05

05-2. 백그라운드 제약과 JobScheduler

서비스

15-3 백그라운드 제약

리시버의 백그라운드 제약

- 브로드캐스 트 리시버를 암시적으로 실행시키는 것에 제약
- 인텐트 필터 정보에 커스텀 액션 (사용자 임의 지정 문자열)을 이용한 경우 암시적 인텐트로 실행되지 않을 수 있습니다.
- 커스텀 액션으로 선언된 리시버를 인텐트로 실행시키기 위해서는 패키지 정보가 명시되어야 합니다.
- 패키지 정보를 명시하기 위해서는 매니페스트 파일에 패키지 공개 정보를 등록해야 합니다.

• 매니페스트에 브로드캐스트 리시버 등록

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

• 패키지 정보를 명시해 리시버 실행

```
val intent = Intent("ACTION_RECEIVER")
intent.setPackage("com.example.test15_outer")
sendBroadcast(intent)
```

• 패키지 공개 정보 등록

```
<queries>
    <package android:name="com.example.test15_outer"/>
</queries>
```

15-3 백그라운드 제약

리시버의 백그라운드 제약

- registerReceiver() 함수로 등록하면 패키지명을 주지 않아도 정상적으로 실행
- registerReceiver() 함수로 리시버를 등록하려면 꼭 RECEIVER_EXPORTED 정보가 설정되어 있어야 외부에서 접근할 수 있습니다.

• 코드에서 브로드캐스트 리시버 등록

```
registerReceiver(receiver, IntentFilter("ACTION_OUTER_RECEIVER"), RECEIVER_EXPORTED)
```

• 암시적 인텐트로 브로드캐스트 리시버 실행

```
val intent = Intent("ACTION_OUTER_RECEIVER")
sendBroadcast(intent)
```

15-3 백그라운드 제약

서비스의 백그라운드 제약

- 서비스는 앱이 백그라운드 상태일 때 인텐트를 전달하면 오류가 발생

```
Not allowed to start service Intent { act=ACTION_OUTER_SERVICE pkg=com.example.test_
outter }: app is in background uid null
```

- 포그라운드 상황
  - 액티비티가 시작되든 일시 중지되든 상관없이 보이는 액티비티가 있을 때
  - 포그라운드 서비스가 있을 때
  - 앱의 서비스에 바인딩하거나 앱의 콘텐츠 프로바이더를 사용해 또 다른 포그라운드 앱이 연결되었을 때

15-3 백그라운드 제약

- 앱이 백그라운드 상황이더라도 다음과 같은 경우에는 서비스가 정상으로 실행
  - 우선순위가 높은 파이어베이스 클라우드 메시징(FCM) 처리
  - SMS/MMS 메시지와 같은 브로드캐스트 수신
  - 알림에서 PendingIntent 실행
  - VPN 앱이 포그라운드로 승격되기 전에 VpnService 시작

- startForegroundService() 함수로 인텐트를 시작하면 앱이 백그라운드 상황에서도 서비스가 실행

• 안드로이드 버전에 따라 백그라운드 상황 대처

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    startForegroundService(intent)
} else {
    startService(intent)
}
```

15-3 백그라운드 제약

- 앱이 백그라운드 상황에서 startForegroundService() 함수로 실행한 서비스는 얼마 후 다음과 같은 오류가 발생하면서 강제로 종료

```
Context.startForegroundService() did not then call Service.startForeground()
```

- startForegroundService() 함수로 실행했다면 빨리 startForeground() 함수를 호출해 포그라운드 상황으로 만들라는 의미

• 서비스 쪽 코드

```
val notification = builder.build()
startForeground(1, notification)
```

15-3 백그라운드 제약

- 포그라운드 서비스는 <service> 태그에 foregroundServiceType을 지정해 주어야 합니다.
- foregroundServiceType의 값은 camera, connectedDevice, dataSync, remoteMessaging, specialUse 등이 있으며 이 중 하나를 지정해 포그라운드에서 어떤 작업을 하는 서비스인지를 등록해야 합니다.
- foregroundServiceType의 값에 맞는 퍼미션도 설정해야 합니다.
- 예처럼 foregroundServiceType="specialUse"라고 설정했다면 android.permission.FOREGROUND_SERVICE_SPECIAL_USE라는 이름의 퍼미션이 등록되어 있어야 합니다.

• 퍼미션 등록

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

15-4 잡 스케줄러

잡 스케줄러의 실행조건

- 잡 스케줄러에 조건으로 명시할 수 있는 상황
  - 네트워크 타입
  - 배터리 충전 상태
  - 특정 앱의 콘텐츠 프로바이더 갱신(대표적으로 갤러리 앱)
  - 실행 주기
  - 최소 지연 시간
  - 시스템 재구동 시 현재 조건 유지 여부

15-4 잡 스케줄러

잡 스케줄러의 3가지 구성 요소

- 잡 스케줄러는 다음 3가지 요소로 구성
  - 잡 서비스: 백그라운드에서 처리할 작업을 구현한 서비스입니다.
  - 잡 인포: 잡 서비스 정보와 실행될 조건을 지정합니다.
  - 잡 스케줄러: 잡 인포를 시스템에 등록합니다.

15-4 잡 스케줄러

- 잡 서비스 — 백그라운드 작업 구현
  - 잡 서비스는 개발자가 만드는 서비스이므로 매니페스트에 <service> 태그로 등록
  - android.permission.BIND_JOB_SER VICE 퍼미션도 포함

• 매니페스트에 잡 서비스 등록

