

---

# Kotlin을 이용한 Android 프로그래밍

## sqlite와 recycleView를 이용한 Todo 리스트 만들기

---

# Contents

---

## I. 에뮬레이터 실행

- I. 일기장 과제 앱을 에뮬레이터에 설치
- II. 앱 디렉터리 경로 및 파일 생성 확인

## II. 실습 예제 진행

- I. 리스트 뷰 사용법 리뷰
- II. 파일 관리 방법

## III. SQLite의 개요

## IV. 내부 DB 생성

- I. 에뮬레이터 내부에 직접 데이터베이스를 생성하고 다루기 - 전용 프로그램 사용
- II. 쿼리 사용

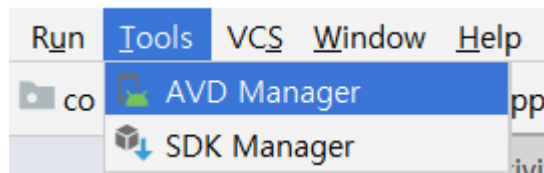
## V. SQLite를 이용한 간단한 앱 만들기

## VI. RecyclerView 개요

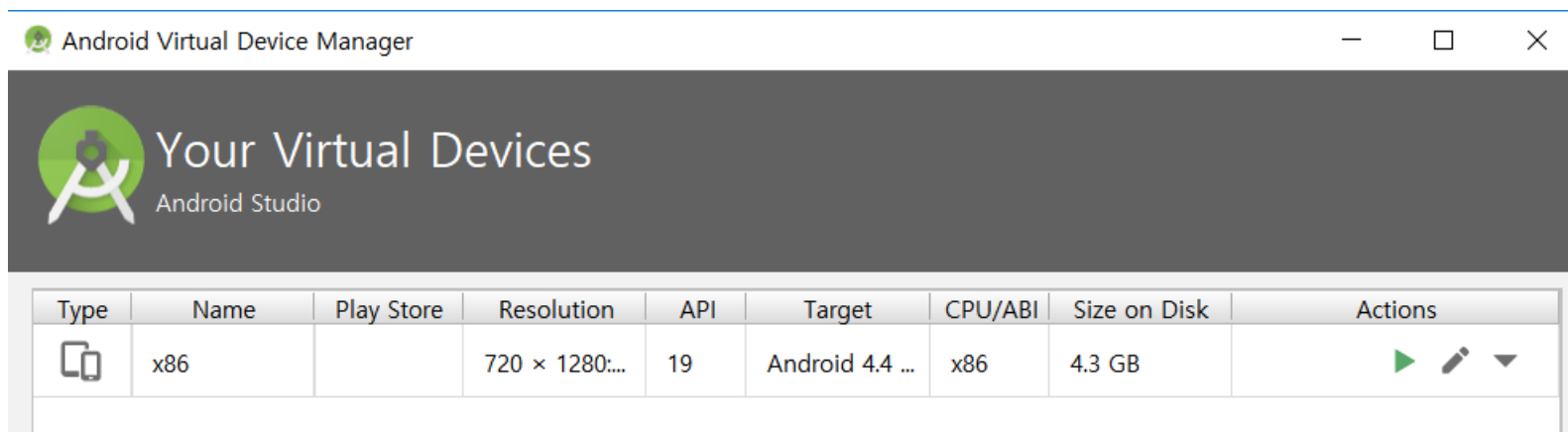
# 에뮬레이터 실행

▶ 다양한 실습을 위하여 안드로이드 에뮬레이터를 실행

▶ Tools - AVD Manager



▶ 미리 생성한 안드로이드 에뮬레이터를 실행



# 애몰레이터 실행

## ▶ 애몰레이터 테스트 환경



Extended controls - x86:5554

- Location
- Cellular
- Battery
- Camera
- Phone
- Directional pad
- Microphone
- Fingerprint
- Virtual sensors
- Bug report
- Snapshots
- Record and Playback
- Settings
- Help

Accelerometer Additional sensors

☒ Rotate ☐ Move

Z-Rot 35.3  
X-Rot -33.1  
Y-Rot 28.5

Device rotation

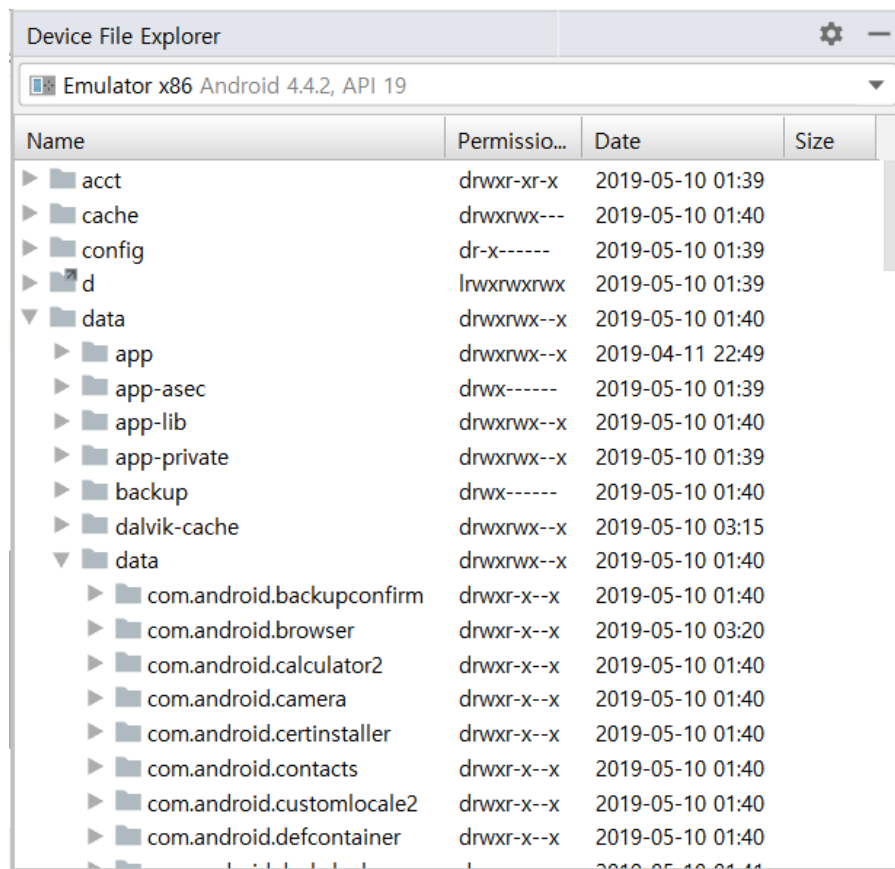
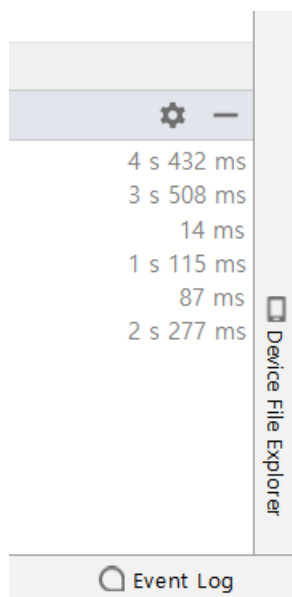
Resulting values

Accelerometer (m/s <sup>2</sup> ):	2.66	8.18	4.71
Gyroscope (rad/s):	0.00	0.00	0.00
Magnetometer (μT):	32.65	15.33	-32.81
Rotation:	ROTATION_0		

# 에뮬레이터 실행

## ▶ 에뮬레이터 내부 파일 시스템 구조

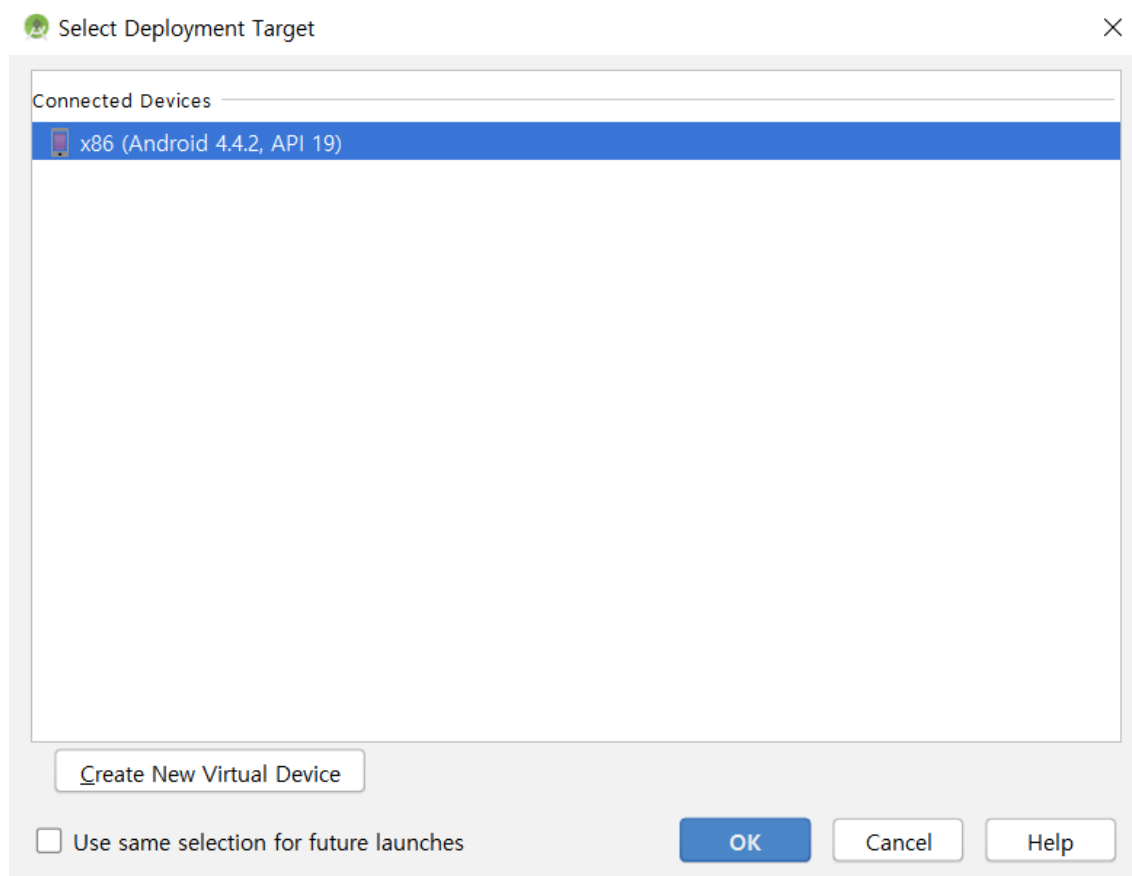
### ▶ 안드로이드 스튜디오의 오른쪽 아래에 Device File Explorer 클릭



# 에뮬레이터 실행

## ▶ 에뮬레이터에 앱 설치

### ▶ 8주차 과제인 일기장 앱 설치



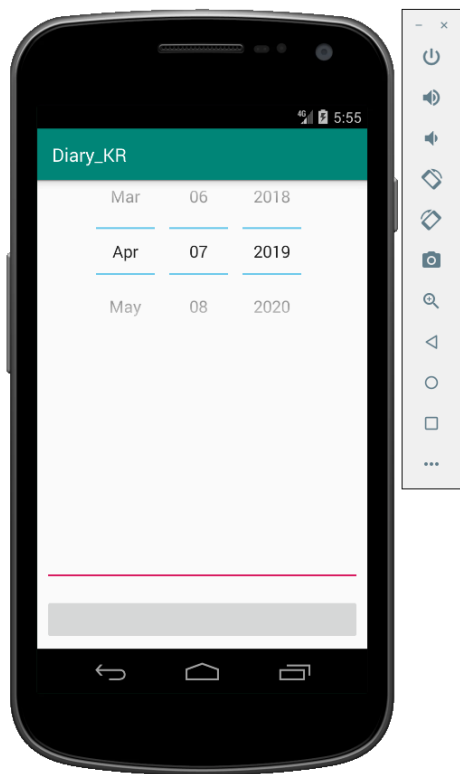
# 애몰레이터 실행

## ▶ 애몰레이터에 앱 설치

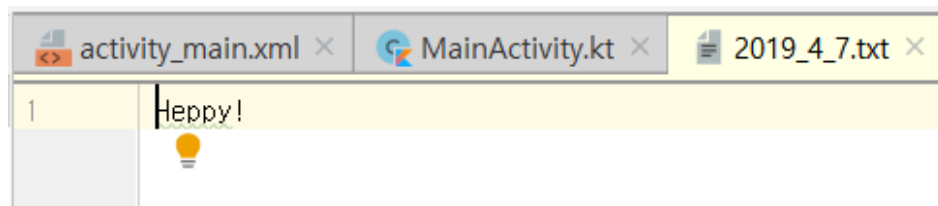
▶ 테스트용으로 일기를 저장한 후 앱 내부 저장소에서 저장된 일기파일 확인

▶ 기본적으로 data/data/ 하위에 앱 디렉터리 생성

▶ 해당 문서를 확인하려면 더블 클릭



▼	kr.co.korearental.diary_kr	drwxr-x--x	2019-05-
▶	cache	drwxrwx--x	2019-05-
▼	files	drwxrwx--x	2019-05-
	2019_4_7.txt	-rw-rw----	2019-05-
▶	lib	lrwxrwxrwx	2019-05-
▶	dontpanic	drwxr-x---	2019-05-



