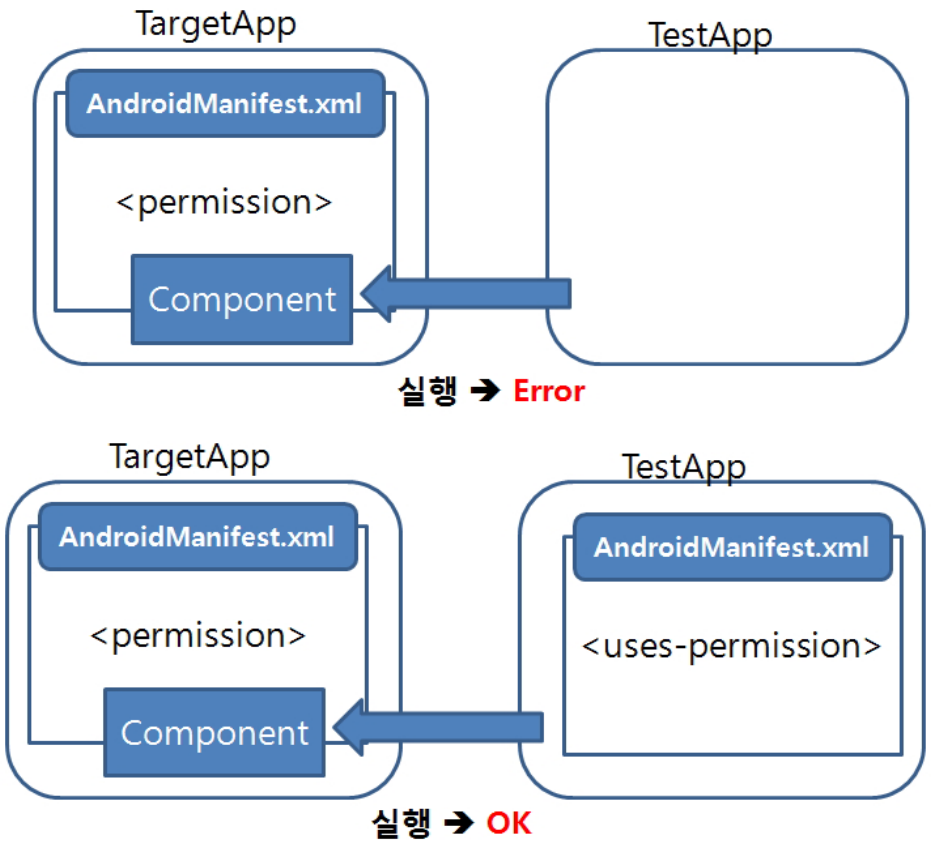
9장. 파일 및 SharedPreferences을 이용한 데이터 영속화

9.1 퍼미션

9.1.1. 퍼미션이란?

- AndroidManifest.xml에 들어가는 설정
- 어떤 앱이 <permission>을 부여했다면 그 앱을 이용하는 앱은 <uses-permission>을 선언

- <permission> 이용



9.1 퍼미션

- AndroidManifest.xml에 <permission> 태그를 추가

```
<permission android:name="com.test.permission.SOME_PERMISSION"
    android:label="SOME Permission"
    android:description="@string/permission"
    android:protectionLevel="normal"/>
```

- name: 퍼미션의 이름
- label, description: 퍼미션에 대한 설명(사용자에게 보이는 문자열)
- protectionLevel: 보호 수준

- protectionLevel을 이용해 보호 수준

- normal: 낮은 수준의 보호. 사용자에게 권한 부여 요청이 필요 없는 경우
- dangerous: 높은 수준의 보호. 사용자에게 권한 부여 요청이 필요한 경우
- signature: 동일한 키로 사인된 앱만 실행
- signatureOrSystem: 안드로이드 시스템 앱이거나 동일 키로 사인된 앱만 실행

9.1 퍼미션

- 컴포넌트에 퍼미션을 적용

```
<activity android:name=".SomeActivity"
    android:permission="com.test.permission.SOME_PERMISSION">
    <intent-filter>
        <action android:name="AAA"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
```

- 보호된 컴포넌트를 이용하는 앱

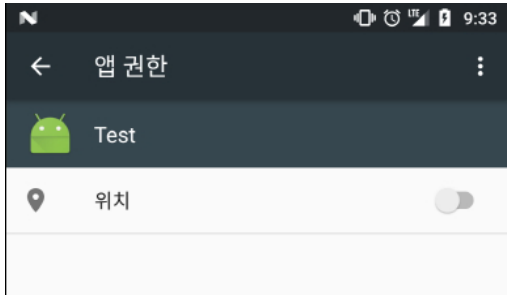
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.test">

