Runtime Permission Requests

Mobile Software
2019 Fall

권한 요청

- 애플리케이션에서 internet 연결과 같은 특별한 작업을 수행하기 위한 사용 권한(permission)을 요청할 때
  - AndroidManifest.xml 파일에 필요한 작업 표시
  - <uses-permission> 태그 사용

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.ourincheon.ch13_project">

    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

권한의 종류

권한	권한 이름	설명
지리 정보 사용	ACCESS_FINE_LOCATION	GPS와 같은 정밀한 위치 정보 사용
전화 걸기	CALL_PHONE	애플리케이션이 전화 걸기 기능 사용
카메라	CAMERA	카메라 사용 가능
일정 정보	READ_CALENDAR	일정 정보 읽기
	WRITE_CALENDAR	일정 정보 쓰기
연락처 정보	READ_CONTACTS	연락처 읽기
	WRITE_CONTACTS	연락처 쓰기
인터넷	INTERNET	인터넷 접속

Marshmallow 권한 관리(1/2)

- 선택적인 권한 부여
  - **Normal permission**
    - App. 설치 시 권한 부여 (~5.0)
      - AndroidManifest.xml에 권한 설정만으로 가능
  - **Dangerous permission**
    - App. 실행 시 권한 획득 (6.0~)
      - targetSdkVersion 23 이상 버전부터 적용
      - User가 시스템 설정에서 취소 가능하기 때문
    - 해당되는 작업 그룹
      - Calendar, Camera, Contacts, Location
      - Microphone, Phone, Sensors, SMS, Storage

Dangerous permission(위험 권한)

Permission Group	Permissions
CALENDAR	READ_CALENDAR, WRITE_CALENDAR
CAMARA	CAMERA
CONTACTS	READ_CONTACTS, WRITE_CONTACTS GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE, CALL_PHONE READ_CALL_LOG, WRITE_CALL_LOG ADD_VOICEMAIL, USE_SIP PROCESS_OUTGOING_CALLS
SENSORS	BODY_SENSORS
SMS	SEND_SMS, RECEIVE_SMS, READ_SMS RECEIVE_WAP_PUSH, RECEIVE_MMS
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE

Marshmallow 권한 관리(2/2)