# 실습 예제 1

## ▶ 파일 저장 실습

- ▶ 8주차 과제를 외부저장소에 저장
- ▶ 액티비티를 추가하여 저장된 일기목록을 리스트 뷰로 나타내기
  - ▶ 앱을 실행하면 저장된 일기목록을 보여주는 화면이 보여지고 목록 중에 하나를 선택하면 수정화면으로 전환, 추가하기 버튼을 누르면 새로쓰기 화면으로 전환
  - ▶ 수정화면은 해당 날짜로 캘린더 뷰가 변경되어 있으며 저장된 일기 내용이 에디트 텍스트에 보여지고 버튼은 “수정하기”로 변경
  - ▶ 새로쓰기 화면은 캘린더 뷰가 오늘 날짜로 되어 있으며 에디트 텍스트에는 아무 내용이 없으며 버튼은 “새로저장”으로 변경
- ▶ 다음 슬라이드의 파일 관리 및 리스트 뷰 관련 자료를 학습 후 진행



# 파일 목록 가져오기

---

## ▶ 저장된 파일 리스트 가져오는 방법

### ▶ `filesDir`

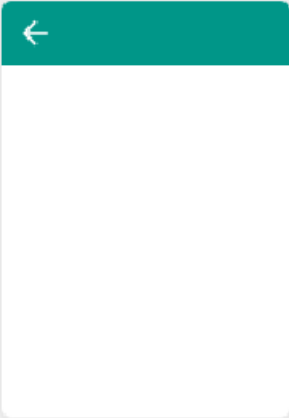
▷ 내부 파일이 저장된 파일 시스템 디렉터리의 절대 경로 정보를 가져옴

### ▶ `listFiles()`

▷ 앱이 현재 저장한 파일명의 문자열 배열을 반환

# 파일 목록 가져오기

## ▶ 액티비티 추가



### Creates a new empty activity

Activity Name:

☒ Generate Layout File

Layout Name:

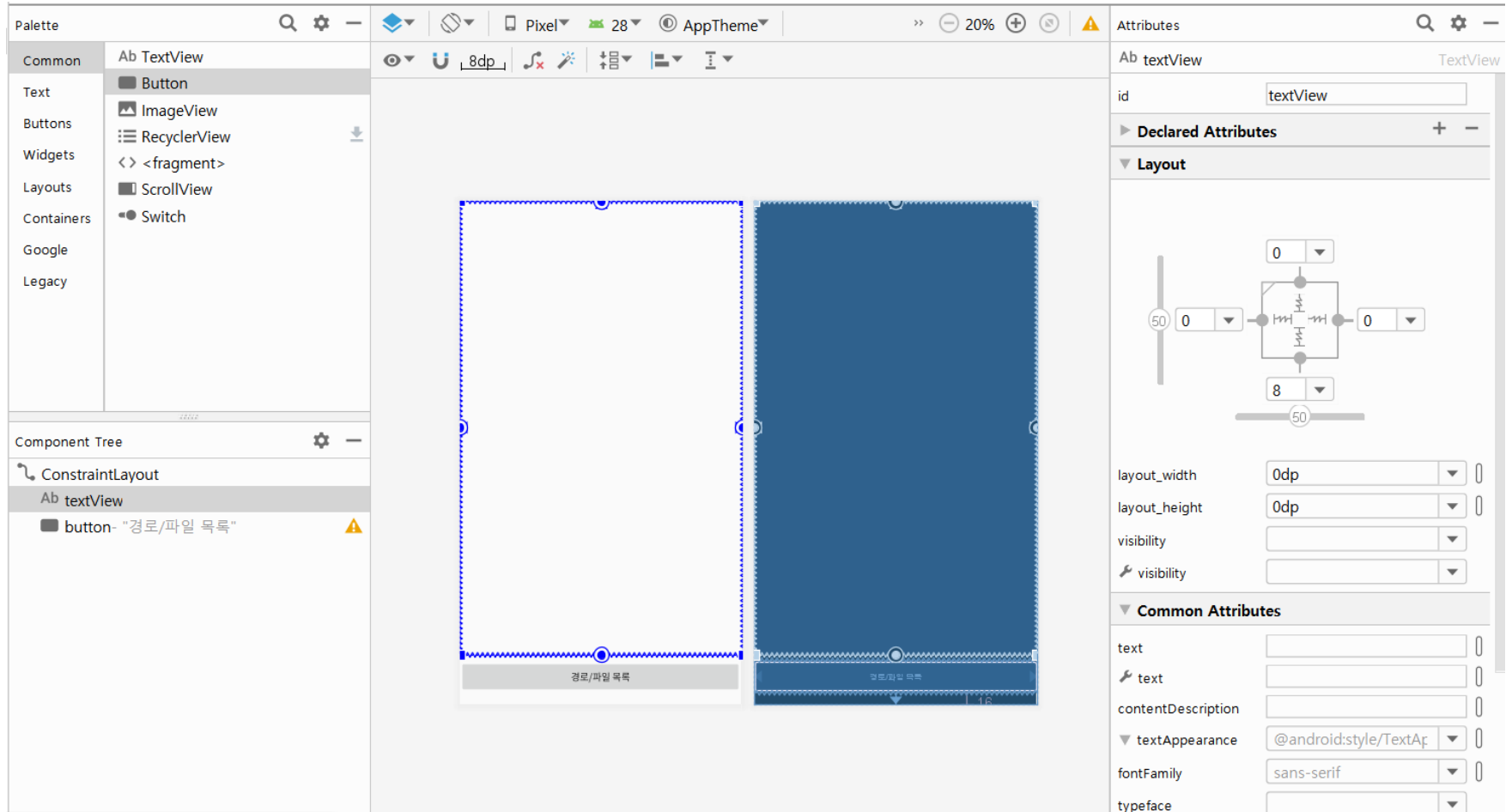
☐ Launcher Activity

Package name:  ▼

Source Language:  ▼

# 파일 목록 가져오기

## ▶ 액티비티 디자인



# 파일 목록 가져오기

## ▶ 액티비티 구현

```
class FileListTest : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_file_list_test)  
  
        button.setOnClickListener {  
            var fileList = filesDir.listFiles()  
            textView.text = Arrays.toString(fileList)  
        }  
    }  
}
```

# Listview Review

## ▶ 리스트뷰 구성하기

- ▶ 특정 화면이 반복적으로 내용만 다르게 추가할 경우에 주로 리스트 뷰를 사용

## ▶ 리스트 뷰 작성 순서

- ▶ 데이터를 정의하고 배열에 저장

- ▶ 리스트 뷰를 화면에 배치하고 정의

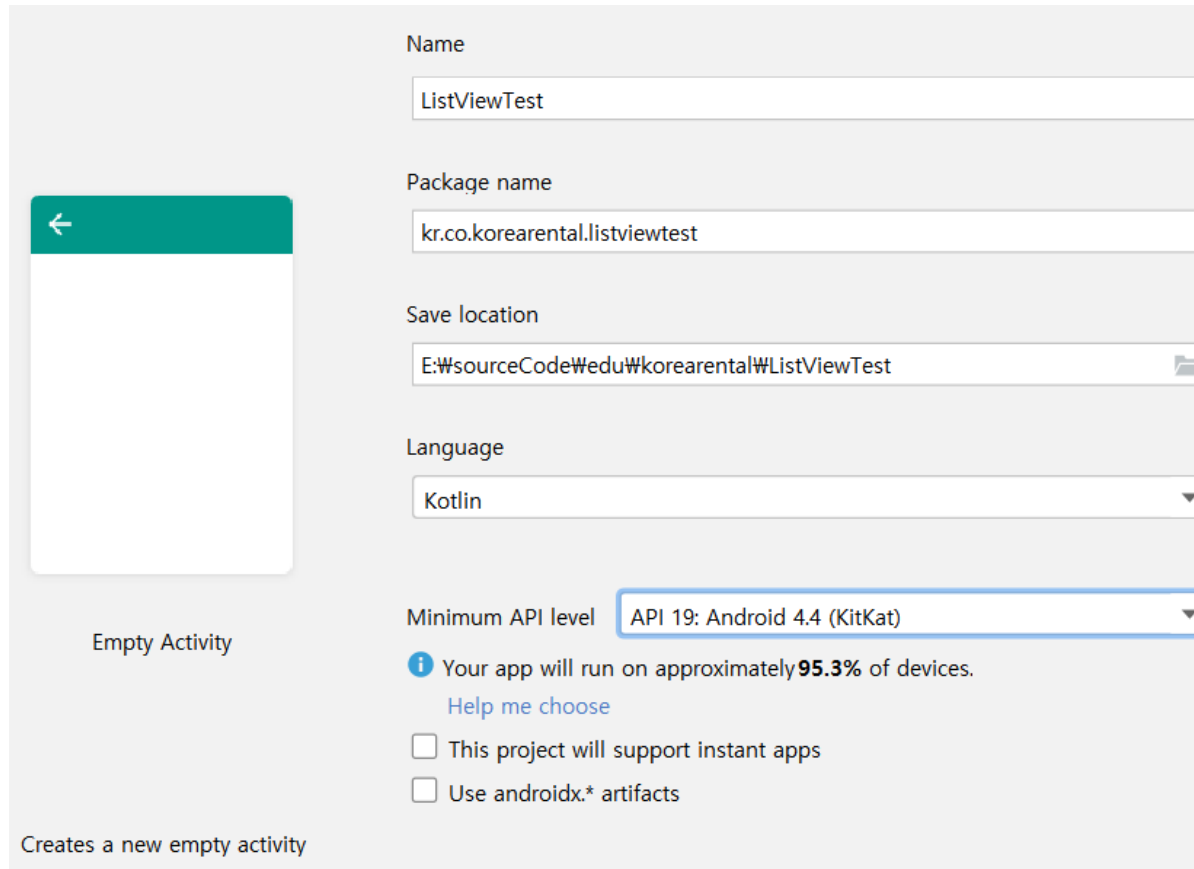
- ▶ 데이터와 리스트뷰를 관리할 어댑터를 생성 - 이 부분이 복잡

- ▶ 리스트뷰의 setAdapter 메소드를 사용하여 생성된 Adapter 객체를 리스트뷰에 넘겨서 리스트를 표현

# Listview Review

## ▶ 리스트뷰 구성하기

### ▶ 예제 프로젝트 생성 : ListViewTest



Empty Activity

Creates a new empty activity

Name  
ListViewTest

Package name  
kr.co.korearental.listviewtest

Save location  
E:\sourceCode\Wedu\Wkorearental\WListViewTest

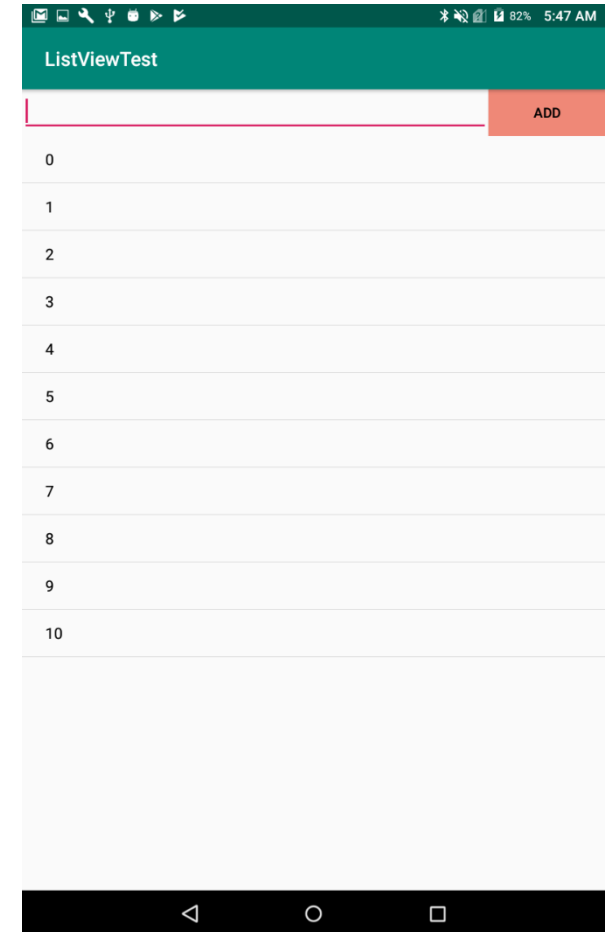
Language  
Kotlin

Minimum API level  
API 19: Android 4.4 (KitKat)