    <uses-permission android:name="com.test.permission.SOME_PERMISSION"/>
    <!--중략-->
</manifest>
```

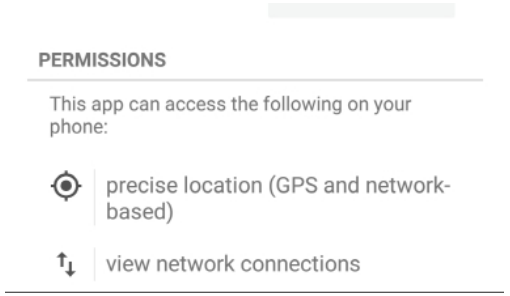
protectionLevel 속성

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

- android.permission.ACCESS_NETWORK_STATE는 “normal”로 선언,
  android.permission.ACCESS_FINE_ LOCATION은 “dangerous”로 선언.

9.1 퍼미션





- 5.1까지는 권한 화면에서 어떤 퍼미션이 사용되는지 정보 성격으로만 알려주지만, 6.0 이상부터는 사용자에게 해당 권 한을 부여할 것인지 선택

9.1 퍼미션

시스템 퍼미션

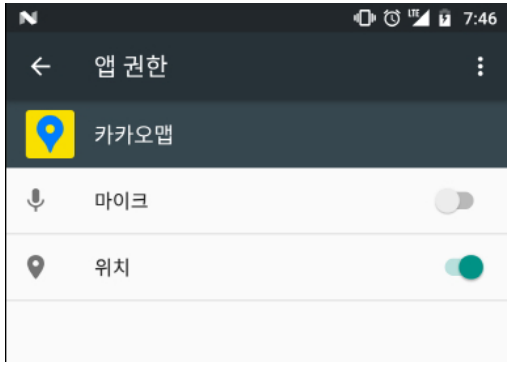
퍼미션 그룹	퍼미션
CALENDAR	READ_CALENDAR
	WRITE_CALENDAR
CAMERA	CAMERA
CONTACTS	READ_CONTACTS
	WRITE_CONTACTS
	GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION
	ACCESS_COARSE_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE
	CALL_PHONE
	READ_CALL_LOG
	WRITE_CALL_LOG
	ADD_VOICEMAIL
	USE_SIP
	PROCESS_OUTGOING_CALLS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

9.1 퍼미션

- ACCESS_FINE_LOCATION: 정확한 위치 정보 액세스
- ACCESS_NETWORK_STATE: 네트워크에 대한 정보 액세스
- ACCESS_WIFI_STATE: 와이파이 네트워크에 대한 정보 액세스
- BATTERY_STATS: 배터리 통계 수집
- BLUETOOTH: 연결된 블루투스 장치에 연결
- BLUETOOTH_ADMIN: 블루투스 장치를 검색하고 페어링
- CALL_PHONE: 다이얼 UI를 거치지 않고 전화를 시작
- CAMERA: 카메라 장치에 액세스
- INTERNET: 네트워크 연결
- READ_CONTACTS: 사용자의 연락처 데이터 읽기
- READ_EXTERNAL_STORAGE: 외부 저장소에서 파일 읽기
- READ_PHONE_STATE: 장치의 전화번호, 네트워크 정보, 진행 중인 통화 상태 등 전화 상태에 대한 읽기
- READ_SMS: SMS 메시지 읽기
- RECEIVE_BOOT_COMPLETED: 부팅 완료 시 수행
- RECEIVE_SMS: SMS 메시지 수신
- RECORD_AUDIO: 오디오 녹음
- SEND_SMS: SMS 메시지 발신
- VIBRATE: 진동 울리기
- WRITE_CONTACTS: 사용자의 연락처 데이터 쓰기
- WRITE_EXTERNAL_STORAGE: 외부 저장소에 파일 쓰기

9.1 퍼미션

9.1.2. 안드로이드 6.0 (API Level 23) 변경사항



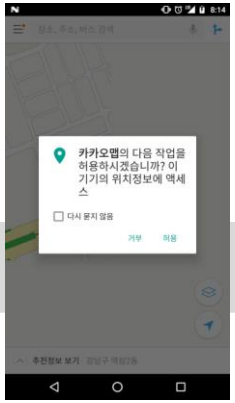
- 6.0 이전까지 퍼미션은 일종의 개발자 신고제
- 6.0부터는 퍼미션을 사용자가 거부 가능

- 퍼미션 상태 를 확인
- int checkSelfPermission (Context context, String permission)
- PERMISSION_GRANTED: 퍼미션이 부여된 상태
- PERMISSION_DENIED: 펴미션이 부여되지 않은 상태

