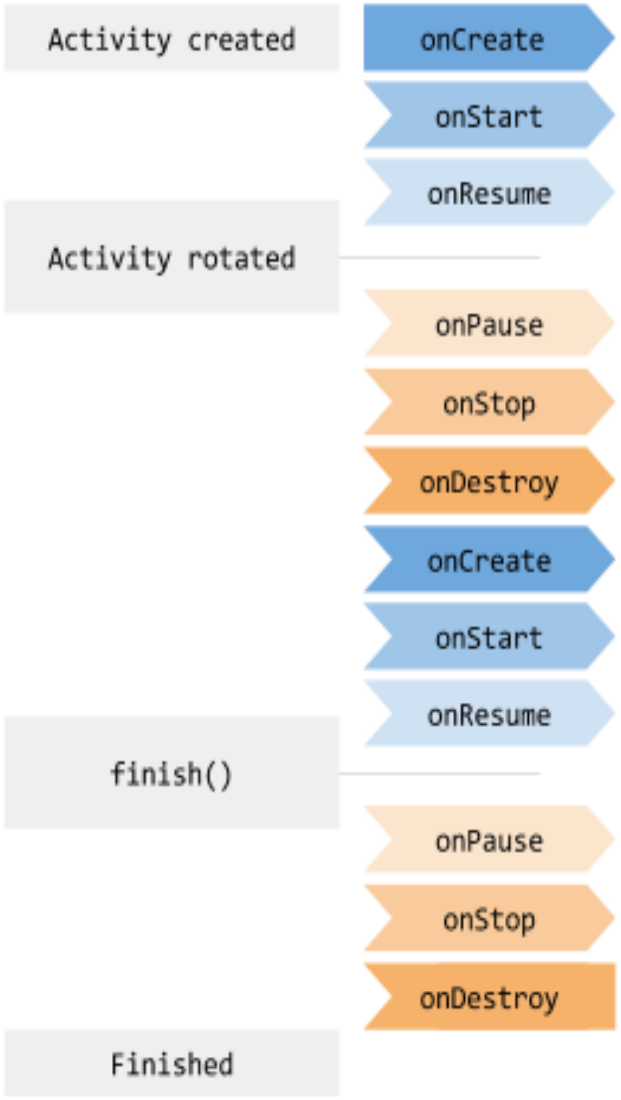
- SingleTask
  - 시스템에 단지 하나의 인스턴스만 생성
  - 인스턴스가 존재하지 않으면 새로 생성하고, 이미 존재하면 onNewIntent() 호출
    - Activity stack : A → B → C
    - Activity stack : A → B → C → D (D실행)
    - Activity stack : A → B (B실행, C와 D는 제거됨)
- SingleInstance
  - SingleTask와 유사하지만, 다른 액티비티들과 같은 Task에 생성될 수 없음
  - Task1 : A → B → C
  - Task2 : D
  - E 와 D 실행
  - Task1 : A → B → C → E
  - Task2 : D
    - onNewIntent() 호출

- SingleInstancePerTask
  - 다른 태스크에 인스턴스가 존재할 수 있음

Intent Flag

- FALG_ACTIVTY_NEW_TASK (“singleTask”)
  - 새로운 태스크를 생성하고, 액티비티를 인스턴스화 함
  - 다른 태스크에 액티비티 인스턴스가 있다면, 새로 생성하지 않음
    - onNewIntent() ➔ onResume() → running
- FLAG_ACTIVITY_SINGLE_TOP (“singleTop”)
  - 액티비티 의 인스턴스가 이미 현재 태스크의 Top에 있으면, 새로 생성하지 않고, onNewIntent() 메소드를 호출해서 실행
- FLAG_ACTIVITY_CLEAR_TOP
  - 액티비티가 현재 태스크에서 실행중이라면, 태스크 위에 있는 다른 액티비티들은 모두 제거하고, Top에 위치 시킴
    - onNewIntent()
    - 액티비티의 launch mode가 디폴트이고, FLAG_ACTIVITY_SINGLE_TOP가 셋팅되어 있지 않으면 re-created 됨

Activity의 Lifecycle



- 새로 생성
  - onCreate ➔ onStart ➔ onResume
- 재사용
  - onNewIntent ➔ onStart ➔ onResume

Activity의 Lifecycle

- 현재 Activity가 실행 상태인지 확인하기 1

```
private fun isAppInForeground(context: Context): Boolean {
    val activityManager : ActivityManager =
        context.getSystemService(Context.ACTIVITY_SERVICE) as ActivityManager
    val runningApps : (Mutable)List<ActivityManager.RunningAppProcessInfo!> =
        activityManager.runningAppProcesses ?: return false

    for (appInfo : ActivityManager.RunningAppProcessInfo! in runningApps) {
        if (appInfo.importance == ActivityManager.RunningAppProcessInfo.IMPORTANCE_FOREGROUND &&
            appInfo.processName == context.packageName
        ) {
            return true
        }
    }
    return false
}
```

Activity의 Lifecycle

- 현재 Activity가 실행 상태인지 확인하기 2

```
class MyApplication : Application() {
    var isAppInForeground = false
        private set

    override fun onCreate() {
        super.onCreate()
        registerActivityLifecycleCallbacks(object : ActivityLifecycleCallbacks {
            override fun onActivityCreated(p0: Activity, p1: Bundle?) {
            }

            override fun onActivityStarted(p0: Activity) {
            }

            override fun onActivityResumed(p0: Activity) {
                isAppInForeground = true
            }

            override fun onActivityPaused(p0: Activity) {
                isAppInForeground = false
            }

            override fun onActivityStopped(p0: Activity) {
            }

            override fun onActivitySaveInstanceState(p0: Activity, p1: Bundle) {
            }

            override fun onActivityDestroyed(p0: Activity) {
            }
        })
    }
}
```

```
<application
    android:name=".MyApplication"
```

* Foreground 확인

```
val application = context?.applicationContext as
                                        MyApplication
if (application.isAppInForeground) {

}
```

예제. SMS 메시지 수신

- 문자 메시지 오면 수신하는 BR 생성
  - 문자 수신 권한 요청 및 허용

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

<uses-feature
    android:name="android.hardware.telephony"
    android:required="false" />
```

정규 표현식 (Regular Expression)

- Regex 클래스
  - https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/-regex/
  - 객체 생성
    - `val regex = Regex("\\d{3}-\\d{4}-\\d{4}")`
    - `val regex = "\\d{3}-\\d{4}-\\d{4}".toRegex()`
    - `val regex = Regex.fromLiteral("\\d{3}-\\d{4}-\\d{4}")`
  - 찾기
    - `regex.containsMatchIn(text)`
    - regex.matchEntire(text)
    - `regex.find(text, idx)`
    - `regex.findAll(text, idx)`

정규표현식

- 정규식 패턴
  - https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

Regex	Meaning
.	Matches any single character.
?	Matches the preceding element once or not at all.
+	Matches the preceding element once or more times.
*	Matches the preceding element zero or more times.
^	Matches the starting position within the string.
$	Matches the ending position within the string.
|	Alternation operator.
[abc]	Matches a or b, or c.
[a-c]	Range; matches a or b, or c.
[^abc]	Negation, matches everything except a, or b, or c.
\s	Matches white space character.
\w	Matches a word character; equivalent to [a-zA-Z_0-9]

수고하셨습니다.