**i** Your app will run on approximately **95.3%** of devices.  
[Help me choose](#)

☐ This project will support instant apps

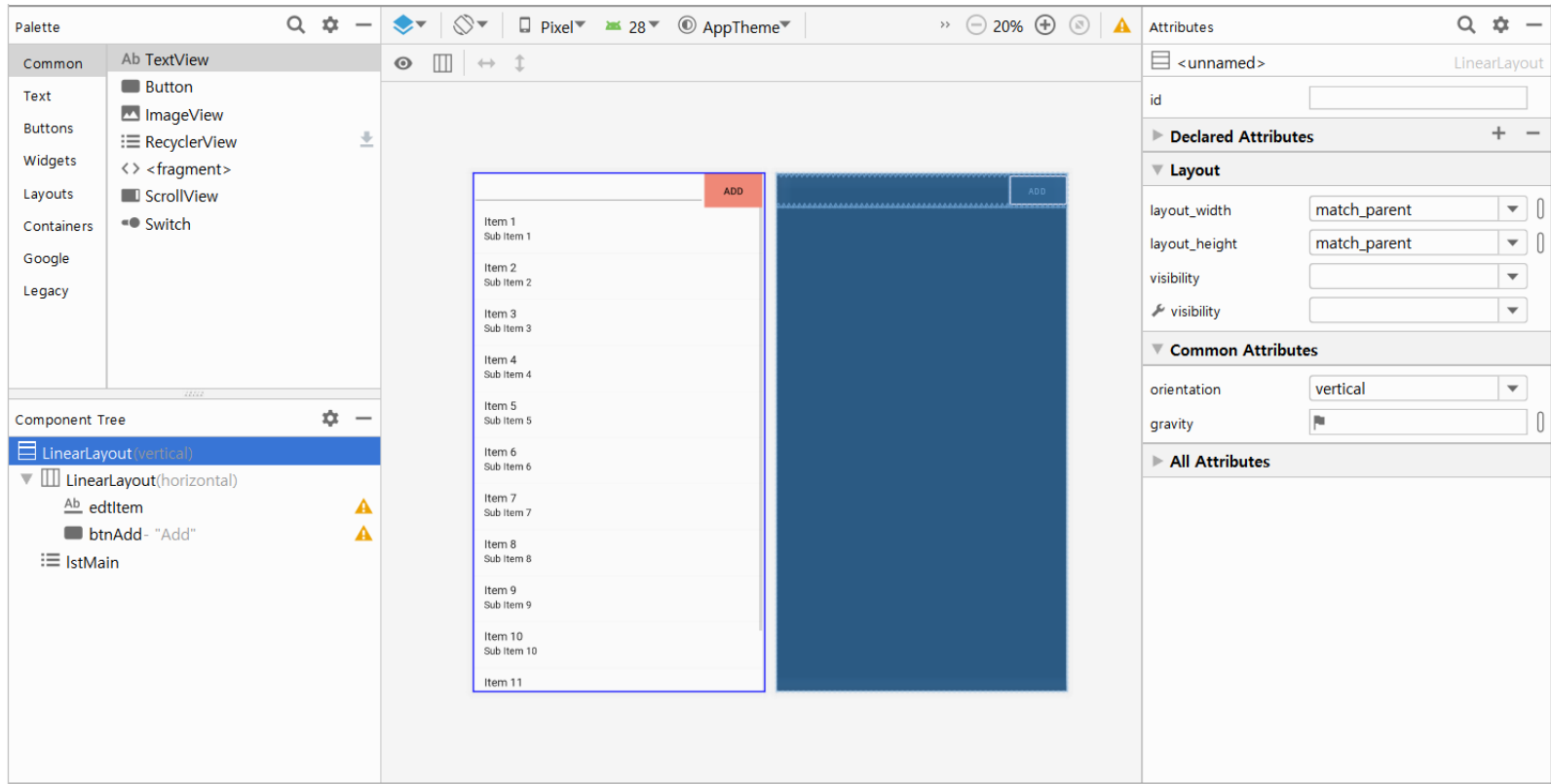
☐ Use androidx.\* artifacts



# Listview Review

## ▶ activity\_main.xml 구현

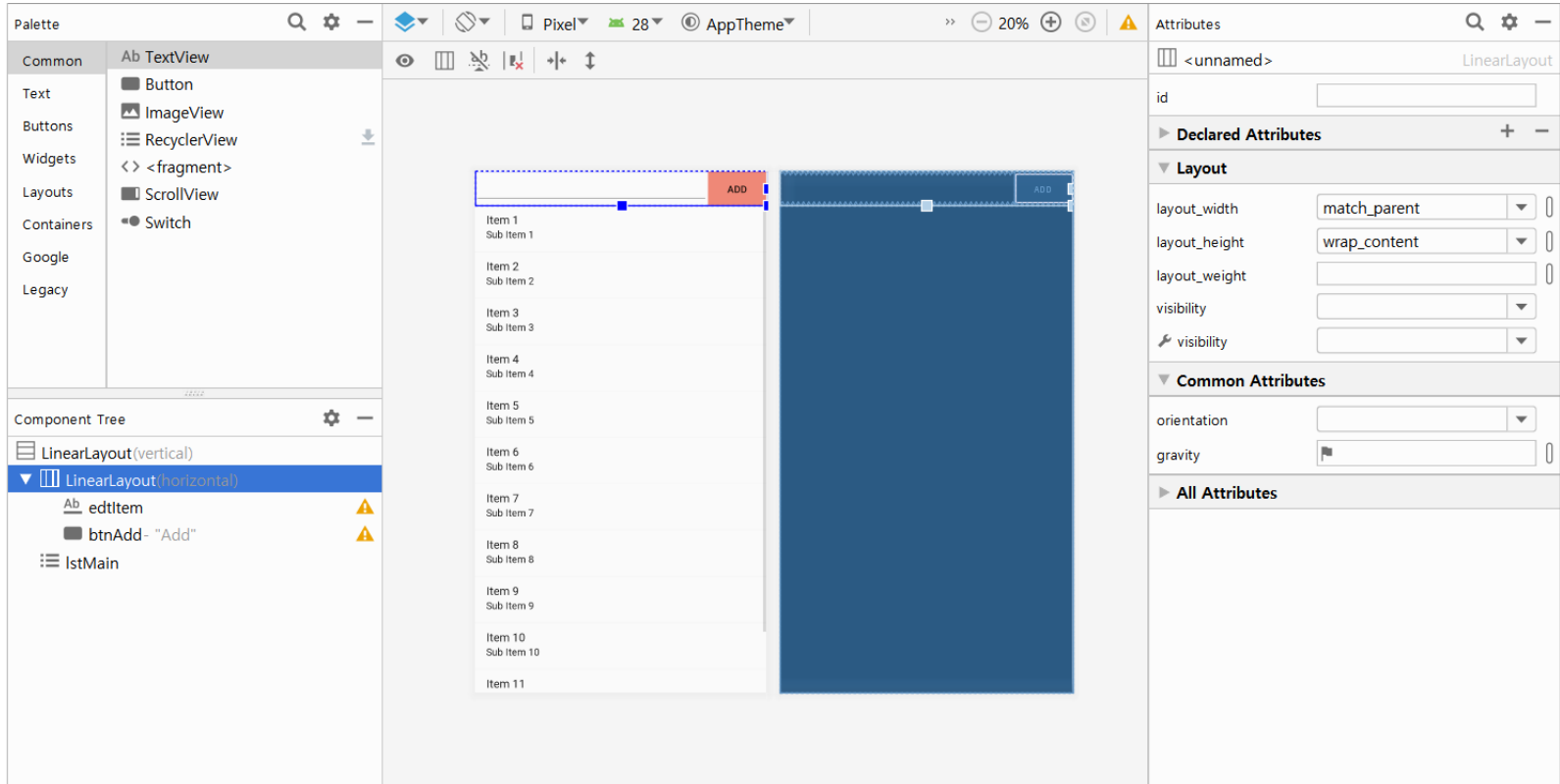
### ▶ 선형 레이아웃으로 작성



# Listview Review

## ▶ activity\_main.xml 구현

▶ 리스트에 내용을 삽입하기 위하여 수평형 선형 레이아웃에 에디트 텍스트와 버튼 추가

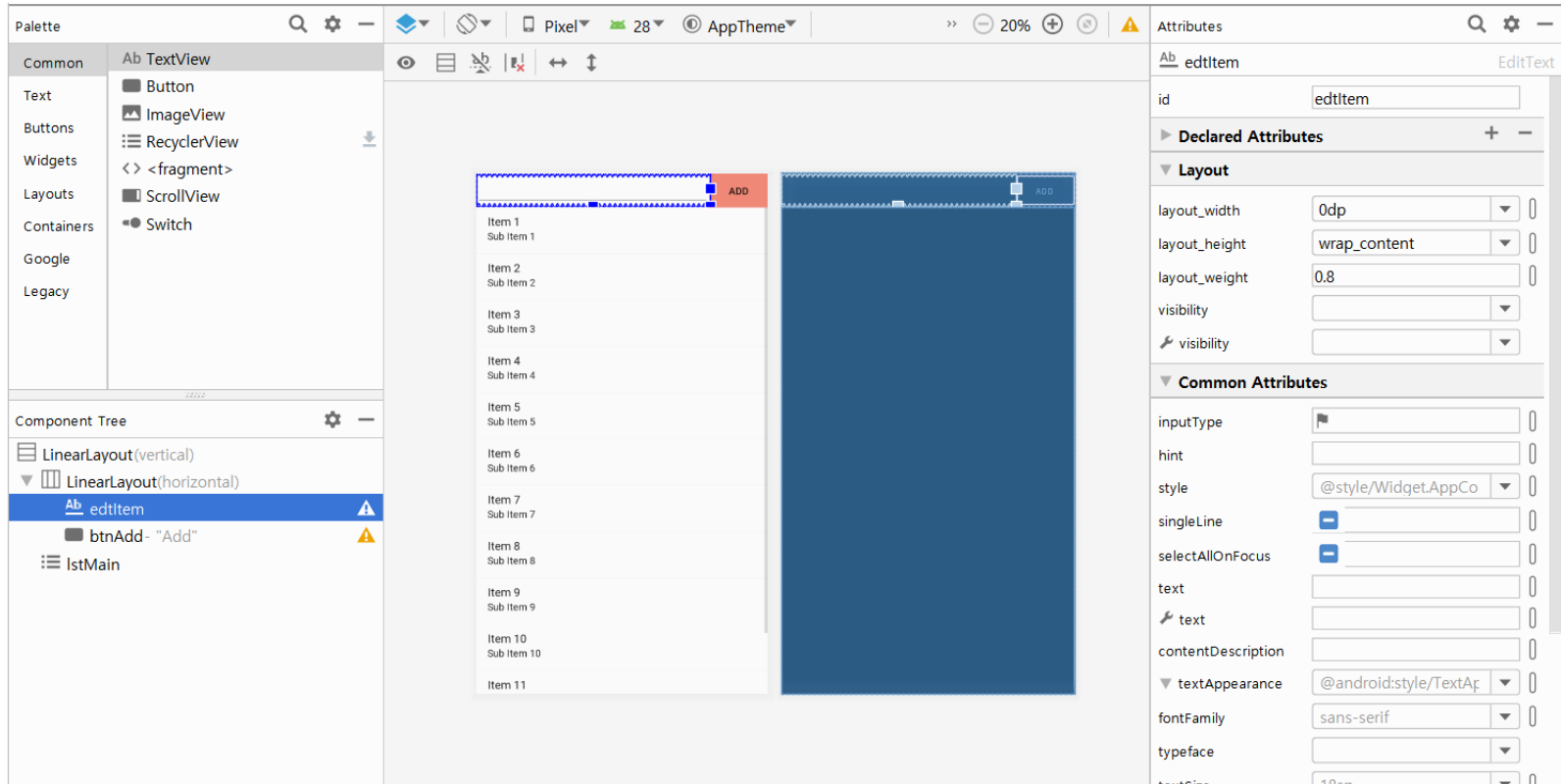




# Listview Review

## ▶ activity\_main.xml 구현

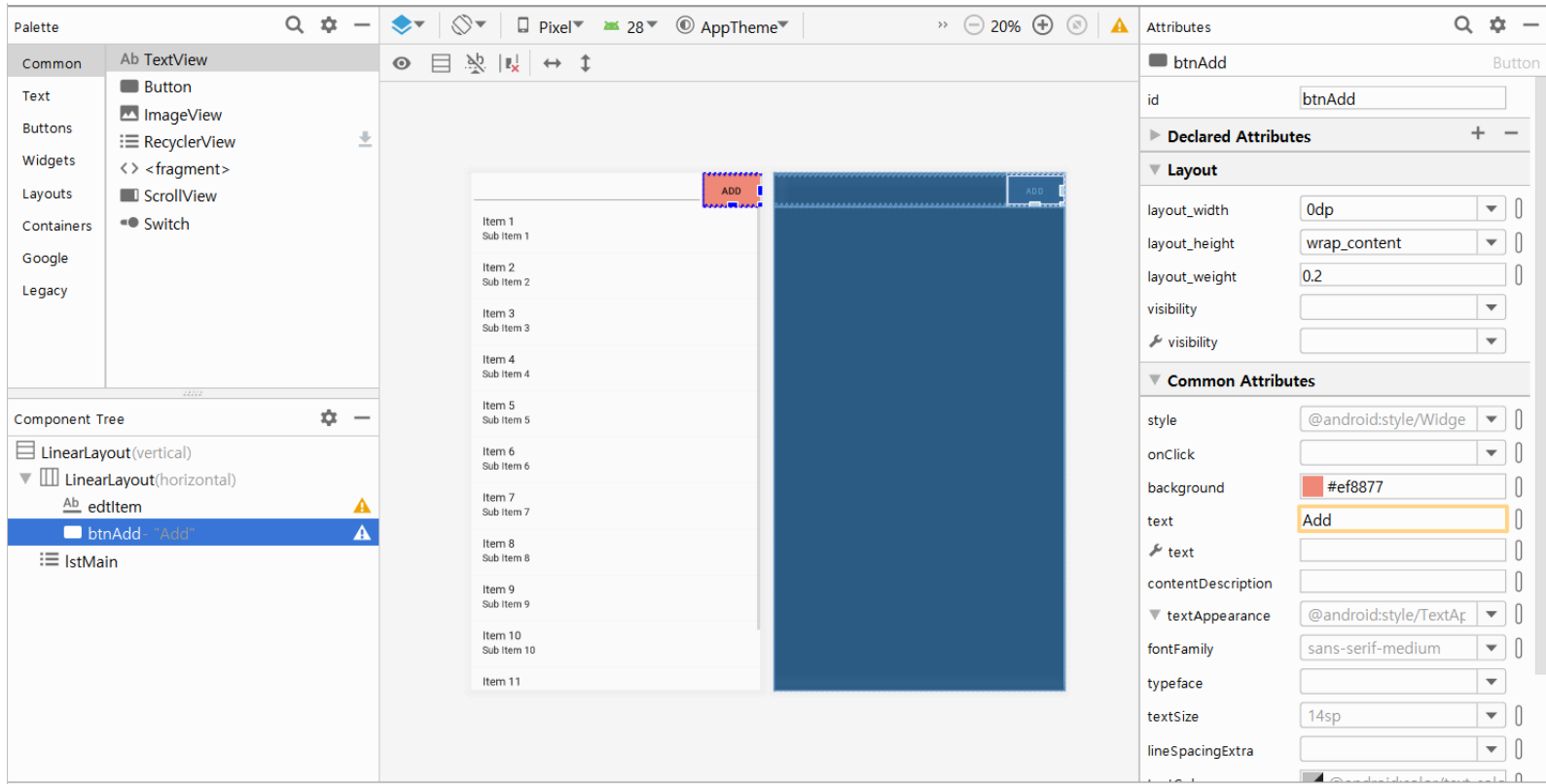
▶ 리스트에 내용을 삽입하기 위하여 수평형 선형 레이아웃에 에디트 텍스트와 버튼 추가