```
if(ContextCompat.checkSelfPermission(this,
    Manifest.permission.READ_EXTERNAL_STORAGE) ==
    PackageManager.PERMISSION_GRANTED){
    //...
}
```

9.1 퍼미션

- 퍼미션 허용을 요청
- void requestPermissions(Activity activity, String[] permissions, int requestCode)




```
ActivityCompat.requestPermissions(this,
new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},200 );
```

- 퍼미션을 허용했는지 판단
- void onRequestPermissionsResult(int requestCode, String[] permissions,
  int[] grantResults)

```
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    if(requestCode==200 && grantResults.length>0) {
        if(grantResults[0]==PackageManager.PERMISSION_GRANTED)
            //.................
        if(grantResults[1]==PackageManager.PERMISSION_GRANTED)
            //.................
    }
}
```

9.2 파일에 읽고 쓰기

파일 관련 프 로그램은 대부분 자바 API를 그대로 사용

- File: 파일 및 디렉터리를 지칭하는 클래스
- FileInputStream: 파일에서 바이트 데이터를 읽기 위한 함수 제공
- FileOutputStream: 파일에 바이트 데이터를 쓰기 위한 함수 제공
- FileReader: 파일에서 문자열 데이터를 읽기 위한 함수 제공
- FileWriter: 파일에 문자열 데이터를 쓰기 위한 함수 제공

Environment

- Environment.getExternalStorageState(): 외부 저장 공간 상태
- Environment.getExternalStorageDirectory().getAbsolutePath(): 외부 저장 공간 경로
- Environment.getDataDirectory().getAbsolutePath(): 내부 저장 공간 경로

9.2 파일에 읽고 쓰기

9.2.1. 외부 저장 공간 이용

- 외부 저장 공간을 제공하는지 판단

```
String state = Environment.getExternalStorageState();
if (state.equals(Environment.MEDIA_MOUNTED)) {
    if (state.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
        externalStorageReadable = true;
        externalStorageWritable = false;
    } else {
        externalStorageReadable = true;
        externalStorageWritable = true;
    }
}else {
    externalStorageReadable = externalStorageWritable = false;
}
```

- 퍼미션

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.test3_9">
    <uses-permission android:name=
                    "android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name=
                    "android.permission.READ_EXTERNAL_STORAGE" />
    <!-- 중략 -->
</manifest>
}
```

9.2 파일에 읽고 쓰기

- 문자열 데이터를 저장

```
FileWriter writer;
try {
    //외          장root 하  에 myApp이
    String dirPath = Environment.getExternalStorageDirectory()
            .getAbsolutePath() + "/myApp";
    File dir = new File(dirPath);
    //폴                    다
    if(!dir.exists()){
        dir.mkdir();
    }
    //myApp 폴     에 myfile.txt 파     일
    File file=new File(dir+"/myfile.txt");
    //파                    다
    if(!file.exists()){
        file.createNewFile();
    }
    //파일 쓰기
    writer = new FileWriter(file, true);
    writer.write(content);
    writer.flush();
    writer.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

9.2 파일에 읽고 쓰기

- createTempFile ( ) 함수이용