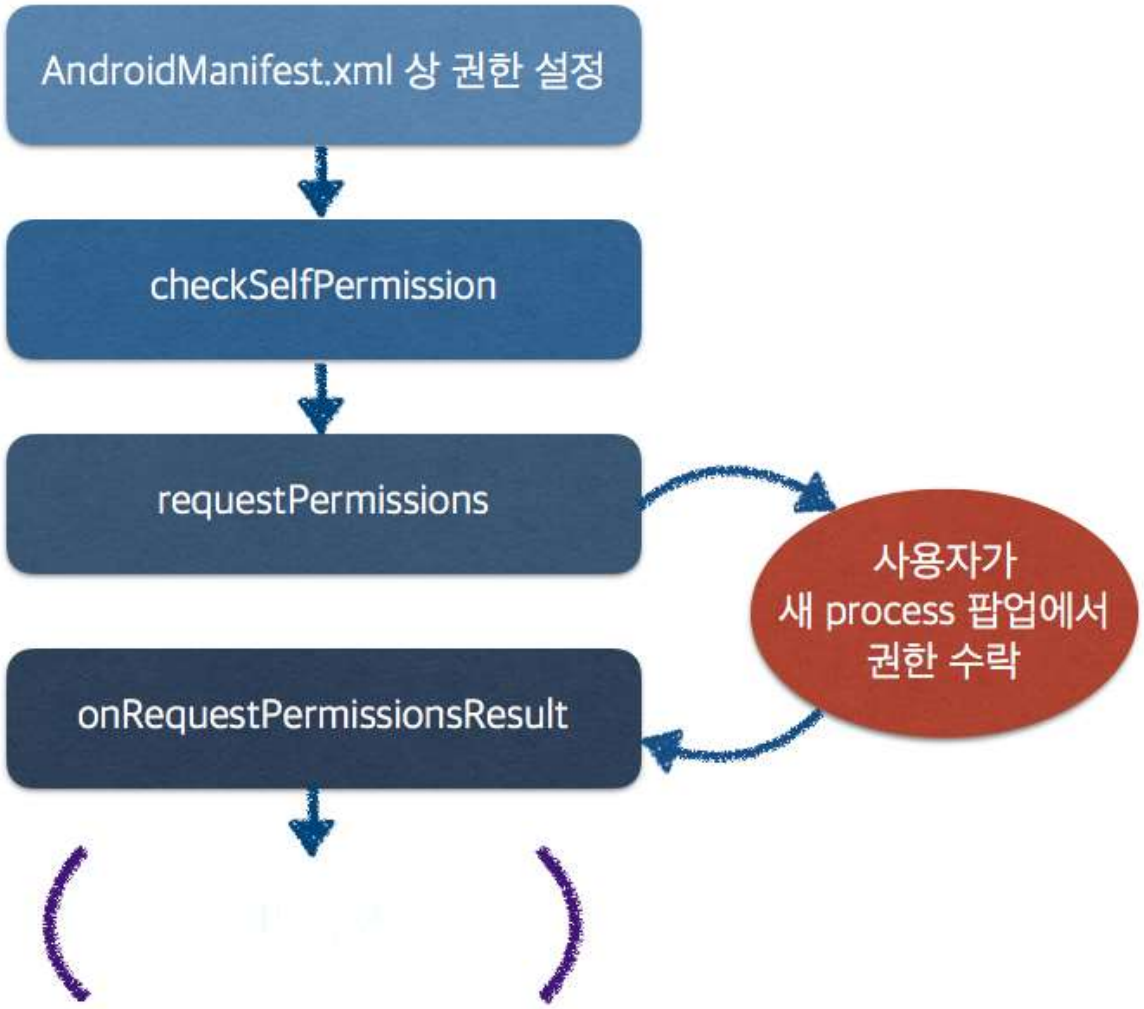


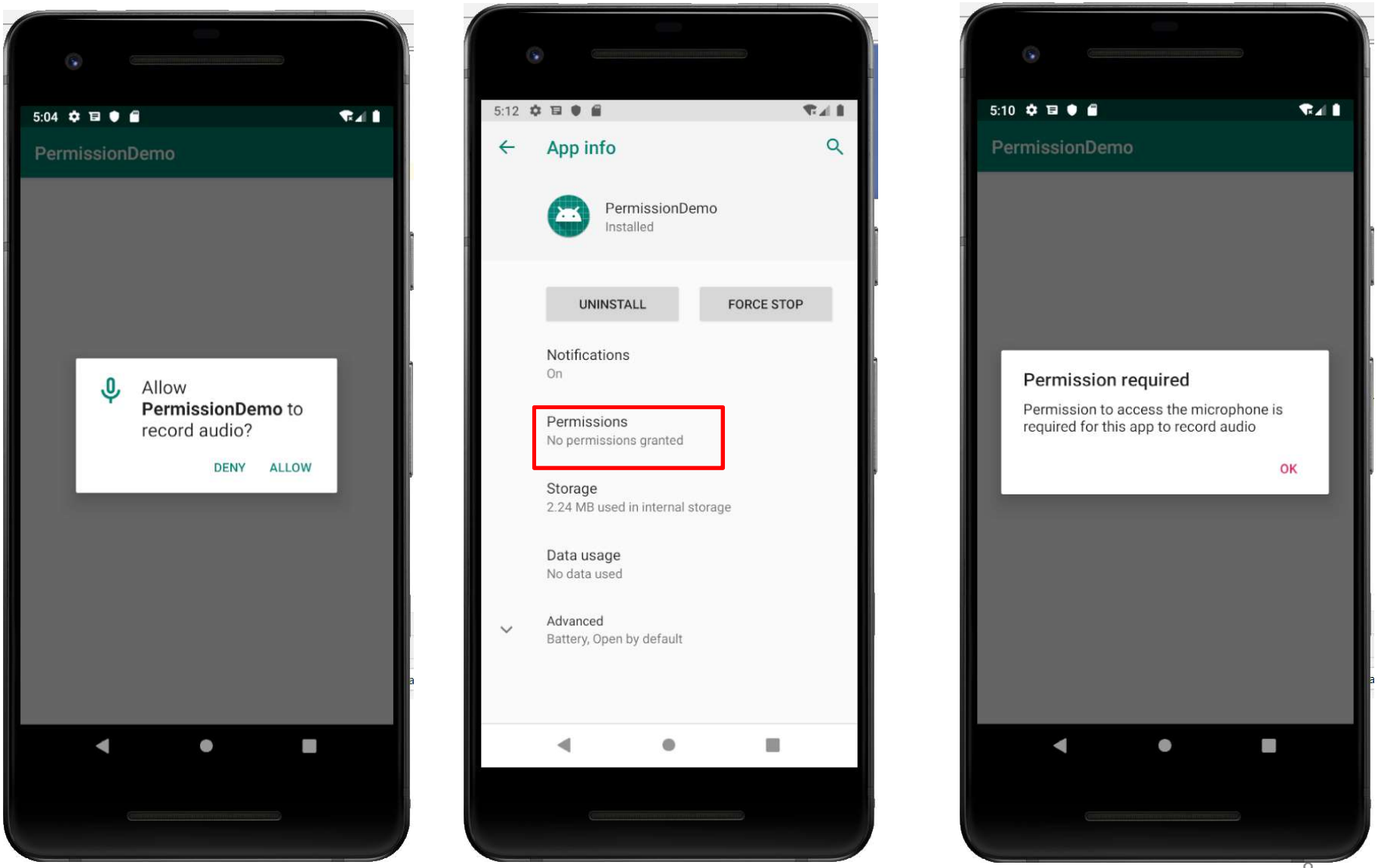
그림 출처: https://realm.io/kr/news/android-marshmellow-permission/