# Listview Review

## ▶ activity\_main.xml 구현

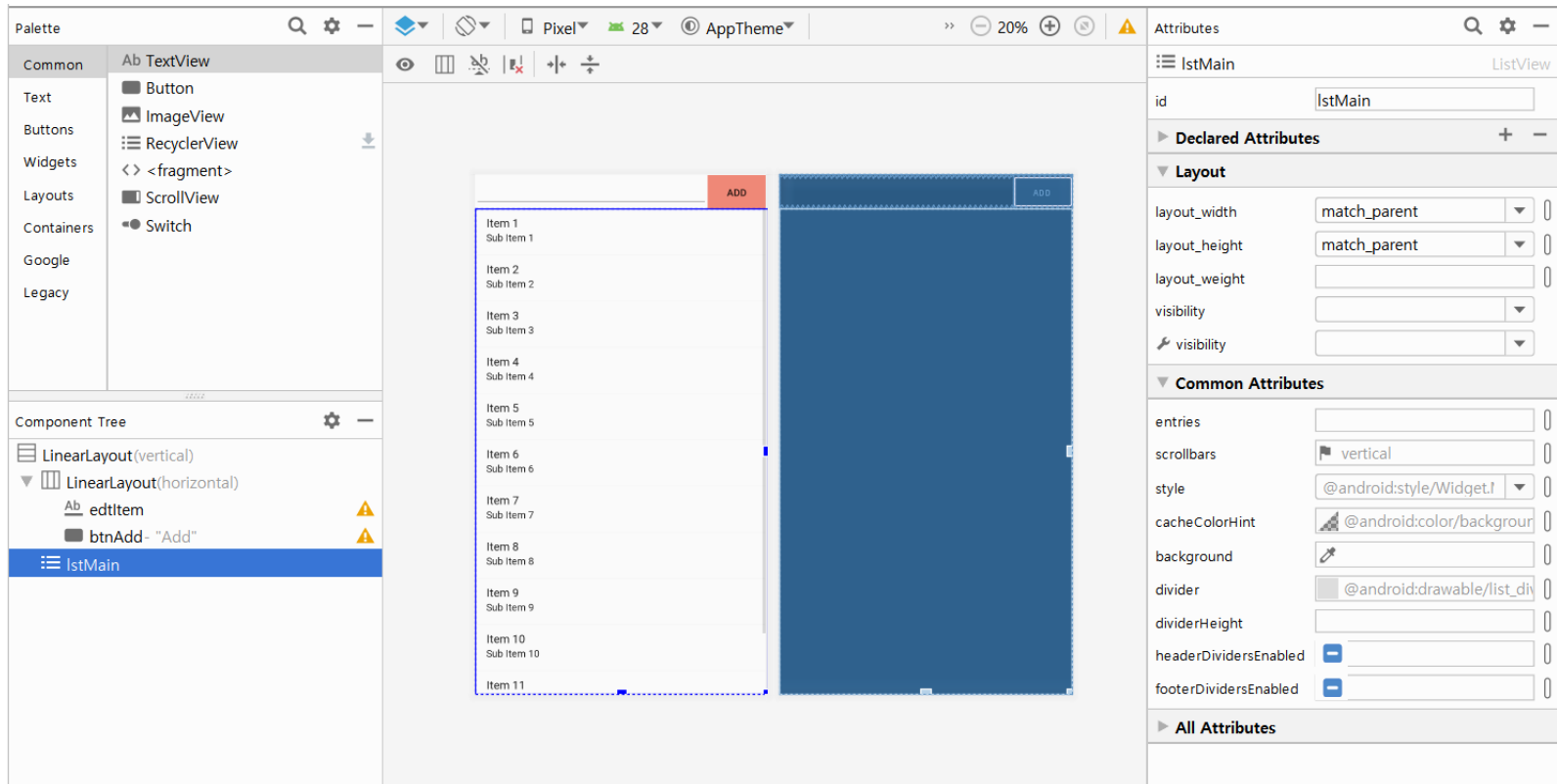
▶ 리스트에 내용을 삽입하기 위하여 수평형 선형 레이아웃에 에디트 텍스트와 버튼 추가



# Listview Review

## ▶ activity\_main.xml 구현

### ▶ 리스트뷰 추가



# Listview Review

## ▶ activity\_main.xml 작성(1)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:weightSum="1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <EditText
            android:id="@+id/edtItem"
            android:layout_weight="0.8"
            android:layout_width="0dp"
            android:layout_height="wrap_content"/>
```

# Listview Review

## ▶ activity\_main.xml 작성(2)

```
<Button
    android:id="@+id/btnAdd"
    android:text="Add"
    android:background="#ef8877"
    android:layout_weight="0.2"
    android:layout_width="0dp"
    android:layout_height="wrap_content"/>

</LinearLayout>

<ListView
    android:id="@+id/lstMain"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</ListView>

</LinearLayout>
```

# Listview Review

## ▶ MainActivity.kt 구현(1)

```
class MainActivity : AppCompatActivity() {  
  
    // nullable하게 lazy 사용  
    val dataList by lazy{  
        mutableListOf<String>();  
    }  
    //이미 만들어져 있는 단순한 Adapter 클래스이며 커스터마이징이 불가  
    var adapter : ArrayAdapter<String>? = null  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        setUpUI()  
    }  
  
    private fun setUpUI() {  
        setUpListView()  
        btnAdd.setOnClickListener {  
            dataList.add(editItem.text.toString())  
            //Adapter 클래스의 getCount()와 getView()호출하여 리스트뷰 갱신  
            adapter?.notifyDataSetChanged()  
        }  
    }  
}
```

# Listview Review

## ▶ MainActivity.kt 구현(2)