```
File tempFile = File.createTempFile("IMG", ".jpg",dir);
```

- 공용 폴더를 사용

```
File file1 = new File(Environment.getExternalStoragePublicDirectory( Environment.DIRECTORY_PICTURES), "a.jpg");
```

- Environment.DIRECTORY_ALARMS: 알람으로 사용할 오디오 파일 저장 폴더
- Environment.DIRECTORY_DCIM: 카메라로 촬영한 사진 저장 폴더
- Environment.DIRECTORY_DOWNLOADS: 다운로드한 파일 저장 폴더
- Environment.DIRECTORY_MUSIC: 음악 파일 저장 폴더
- Environment.DIRECTORY_MOVIES: 영상 파일 저장 폴더
- Environment.DIRECTORY_NOTIFICATIONS: 알림음으로 사용할 오디오 파일 저장 폴더
- Environment.DIRECTORY_PICTURES: 이미지 파일 저장 폴더

9.2 파일에 읽고 쓰기

- 파일을 읽기

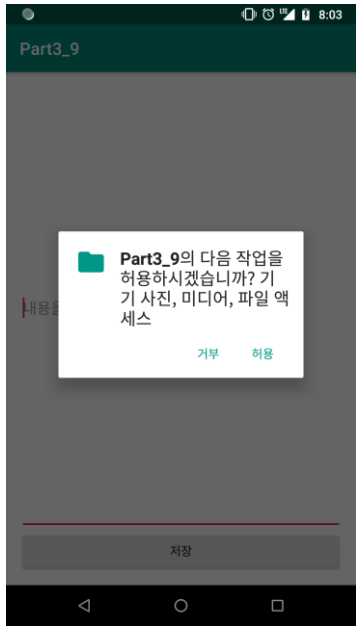
```
file = new File(Environment.getExternalStorageDirectory()
            .getAbsolutePath()+ "/myApp/myfile.txt");
}
try {
    BufferedReader reader= new BufferedReader(new FileReader(file));
    StringBuffer buffer=new StringBuffer();
    String line;
    while ((line=reader.readLine()) != null){
        buffer.append(line);
    }
    reader.close();
}catch (Exception e){
    e.printStackTrace();
}
```

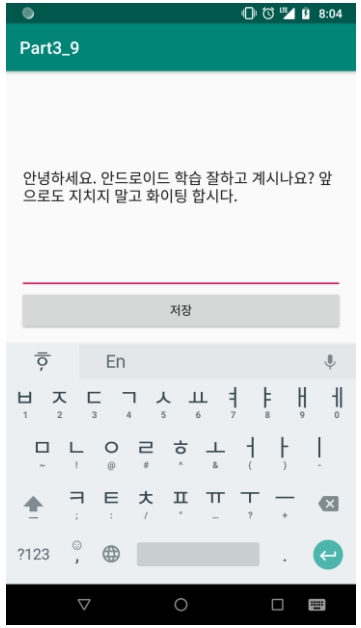
9.2.2. 내부 저장 공간 이용

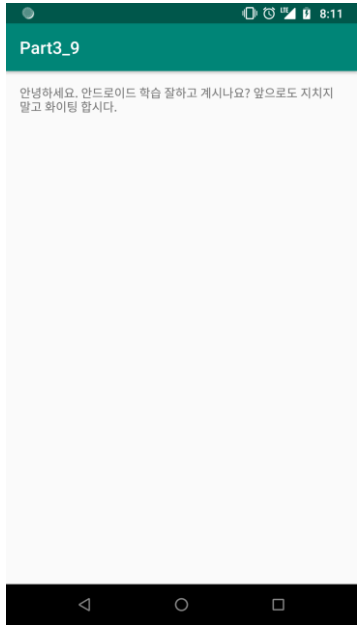
- 앱의 데이터를 내부 저장 공간에 저장하려면 getFileDir ( ) 함 수로 경로 획득

Step by Step 9-1 – 파일 다루기

- 모듈 생성
- 퍼미션 부여
- 결과 확인을 위한 액티비티 생성
- activity_read_file.xml 작성
- ReadFileActivity 작성
- activity_main.xml 복사
- MainActivity 작성







9.3 SharedPreferences

9.3.1. SharedPreferences

- 데이터를 간단하게 키-값(key-value) 성격으로 저장
- 파일(XML)로 저장

SharedPreference 객체를 하나 획 득
- getPreferences(int mode)
- getSharedPreferences(String name, int mode)
- PreferenceManager.getDefaultSharedPreferences(Context context)

```
SharedPreferences sharedPref = getPreferences(Context.MODE_PRIVATE);
```

```
SharedPreferences sharedPref = getSharedPreferences("my_prefs",
Context.MODE_PRIVATE);
```

```
SharedPreferences sharedPref= PreferenceManager.getDefaultSharedPreferences(this);
```

- MODE_PRIVATE: 자기 앱 내에서 사용. 외부 앱에서 접근 불가
- MODE_WORLD_READABLE: 외부 앱에서 읽기 가능
- MODE_WORLD_WRITEABLE: 외부 앱에서 쓰기 가능

9.3 SharedPreferences

데이터를 저장하려면 Editor 클래스의 함수를 이용

- putBoolean(String key, boolean value)
- putFloat(String key, float value)
- putInt(String key, int value)
- putLong(String key, long value)
- putString(String key, String value)

```
SharedPreferences.Editor editor=sharedPref.edit();
editor.putString("data1", "hello");
editor.putInt("data2", 100);
editor.commit();
```

데이터를 획득

- getBoolean(String key, boolean defValue)
- getFloat(String key, float defValue)
- getInt(String key, int defValue)
- getLong(String key, long defValue)
- getString(String key, String defValue)

```
String data1=sharedPref.getString("data1", "none");
int data2=sharedPref.getInt("data2", 0);
```