실습 준비

- 새 프로젝트 생성
  - **Activity** : **Empty Activity**
  - **Application name** : **PermissionDemo**
  - **Minimum API level : API 26** (Oreo)
  - **Activity name** : **MainActivity.kt** (자동 생성)
  - **Layout name**
    - **activity_main.xml** (자동 생성)

- 자동 생성된 레이아웃 XML 파일은 1개
  - **activity_main.xml** 의 root layout은 **ConstraintLayout**

Dangerous Permission



컴파일 타임 권한 허용 – Manifest

AndroidManifest.xml

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.ourincheon.permissiondemo">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="PermissionDemo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

(a) 위험 권한 허용 확인

MainActivity.kt

```
import android.Manifest
import android.content.pm.PackageManager
import android.os.Bundle
import android.util.Log
import androidx.appcompat.app.AppCompatActivity
import androidx.core.content.ContextCompat

class MainActivity : AppCompatActivity() {

    private val TAG = "PermissionDemo"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        setupPermission()
    }

    private fun setupPermission() {
        val permission = ContextCompat.checkSelfPermission(this,
            Manifest.permission.RECORD_AUDIO)

        if (permission != PackageManager.PERMISSION_GRANTED) {
            Log.i(TAG, "Permission to record denied")
        }
    }
}
```

Manifest에서 권한을 설정했음에도 승인 거절 메시지 출력 (**6.0 이상**)

```
2019-10-18 13:50:45.129 5370-5370/? I/PermissionDemo: Permission to record denied
```

(b) 위험 권한 요청

MainActivity.kt

```
private val RECORD_REQUEST_CODE = 101

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_permission_demo)

    setupPermission()
}

private fun setupPermission() {
    val permission = ContextCompat.checkSelfPermission(this,
        Manifest.permission.RECORD_AUDIO)

    if (permission != PackageManager.PERMISSION_GRANTED) {
        // Log.i(TAG, "Permission to record denied")
        makeRequest()
    }
}

private fun makeRequest() {
    ActivityCompat.requestPermissions(this,
        arrayOf(Manifest.permission.RECORD_AUDIO),
        RECORD_REQUEST_CODE)
}
```

앱이 처음 실행되었을 때 위험 권한 요청

(c) 위험 권한 승인 여부 확인

```
override fun onRequestPermissionsResult(requestCode: Int,
                permissions: Array<out String>, grantResults: IntArray) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    when (requestCode) {
        RECORD_REQUEST_CODE -> {
            if (grantResults.isEmpty() || grantResults[0] !=
                        PackageManager.PERMISSION_GRANTED) {
                Log.i(TAG, "Permission has been denied by user")
            } else {
                Log.i(TAG, "Permission has been granted by user")
            }
        }
    }
}
```

```
2019-10-18 14:06:02.212 5667-5667/edu.ourincheon.permissiondemo I/PermissionDemo: Permission has been denied by user
```

(d) 위험 권한 요청 이유 설명

```
private fun setupPermission() {
    val permission = ContextCompat.checkSelfPermission(this,
        Manifest.permission.RECORD_AUDIO)

    if (permission != PackageManager.PERMISSION_GRANTED) {
        // Log.i(TAG, "Permission to record denied")
        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                Manifest.permission.RECORD_AUDIO)) {
            val builder = AlertDialog.Builder(this)
            builder.setMessage("Permission to access the microphone is required" +
                    "for this app to record audio")
                .setTitle("Permission required")
            builder.setPositiveButton("OK"){ dialog, id ->
                Log.i(TAG, "Clicked")
                makeRequest()
            }
            val dialog = builder.create()
            dialog.show()
        } else {
            makeRequest()
        }
    }
}
```

앱 처음 실행했을 때 거절한 후
다시 해당 앱을 실행시키면
위험 권한을 요청하는 이유를
설명하는 다이얼로그 창이 나타남