```
private fun setUpListView() {  
    // Data 추가  
    (0..10).map { dataList.add(it.toString()) }  
    // 간단한 adapter 생성  
    adapter = ArrayAdapter(this, android.R.layout.simple_list_item_1, dataList)  
    // adapter 설정  
    lstMain.adapter = adapter  
    // Item 클릭 이벤트 핸들러 설정, 사용자가 리스트에서 클릭한 아이템(position)의 번호를 내부에 전달  
    lstMain.setOnItemClickListener { parent, view, position, id ->  
        // 삭제하기  
        dataList.removeAt(position)  
        // Adapter 클래스의 getCount()와 getView() 호출하여 리스트뷰 갱신  
        adapter?.notifyDataSetChanged()  
    }  
}
```

# SQLite 개요

## ▶ SQLite의 활용

- ▶ -앱을 개발하는 과정에서 인터넷이 연결되지 않는 상황을 고려해야 함
  - ▷ 무선 네트워크는 유선에 비하여 신뢰성이 비교적 낮음
  - ▷ 서버에서 받은 데이터의 일정 부분을 스마트폰에 저장하여 네트워크 연결이 원활하지 않은 상황에 대응해야 함
- ▶ 좋은 사용자 경험을 제공하려면 인터넷에 연결되지 않더라도 앱의 일부분이 동작할 수 있어야 함
  - ▷ 엘리베이터 등의 통신이 불가능한 지역에서도 사용자는 앱을 이용하려 시도하는 경향이 있음
- ▶ 앱에 데이터를 영구적으로 저장할 수 있는 메커니즘이 필요
  - ▷ sharedPreference는 앱 설정 등의 적은 양의 데이터를 저장할 때 유용
  - ▷ 데이터 베이스는 구조화된 다량의 데이터를 저장해야 할 경우 유용



# SQLite 개요

## ▶ 데이터베이스 정의

- ▶ 대용량의 데이터를 체계적으로 구성하고 관리하는 것
- ▶ 지속적이고 대량으로 발생하는 정보를 보관하기 위해 사용

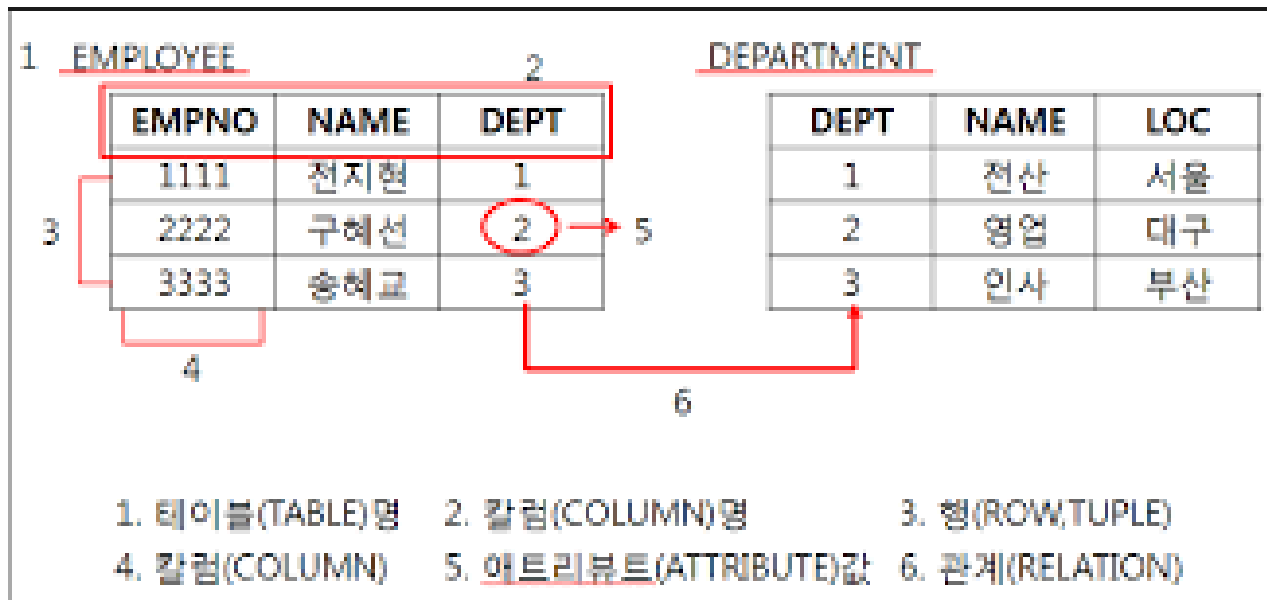
## ▶ 데이터베이스 관리 시스템(DBMS: DataBase Management System)

- ▶ 다양한 형태의 지속적이고 대량으로 발생하는 정보를 보관하려면 기존의 파일 시스템으로는 한계가 있음
- ▶ 데이터베이스는 여러 사용자나 시스템이 서로 공유할 수 있어야 함
- ▶ 데이터베이스 관리 시스템은 이러한 데이터베이스를 관리해주는 시스템 또는 소프트웨어를 말함
- ▶ DBMS는 크게 계층형(Hierarchical), 망형(Network), 관계형(Relational), 객체지향형(Object-Oriented), 객체관계형(Object-Relational) DBMS 등의 유형으로 나뉨
- ▶ PC 환경에서 주로 운영되는 DBMS는 MS의 MSSQL Server, Access / Oracle의 Oracle Database, MySQL / IBM의 DB2 등이 있음

# SQLite 개요

## ▶ 관계형 데이터베이스

- ▶ DBMS 중 가장 많이 사용되는 유형
- ▶ 키(key)와 값(value)들의 간단한 관계를 테이블화 시킨 데이터 베이스
  - ▶ 테이블의 각 로우(row)에는 고유한 키가 존재하며 다른 테이블들의 로우와 연결 가능
  - ▶ 일대일, 일대다, 다대다 등의 테이블 간의 관계가 존재
- ▶ SQLite([www.sqlite.org](http://www.sqlite.org))은 오픈소스로 만들어진 관계형 DBMS



# SQLite 개요

## ▶ 관계형 데이터베이스의 특징

### ▶ 장점

- ▶ 업무가 변화할 경우에 다른 DBMS에 비해 변화에 쉽게 순응할 수 있는 구조
- ▶ 유지보수 측면에서도 편리
- ▶ 대용량 데이터 관리와 데이터 무결성(Integration)을 보장
  - 무결성 : 데이터의 정확성, 일관성, 유효성이 유지되는 것

### ▶ 단점

- ▶ 시스템 자원을 많이 차지해서 시스템이 전반적으로 느려짐

# SQLite 개요

## ▶ SQLite 특징

- ▶ SQLite는 임베디드 데이터베이스로 개발된 경량(Light-weight) RDBMS
  - ▶ PC에서 사용되는 DB는 자원이 한정되어 있는 스마트폰 환경에서 사용하기에 비효율적이기 때문에 안드로이드에서는 경량화된 SQLite를 사용
- ▶ 데이터 조회 속도가 빠르고 표준 SQL을 지원
  - ▶ 기존 웹이나 PC에서 사용되던 업무용 프로그램의 데이터 관리 기능을 그대로 사용 가능
  - ▶ 스마트폰 환경에 최적화(경량화)되어 있지만 기본적인 개념과 동작 방식은 PC에서 사용되는 DB와 대부분 동일

# SQLite 개요

## ▶ 데이터베이스 관련 용어

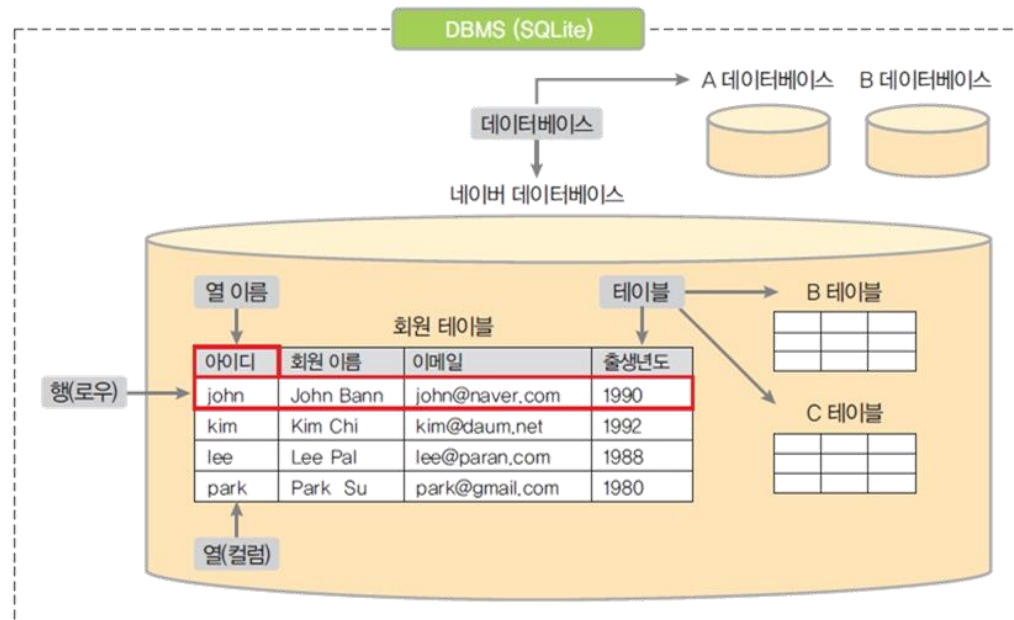
▶ 데이터 : 하나하나의 단편적인 정보를 뜻함

▶ 테이블 : 데이터가 표 형태로 표현된 것

▶ 데이터베이스(DB) : 테이블이 저장되는 장소로 주로 원통 모양으로 표현

▷ 각 데이터베이스는 서로 다른 고유한 이름이 있어야 함

▶ DBMS : 데이터베이스를 관리하는 시스템 또는 소프트웨어를 말함(ex: SQLite)



# SQLite 개요

## ▶ 데이터베이스 관련 용어

- ▶ 열(컬럼 또는 필드) : 각 테이블은 1개 이상의 열로 구성됨
- ▶ 열 이름 : 각 열을 구분하는 이름, 열 이름은 각 테이블 안에서는 중복되지 않아야 함
- ▶ 데이터 형식 : 열의 데이터 형식, 테이블을 생성할 때 열 이름과 함께 지정해줘야 함
- ▶ 행(로우) : 실제 데이터를 뜻함
- ▶ SQL(Structured Query Language, 구조화된 질의 언어) : 사용자와 DBMS가 소통하기 위한 언어
- ▶ 데이터 베이스 모델링 : 현실 세계에서 사용되는 데이터를 DBMS안에 어떻게 옮겨 놓을지를 결정하는 과정



# SQLite 개요

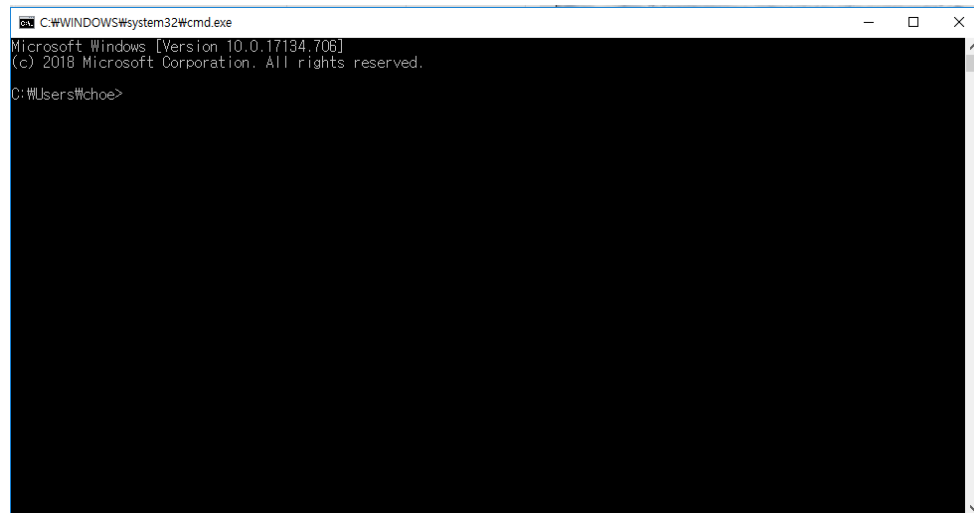
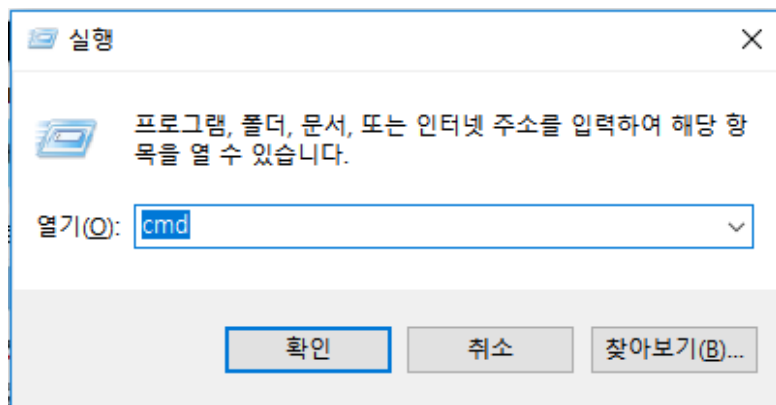
## ▶ 한국렌탈 데이터 베이스 구축하기

▶ 에뮬레이터가 실행 중인 상태에서 에뮬레이터 내부에 데이터베이스 구축

▶ 명령 프롬프트 실행

▷ [윈도우 키] + r

▷ cmd 입력 후 엔터



# SQLite 개요

## ▶ 한국렌탈 데이터 베이스 구축하기

▶ adb(android device bridge)를 사용하여 에뮬레이터나 기기 내부에 진입

▶ 명령 프롬프트에서 adb.exe가 있는 폴더로 이동

▶ 디폴트 경로 : C:\Users\choe\AppData\Local\Android\Sdk\platform-tools

이름	수정한 날짜	유형	크기
api	2019-05-10 오후 7:47	파일 폴더	
lib64	2019-05-10 오후 7:47	파일 폴더	
systrace	2019-05-10 오후 7:47	파일 폴더	
adb.exe	2019-05-10 오후 7:47	응용 프로그램	1,928KB
AdbWinApi.dll	2019-05-10 오후 7:47	응용 프로그램 확장	96KB
AdbWinUsbApi.dll	2019-05-10 오후 7:47	응용 프로그램 확장	62KB
deployagent	2019-05-10 오후 7:47	파일	1KB
deployagent.jar	2019-05-10 오후 7:47	Executable Jar File	875KB
deploypatchgenerator.jar	2019-05-10 오후 7:47	Executable Jar File	3,830KB
dmtracedump.exe	2019-05-10 오후 7:47	응용 프로그램	194KB
etc1tool.exe	2019-05-10 오후 7:47	응용 프로그램	362KB
fastboot.exe	2019-05-10 오후 7:47	응용 프로그램	1,336KB
hprof-conv.exe	2019-05-10 오후 7:47	응용 프로그램	40KB
libwinpthread-1.dll	2019-05-10 오후 7:47	응용 프로그램 확장	207KB
make_f2fs.exe	2019-05-10 오후 7:47	응용 프로그램	395KB
mke2fs.conf	2019-05-10 오후 7:47	CONF 파일	2KB
mke2fs.exe	2019-05-10 오후 7:47	응용 프로그램	1,002KB
NOTICE.txt	2019-05-10 오후 7:47	텍스트 문서	312KB
package.xml	2019-05-10 오후 7:47	XML 문서	18KB
source.properties	2019-05-10 오후 7:47	PROPERTIES 파일	1KB
sqlite3.exe	2019-05-10 오후 7:47	응용 프로그램	1,210KB



# SQLite 개요

## ▶ 한국렌탈 데이터 베이스 구축하기

▶ cd(change directory) 명령어를 이용하여 앞의 폴더로 이동

▶ cd C:\Users\choe\AppData\Local\Android\Sdk\platform-tools

▶ dir 명령어로 현재 작업 디렉터리 내부에 adb.exe 존재 여부 확인

```
C:\Users\choe>cd C:\Users\choe\AppData\Local\Android\Sdk\platform-tools
```

```
C:\Users\choe\AppData\Local\Android\Sdk\platform-tools>dir
```

C 드라이브의 볼륨에는 이름이 없습니다.  
볼륨 일련 번호: CCF7-71AE

C:\Users\choe\AppData\Local\Android\Sdk\platform-tools 디렉터리

2019-05-10	오후 07:47	<DIR>	.
2019-05-10	오후 07:47	<DIR>	
2019-05-10	오후 07:47	1,974,272	adb.exe
2019-05-10	오후 07:47	97,792	AdbWinApi.dll
2019-05-10	오후 07:47	62,976	AdbWinUsbApi.dll

# SQLite 개요

## ▶ 한국렌탈 데이터 베이스 구축하기

### ▶ adb shell 명령어로 안드로이드 에뮬레이터 내부로 진입

▷ adb shell      #진입할 수 있는 기기가 하나만 연결된 상태인 경우 바로 접속

### ▶ 아래와 같이 메시지가 보일 경우 여러 에뮬레이터가 동작 중이거나 실제 기기가 연결된 상태

```
C:\Users\choe\AppData\Local\Android\Sdk\platform-tools>adb shell  
error: more than one device/emulator
```

### ▶ 이럴 경우 에뮬레이터를 종료하거나 기기를 연결 해제

### ▶ 또는 접속하고자 하는 기기를 선택하여 진입

### ▶ 연결 가능한 기기 목록 보기

#### ▷ adb devices

```
C:\Users\choe\AppData\Local\Android\Sdk\platform-tools>adb devices  
List of devices attached  
87e34f03      device  
emulator-5554      device
```

# SQLite 개요

## ▶ 한국렌탈 데이터 베이스 구축하기

### ▶ 리스트에서 원하는 환경을 선택하여 진입

▶ adb -s [device name] shell #실제 기기는 파일시스템의 접근에 한계가 있음

```
C:\Users\choe\AppData\Local\Android\Sdk\platform-tools>adb -s emulator-5554 shell  
root@generic_x86:/ #
```

▶ 진입에 성공하면 프롬프트가 변경됨

### ▶ 다이어리 앱의 내부에 데이터베이스를 생성하기 위하여 해당 앱 디렉터리로 이동

▶ cd /data/data/kr.co.korearental.diary\_kr

### ▶ ls 명령어로 내부 디렉터리 구조 확인

▶ 안드로이드 스튜디오의 Device File Explorer와 구조가 동일

```
root@generic_x86:/data/data/kr.co.korearental.diary_kr # ls  
cache  
files  
lib
```

▼	kr.co.korearental.diary_kr	drwxr-x--x	2019-05-10 11:40
▶	cache	drwxrwx--x	2019-05-10 05:44
▶	files	drwxrwx--x	2019-05-10 05:54
▶	lib	lrwxrwxrwx	2019-05-10 11:40

# SQLite 개요

## ▶ 한국렌탈 데이터 베이스 구축하기

### ▶ 해당 앱 디렉터리에 데이터 베이스가 저장될 databases 디렉터를 생성

#### ▷ mkdir databases

- mkdir은 make directory의 약어

### ▶ 생성된 디렉터리로 이동

#### ▷ cd databases

### ▶ pwd 명령어로 현재 디렉터리 경로 확인

#### ▷ pwd

- pwd는 print working directory의 약어