```
<service
    android:name=".MyJobService"
    android:enabled="true"
    android:exported="true"
    android:permission="android.permission.BIND_JOB_SERVICE"></service>
```

15-4 잡 스케줄러

- JobService를 상속받아 작성
- onStartJob(), onStopJob() 함수는 반드시 재정의
- onStartJob() 함수에는 백그라운드에서 처리할 작업을 구현
- 함수의 반환값은 Boolean 타입인데 true인지 false인지에 따라서 다르게 동작
  - false: 작업이 완벽하게 종료되었음을 의미합니다.
  - true: 작업이 아직 끝나지 않았음을 의미합니다.

• 잡 서비스 클래스 작성

```
@TargetApi(Build.VERSION_CODES.LOLLIPOP)
class MyJobService : JobService() {
    override fun onCreate() {
        super.onCreate()
        Log.d("kkang","MyJobService.......onCreate()")
    }
    override fun onDestroy() {
        super.onDestroy()
        Log.d("kkang","MyJobService.......onDestroy()")
    }
    override fun onStartJob(params: JobParameters?): Boolean {
        Log.d("kkang","MyJobService.......onStartJob()")
        return false
    }
    override fun onStopJob(params: JobParameters?): Boolean {
        Log.d("kkang","MyJobService.......onStopJob()")
        return false
    }
}
```

15-4 잡 스케줄러

- onStopJob() 함수의 반환값도 Boolean 타입인데 true, false에 따라 다르게 동작
  - false: 잡 스케줄러 등록을 취소합니다.
  - true: 잡 스케줄러를 재등록합니다.

- 잡 인포 — 잡 서비스의 실행 조건 정의
  - JobInfo.Builder에 지정한 잡 서비스가 실행되는 조건을 명시
    - setPersisted(true): 기기를 재부팅해도 작업 등록을 유지해야 하는지를 설정합니다.
    - setPeriodic(long intervalMillis): 작업의 실행 주기를 설정합니다.
    - setMinimumLatency(long minLatencyMillis): 작업의 실행 지연 시간을 설정합니다.
    - setOverrideDeadline(long maxExecutionDelayMillis): 다른 조건에 만족하지 않더라도 작업이 이 시간 안에 실행되어야 함을 설정합니다.
    - setRequiredNetworkType(int networkType): 네트워크 타입을 설정합니다.
    - setRequiresBatteryNotLow(boolean batteryNotLow): 배터리가 낮은 상태가 아님을 설정합니다.
    - setRequiresCharging(boolean requiresCharging): 배터리가 충전 상태인지를 설정합니다.

• 잡 서비스의 실행 조건 정의

```
var jobScheduler: JobScheduler? = getSystemService<JobScheduler>()

JobInfo.Builder(1, ComponentName(this, MyJobService::class.java)).run {
    setRequiredNetworkType(JobInfo.NETWORK_TYPE_UNMETERED)
    jobScheduler?.schedule(build())
}
```

15-4 잡 스케줄러

- 잡 스케줄러 — 잡 서비스 등록 시 데이터 전달
  - 잡 서비스를 JobInfo 객체를 이용해 시스템에 등록하면 조건에 만족할 때 실행
  - 잡 서비스에 데이터를 전달하려면 JobInfo.Builder의 setExtras() 함수를 이용

• 잡 서비스에 데이터 전달

```
var jobScheduler: JobScheduler? = getSystemService<JobScheduler>()
val extras = PersistableBundle()
extras.putString("extra_data", "hello kkang")
val builder = JobInfo.Builder(1, componentName)
builder.setRequiredNetworkType(JobInfo.NETWORK_TYPE_UNMETERED)
builder.setRequiresCharging(true)
builder.setExtras(extras)
val jobInfo = builder.build()
jobScheduler!!.schedule(jobInfo)
```

15-4 잡 스케줄러

- 전달한 데이터를 잡 서비스에서 가져올 때는 onStartJob() 함수의 매개변수를 이용() 함수를 이용

• 잡 서비스에서 데이터 가져오기

```
override fun onStartJob(jobParameters: JobParameters): Boolean {
    jobParameters.extras.getString("extra_data")
    (... 생략 ...)
    return false
}
```

감사합니다

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

윌리엄 셰익스피어
William Shakespeare