```
root@generic_x86:/data/data/kr.co.korearental.diary_kr # mkdir databases
root@generic_x86:/data/data/kr.co.korearental.diary_kr # cd databases
root@generic_x86:/data/data/kr.co.korearental.diary_kr/databases # pwd
/data/data/kr.co.korearental.diary_kr/databases
```

### ▶ 기본적으로 SQLite를 이용한 데이터는 아래의 경로에 저장

#### ▷ data/data/[package\_name]/databases

# SQLite 개요

## ▶ 한국렌탈 데이터 베이스 구축하기

▶ sqlite3 명령을 실행하여 krDB라는 이름의 데이터베이스를 생성한 후 database 모드로 전환

▶ sqlite3 krDB

```
/data/data/kr.co.korearental.diary_kr/databases  
3 krDB  
SQLite version 3.7.11 2012-03-20 11:35:50  
Enter ".help" for instructions  
Enter SQL statements terminated with a ";"  
sqlite>
```

# SQLite 개요

## ▶ 한국렌탈 데이터 베이스 구축하기

### ▶ 테이블 생성

▷ CREATE TABLE 테이블이름 (열이름1 데이터형식, 열이름2 데이터형식, ...);

▷ 아래 그림과 동일한 스키마의 테이블 생성

열 이름

회원 테이블

아이디	회원 이름	이메일	출생년도
john	John Bann	john@naver.com	1990
kim	Kim Chi	kim@daum.net	1992
lee	Lee Pal	lee@paran.com	1988
park	Park Su	park@gmail.com	1980

▷ CREATE TABLE memberTable (id char(10), userName char(20), email char(30), birthYear int);

```
|sqlite> CREATE TABLE memberTable (id char(10), userName char(20), email char(30), birthYear int);
```

■ 참고로 데이터베이스에서 테이블을 삭제하는 SQL문은 ➔ DROP TABLE 테이블명

# SQLite 개요

## ▶ 한국렌탈 데이터 베이스 구축하기

### ▶ 생성된 테이블 목록 확인

#### ▷ .table

```
sqlite> .table  
memberTable
```

### ▶ 특정 테이블의 스키마(속성) 확인

#### ▷ .schema memberTable

```
sqlite> .schema memberTable  
CREATE TABLE memberTable (id char(10), userName char(20), email char(30), birthYear int);
```

# SQLite 개요

## ▶ SQL문 작성 방법

### ▶ SQL문은 대소문자를 구분하지 않음

▷ 예외적으로 SQLite의 자체 명령은 소문자로 입력하고 명령의 시작부분에 점.을 붙이임

### ▶ 모든 SQL문의 끝에는 세미콜론;을 붙임

▷ SQLite의 자체 명령은 끝에 세미콜론을 붙이지 않아도 됨

### ▶ 자주 사용하는 SQLite 명령

▷ .table : 현재 데이터베이스의 테이블 목록을 출력

▷ .schema [table name] 테이블의 열, 데이터 형식의 정보를 출력

▷ .header on : SELECT문으로 출력할 때 헤더(필드명)를 보여줌

▷ .mode column : SELECT문으로 출력할 경우 컬럼 모드로 출력(필드를 기준으로 정렬)

▷ .exit : SQLite를 종료



# SQLite 개요

## ▶ 데이터 입력

▶ INSERT INTO 테이블명 VALUES (값1, 값2, ...);

▷ 열의 순서대로 입력

▶ 아래 테이블과 같이 데이터 입력

아이디	회원 이름	이메일	출생년도
john	John Bann	john@naver.com	1990
kim	Kim Chi	kim@daum.net	1992
lee	Lee Pal	lee@paran.com	1988
park	Park Su	park@gmail.com	1980

▶ 2개의 레코드만 입력한 후 뒷 페이지에서 .header on과 .mode column 명령으로 결과 화면 출력

▷ INSERT INTO memberTable VALUES('john', 'John Bann', 'john@naver.com', 1990);

▷ INSERT INTO memberTable VALUES('kim', 'Kim Chi', 'kim@daum.net', 1992);

- 문법을 익히기 위하여 직접 입력하거나 강의자료에서 카피한 후 cmd창에 우클릭하여 붙여넣기

# SQLite 개요

## ▶ 데이터 조회

### ▶ 입력한 데이터 확인

▶ .header on

▶ .mode column

▶ SELECT \* FROM memberTable;

```
sqlite> .header on
sqlite> .mode column
sqlite> SELECT * FROM memberTable;
```

id	userName	email	birthYear
john	John Bann	john@naver.com	1990
kim	Kim Chi	kim@daum.net	1992

# SQLite 개요

## ▶ 데이터 입력

▶ 나머지 레코드는 SQLite 관리 프로그램을 이용하여 입력

▶ SQLite Database Browser 활용

■ <https://sqlitebrowser.org/dl/>



## Downloads

### Windows

Our latest release (3.11.2) for Windows:

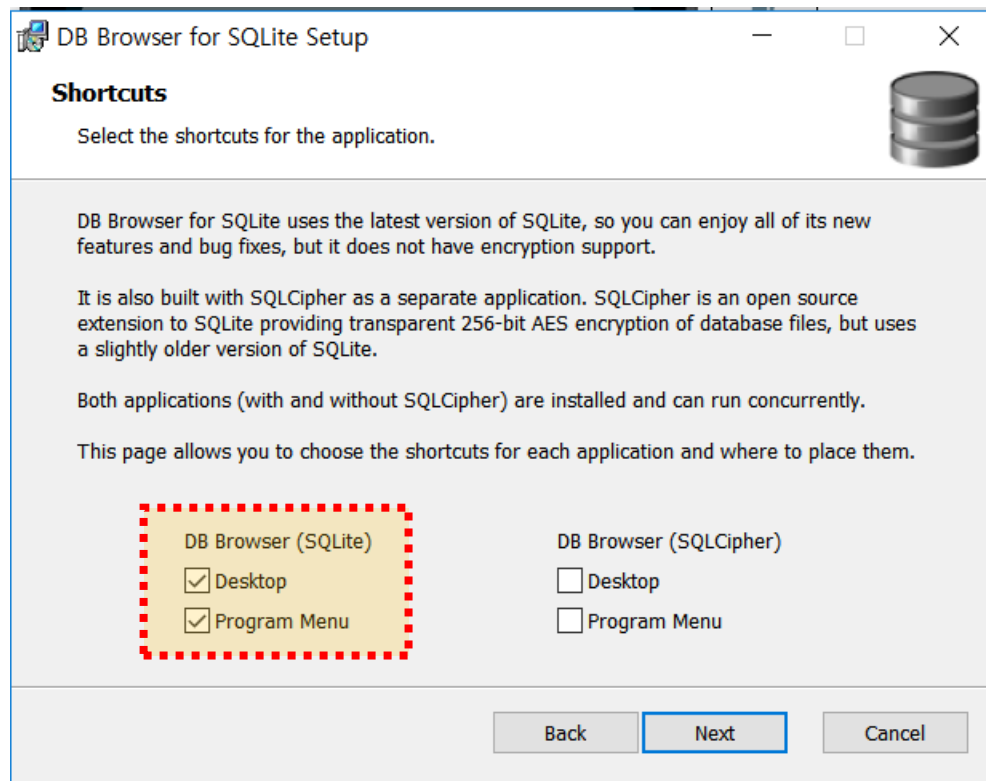
- DB Browser for SQLite - Standard installer for 32-bit Windows & Windows XP
- DB Browser for SQLite - .zip (no installer) for 32-bit Windows & Windows XP
- DB Browser for SQLite - Standard installer for 64-bit Windows
- DB Browser for SQLite - .zip (no installer) for 64-bit Windows
- Note - There's no PortableApp version for 3.11.1 (yet). It'll hopefully be ready in a few days.

# SQLite 개요

## ▶ 데이터 입력

### ▶ SQLite Database Browser 설치

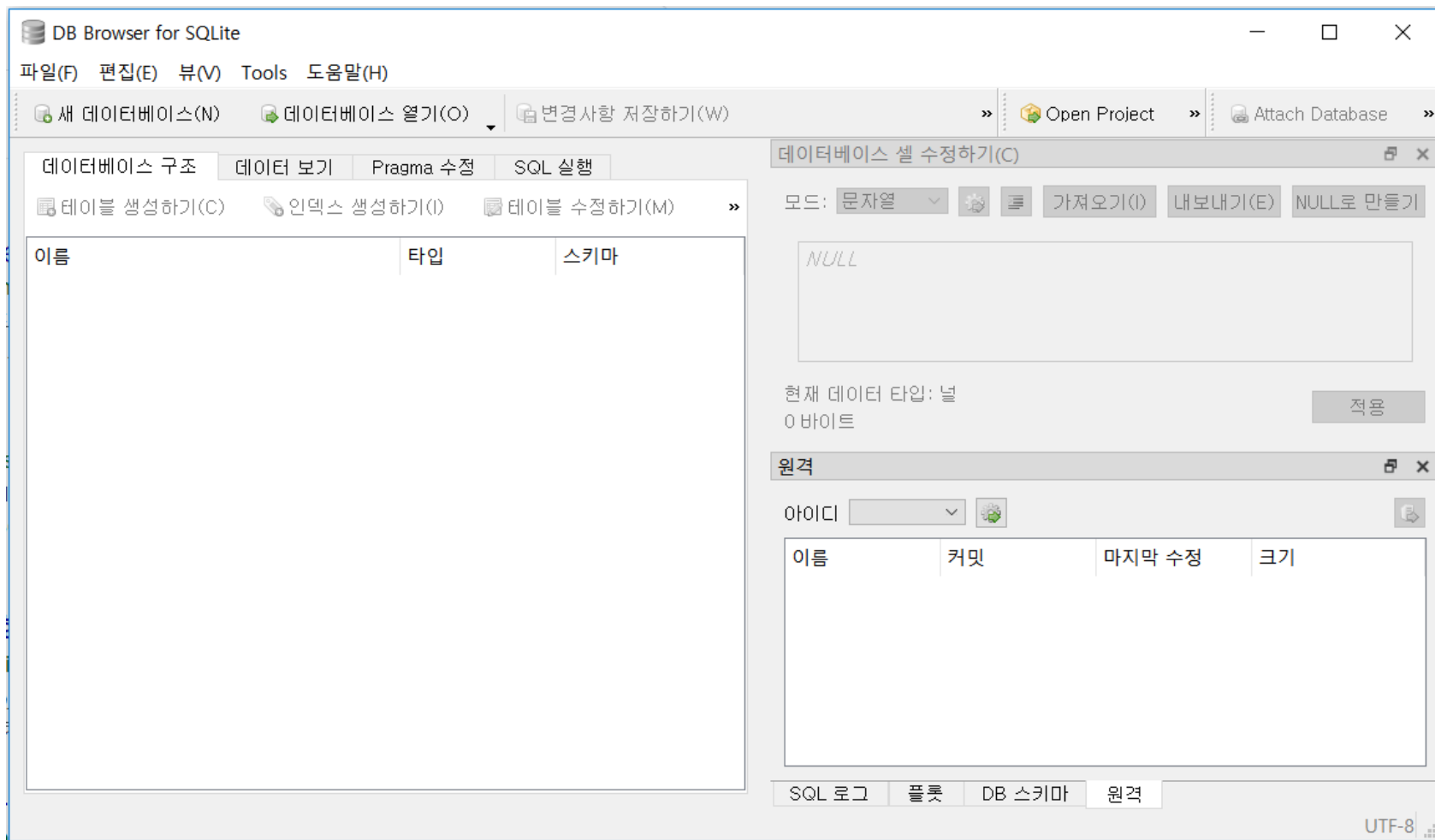
#### ▶ 바탕화면과 프로그램 메뉴에서 실행할 수 있도록 체크



# SQLite 개요

## ▶ 데이터 입력

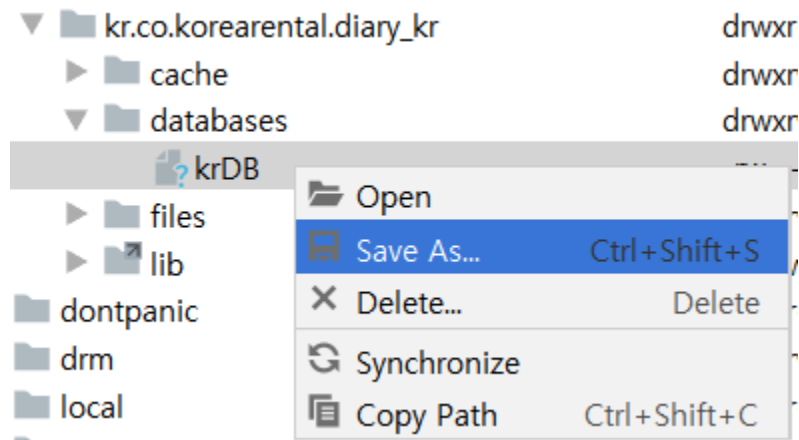
### ▶ SQLite Database Browser 설치 완료



# SQLite 개요

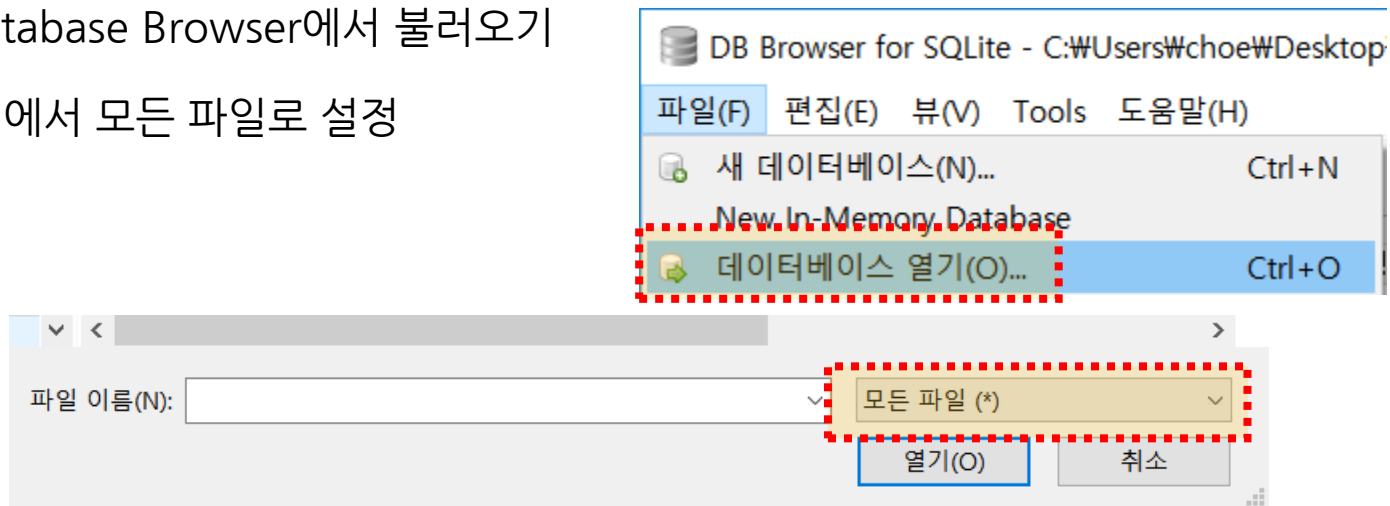
## ▶ 데이터 입력

- ▶ 파일 탐색기에서 데이터베이스 파일을 적당한 위치에 다운로드



- ▶ SQLite Database Browser에서 불러오기

- ▶ 파일열기에서 모든 파일로 설정







# SQLite 개요

## ▶ 데이터 입력

### ▶ 새 레코드 추가 후 데이터 입력

아이디	회원 이름	이메일	출생년도
john	John Bann	john@naver.com	1990
kim	Kim Chi	kim@daum.net	1992
lee	Lee Pal	lee@paran.com	1988
park	Park Su	park@gmail.com	1980

데이터베이스 구조   데이터 보기   Pragma 수정   SQL 실행

테이블(T): memberTable     새 레코드 레코드 삭제

	id	userName	email	birthYear
	필터	필터	필터	필터
1	john	John Bann	john@naver.com	1990
2	kim	Kim Chi	kim@daum.net	1992
3	lee	Lee Pal	lee@paran.com	1988
4	park	Park Su	park@gmail.com	1980

# SQLite 개요

## ▶ 데이터 입력

### ▶ 새 레코드 추가 후 데이터 입력 후 저장

아이디	회원 이름	이메일	출생년도
john	John Bann	john@naver.com	1990
kim	Kim Chi	kim@daum.net	1992
lee	Lee Pal	lee@paran.com	1988
park	Park Su	park@gmail.com	1980

데이터베이스 구조   데이터 보기   Pragma 수정   SQL 실행

테이블(T): memberTable

새 레코드   레코드 삭제

	id	userName	email	birthYear
필터	필터	필터	필터	
1	john	John Bann	john@naver.com	1990
2	kim	Kim Chi	kim@daum.net	1992
3	lee	Lee Pal	lee@paran.com	1988
4	park	Park Su	park@gmail.com	1980

파일(F)   편집(E)   뷰(V)   Tools   도움말(H)

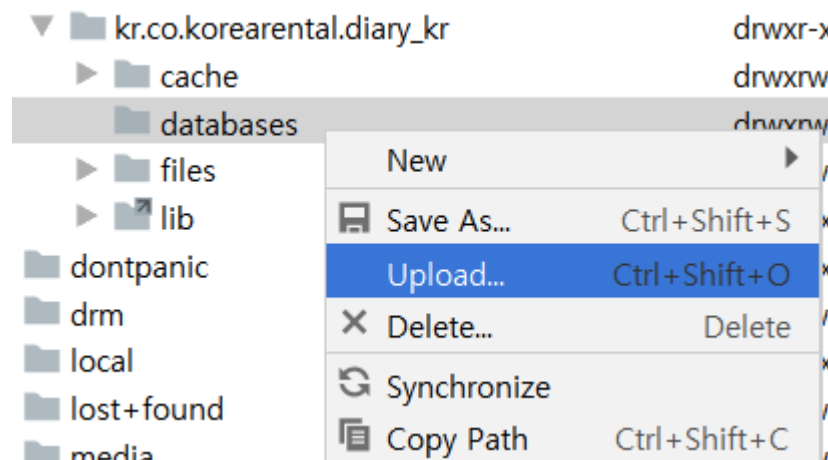
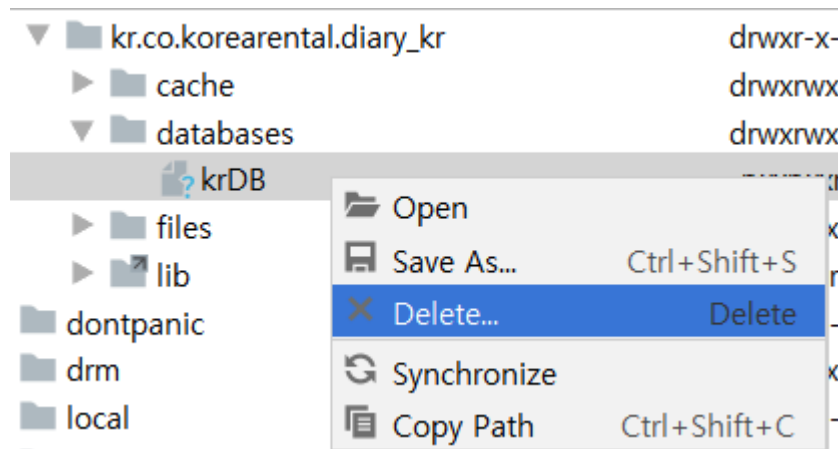
- 새 데이터베이스(N)...   Ctrl+N  
New In-Memory Database
- 데이터베이스 열기(O)...   Ctrl+O  
Open Database Read Only...
- Attach Database...
- 데이터베이스 닫기(C)   Ctrl+W
- 변경사항 저장하기(W)   Ctrl+S**
- 변경사항 취소하기(R)



# SQLite 개요

## ▶ 데이터 입력

▶ 기존의 데이터베이스 파일 삭제 후 레코드가 추가된 파일로 업로드



```
sqlite> .table
memberTable
sqlite> .header on
sqlite> .mode column
sqlite> SELECT * FROM memberTable
...> ;
```

id	userName	email	birthYear
john	John Bann	john@naver.com	1990
kim	Kim Chi	kim@daum.net	1992
lee	Lee Pal	lee@paran.com	1988
park	Park Su	park@gmail.com	1980

# SQLite 개요

## ▶ 쿼리 추가 실습

- ▶ memberTable테이블에서 출생년도가 1990년 이전인 레코드들의 id와 birthyear 정보를 검색

▷ SELECT id, birthYear FROM memberTable WHERE birthYear <= 1990;₩

```
sqlite> SELECT id, birthYear FROM memberTable WHERE birthYear <= 1990;
```

id	birthYear
john	1990
lee	1988
park	1980

- ▶ memberTable테이블에서 id가 park인 레코드의 모든 정보를 검색

▷ SELECT \* FROM memberTable WHERE id = 'park';

```
sqlite> SELECT * FROM memberTable WHERE id = 'park';
```

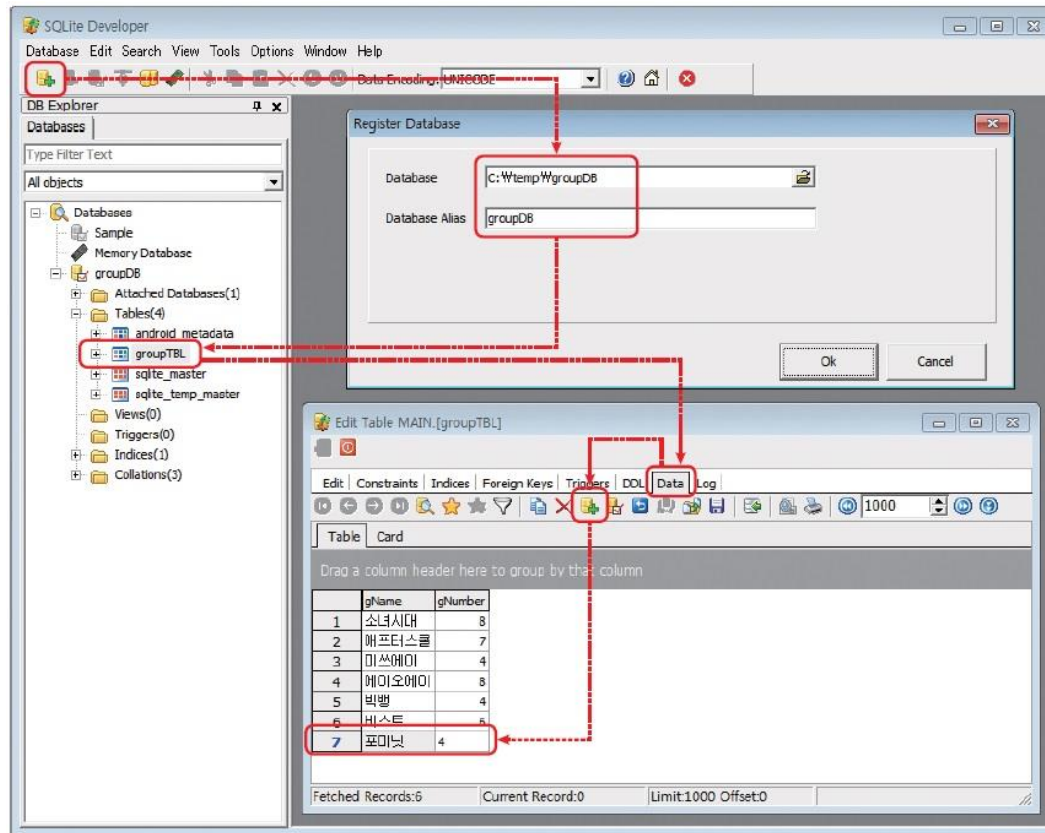
id	userName	email	birthYear
park	Park Su	park@gmail.com	1980

# SQLite 개요

## ▶ 데이터 입력

### ▶ SQLite Developer

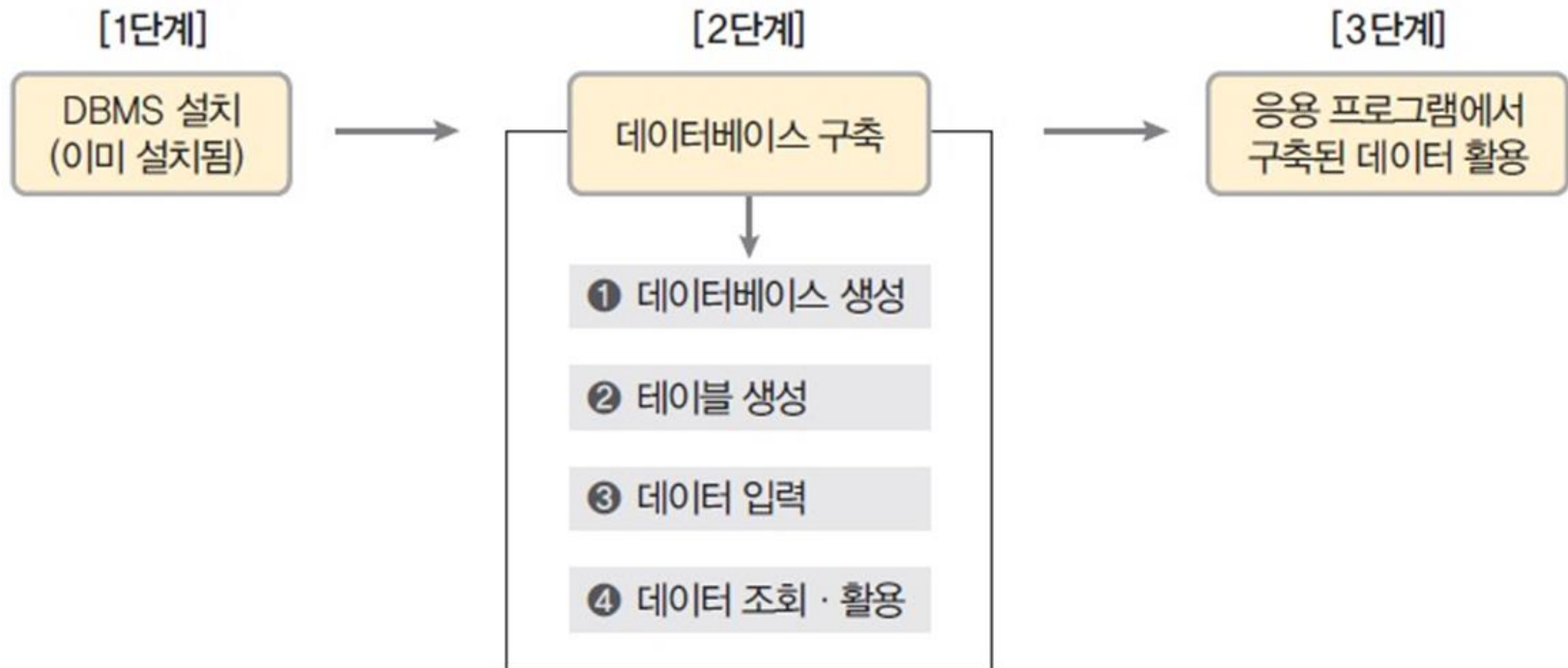
▶ <http://www.sqlitedeveloper.com/download>



# SQLite 개요

## ▶ 데이터 베이스 활용 순서

### ▶ 일반적인 RDBMS와 SQLite에 공통적으로 적용되는 방식



# SQLite를 이용한 간단한 앱 만들기

## ▶ SQLiteDatabase 클래스

- ▶ 데이터를 저장하고, 가져오고 수정/삭제하는 모든 SQL 질의문은 해당 클래스의 함수를 사용
- ▶ 위의 작업을 수행하려면 openOrCreateDatabase()로 SQLiteDatabase 클래스의 객체를 가져와야 함

```
val db = openOrCreateDatabase("memodb", Context.MODE_PRIVATE, null)
```

- ▶ 첫번째 매개변수 : 개발자가 지정하는 데이터베이스 파일명이며 이를 통하여 DB파일을 구분
- ▶ 두번째 매개변수 : 사용 모드로 MODE\_PRIVATE, MODE\_WORLD\_READABLE, MODE\_WORLD\_WRITEABLE중의 하나를 지정 / 두 번째, 세 번째 모드는 다른 애플리케이션에서도 이 데이터베이스 파일에 접근할 수 있도록 함
- ▶ 세번째 매개변수 : CursorFactory는 선택적으로 지정할 수 있으며, NULL이 아닌 객체를 지정할 경우에는 쿼리의 결과값으로 리턴되는 커서를 저장할 객체가 전달됨

# SQLite를 이용한 간단한 앱 만들기

## ▶ SQLiteDatabase 클래스

▶ `openOrCreateDatabase()` 호출하여 리턴되는 `SQLiteDatabase` 객체는 `name` 변수로 지정한 데이터베이스에 접근할 수 있는 메소드를 정의하고 있어서 데이터베이스를 열거나 만든 후에는 이 객체를 참조

▶ `execSQL(sql: String)`

- insert, update 등의 SQL 수행

▶ `rawQuery(sql: String, selectionArgs: Array<String>)`

- select SQL 수행

▶ 데이터베이스에 데이터를 저장하려면 insert문을 사용

```
db.execSQL("insert into tb_memo (title, content) values (?,?)", arrayOf <String>("hello", "world"))
```

▶ 첫번째 매개변수가 SQL문이며 데이터 부분(values)을 ?로 작성했다면 두 번째 매개변수에서 ?에 대응하는 데이터를 지정

▶ 위의 예제는 ?가 두개 이므로 문자열 두개를 가지는 배열을 적용

- 배열의 순서대로 적용

# SQLite를 이용한 간단한 앱 만들기

## ▶ SQLiteDatabase 클래스

▶ 데이터베이스에서 저장된 데이터를 찾아서 가져오려면 select문을 사용

▶ rawQuery()를 사용

```
val cursor = db.rawQuery("select title, content from tb_memo order by _id desc limit 1", null)
```

▶ 위의 메소드도 마찬가지로 첫번째 매개변수가 SQL문이고 두번째 매개변수는 대응하는 데이터

- ?가 없으므로 두번째 인자는 null

▶ rawQuery()는 커서(Cursor) 객체를 리턴

- 커서는 선택된 행(row)의 집합

# SQLite를 이용한 간단한 앱 만들기

## ▶ SQLiteDatabase 클래스

- ▶ 행을 선택하지 않은 상태에서 열 데이터를 추출할 수 없음
- ▶ 커서의 행을 선택하는 함수
  - ▷ moveToNext(): 순서상으로 다음 행 선택
  - ▷ moveToFirst(): 가장 첫 번째 행 선택
  - ▷ moveToLast(): 가장 마지막 행 선택
  - ▷ moveToPrevious(): 순서상으로 이전 행 선택

```
while (cursor.moveToNext()){  
    textView.setText(cursor.getString(0));  
    contentView.setText(cursor.getString(1));  
}
```

- ▷ moveToNext()를 이용하여 행을 선택하고 선택된 행의 getString()함수를 이용하여 열 데이터를 가져옴
  - 0이면 해당 행에서 첫번째 열의 데이터를 가져옴



# SQLite를 이용한 간단한 앱 만들기

## ▶ SQLiteOpenHelper 클래스

### ▶ 데이터베이스 관리를 위한 클래스

▶ 데이터의 저장(insert)이나 획득(select) 등의 작업은 앞서 학습한 SQLiteDatabase 클래스로 작업하고 테이블 생성이나 스키마 변경 등의 작업은 SQLiteOpenHelper 클래스로 구현

▶ SQLiteOpenHelper 클래스는 추상 클래스이므로 서브 클래스를 만들어서 사용

▶ 필수적으로 부모생성자와, onCreate, onUpgrade 이렇게 3가지를 작성해야 함:

```
class DBHelper(context: Context): SQLiteOpenHelper(context, "memodb", null, 1) {  
    override fun onCreate(db: SQLiteDatabase) {  
        //...  
    }  
    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {  
        //...  
    }  
}
```

# SQLite를 이용한 간단한 앱 만들기

## ▶ SQLiteOpenHelper 클래스

```
class DBHelper(context: Context): SQLiteOpenHelper(context, "memodb", null, 1) {  
    override fun onCreate(db: SQLiteDatabase) {  
        //...  
    }  
    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {  
        //...  
    }  
}
```

▶ 생성자 호출 구문(SQLiteOpenHelper(context, "memodb", null, 1))에서

- ▶ 첫 번째 파라미터 : 안드로이드의 Context 정보이며 객체를 사용하는 클래스를 입력
- ▶ 두 번째 파라미터 : 생성할 DB의 이름
- ▶ 세 번째 파라미터 : 데이터 조회 시에 리턴하는 커서를 저장하는 객체
- ▶ 네 번째 파라미터 : 정수 타입의 버전 정보는 데이터베이스 업그레이드를 위해 사용하며 기존에 만들어져 있는 DB의 버전 정보와 다르게 지정하여 DB의 스키마나 데이터를 업데이트함
  - 같은 데이터베이스 명이라도 버전이 틀리면 다른 데이터베이스로 간주

# SQLite를 이용한 간단한 앱 만들기

## ▶ SQLiteOpenHelper 클래스

```
class DBHelper(context: Context): SQLiteOpenHelper(context, "memodb", null, 1) {  
    override fun onCreate(db: SQLiteDatabase) {  
        //...  
    }  
    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {  
        //...  
    }  
}
```

### ▶ onCreate( )

- ▶ DB이름을 생성자에의 인자로 전달했을 때 해당 데이터베이스가 존재하지 않으면 onCreate() 호출되고, 동일한 이름의 데이터베이스가 존재하는 경우에는 호출되지 않음
- ▶ 주로 내부에 table 생성 코드를 작성

### ▶ onUpgrade( )

- ▶ 생성자에 전달된 DB이름의 데이터베이스는 존재하지만 버전이 다른 경우에 호출
- ▶ 주로 새로운 table을 추가하거나 삭제하는 코드를 작성

# SQLite를 이용한 간단한 앱 만들기

## ▶ SQLiteOpenHelper 클래스

```
val helper = DBHelper(this)
val db = helper.writableDatabase
```

- ▶ 객체를 생성하기 위해서는 읽기 전용으로 생성하는 `readableDatabase()`나 쓰기도 가능한 `writableDatabase()`로 생성

## ▶ 생성자 개요

# SQLite를 이용한 간단한 앱 만들기

## ▶ 다양한 함수의 활용

### ▶ SQL문을 수행하기 위하여 다양한 함수를 사용

▷ insert(table: String, nullColumnHack: String, values: ContentValues)

▷ update(table: String, values: ContentValues, whereClause: String, whereArgs: Array<String>)

▷ delete(table: String, whereClause: String, whereArgs: Array<String>)

▷ query(table: String, columns: Array<String>, selection: String, selectionArgs: Array<String>, groupBy: String, having: String, orderBy: String, limit: String)

▶ execSQL()와 rawQuery()는 개발자가 직접 SQL문을 매개변수로 작성하지만 위의 함수는 SQL문을 작성하기 위한 정보만 매개변수로 전달하면 자동으로 SQL문을 생성하여 실행

▶ ContentValues 클래스는 insert, update의 데이터를 표현하는 집합 객체

▷ key - value 형식으로 구성되며 key 값은 테이블에서 열을 의미

▷ nullColumnHack은 추가할 데이터가 null인 경우 대체할 문자열

# SQLite를 이용한 간단한 앱 만들기

## ▶ 다양한 함수의 활용

### ▶ USER\_TB 테이블에 values 객체의 내용을 삽입하는 코드

```
val values = ContentValues()  
values.put("name", "KT_LOG")  
values.put("phone", "0100000")  
db.insert("USER_TB", null, values)
```

### ▶ USER\_TB테이블에서 ID가 KT\_LOG인 사용자를 찾아서 name과 phone 정보를 가져오는 코드

#### ▶ group by, having, order by 조건을 불필요하여 null로 전달

```
val c = db.query("USER_TB", arrayOf ("name", "phone"), "ID=?", arrayOf ("KT_LOG"), null, null, null)
```

# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ 간단한 메모장 앱을 구현

- ▶ 사용자가 입력한 텍스트 정보를 저장하고 불러오는 기능

## ▶ 프로젝트 생성

- ▶ 프로젝트 명 : DBTestApp
- ▶ minSdkVersion : 19(Android 4.4 KitKat)
- ▶ 기본 액티비티 : EmptyActivity

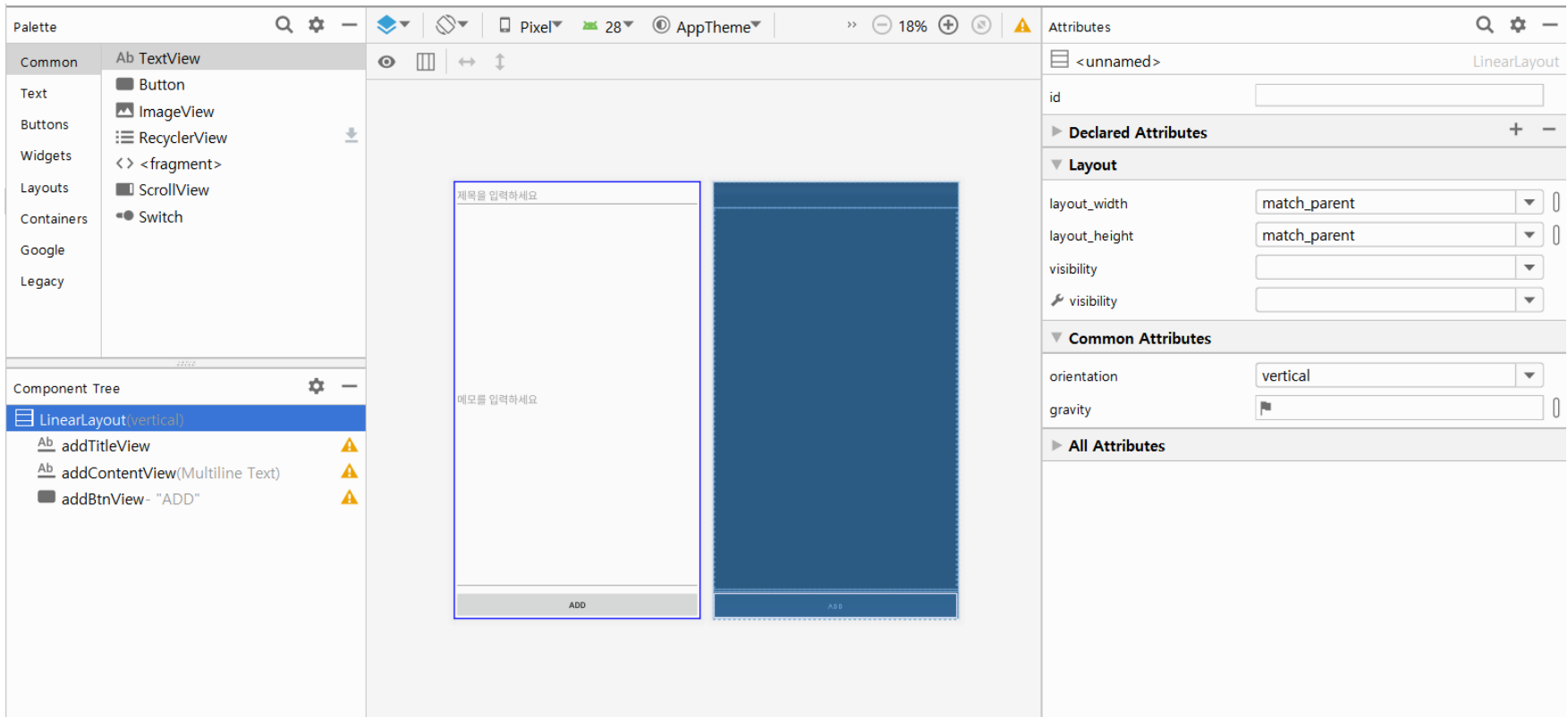
## ▶ MainActivity에서 저장하는 메모를 불러오는 액티비티 추가

- ▶ ReadDBActivity.kt

# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_main.xml 구성

### ▶ 간단하게 LinearLayout 사용

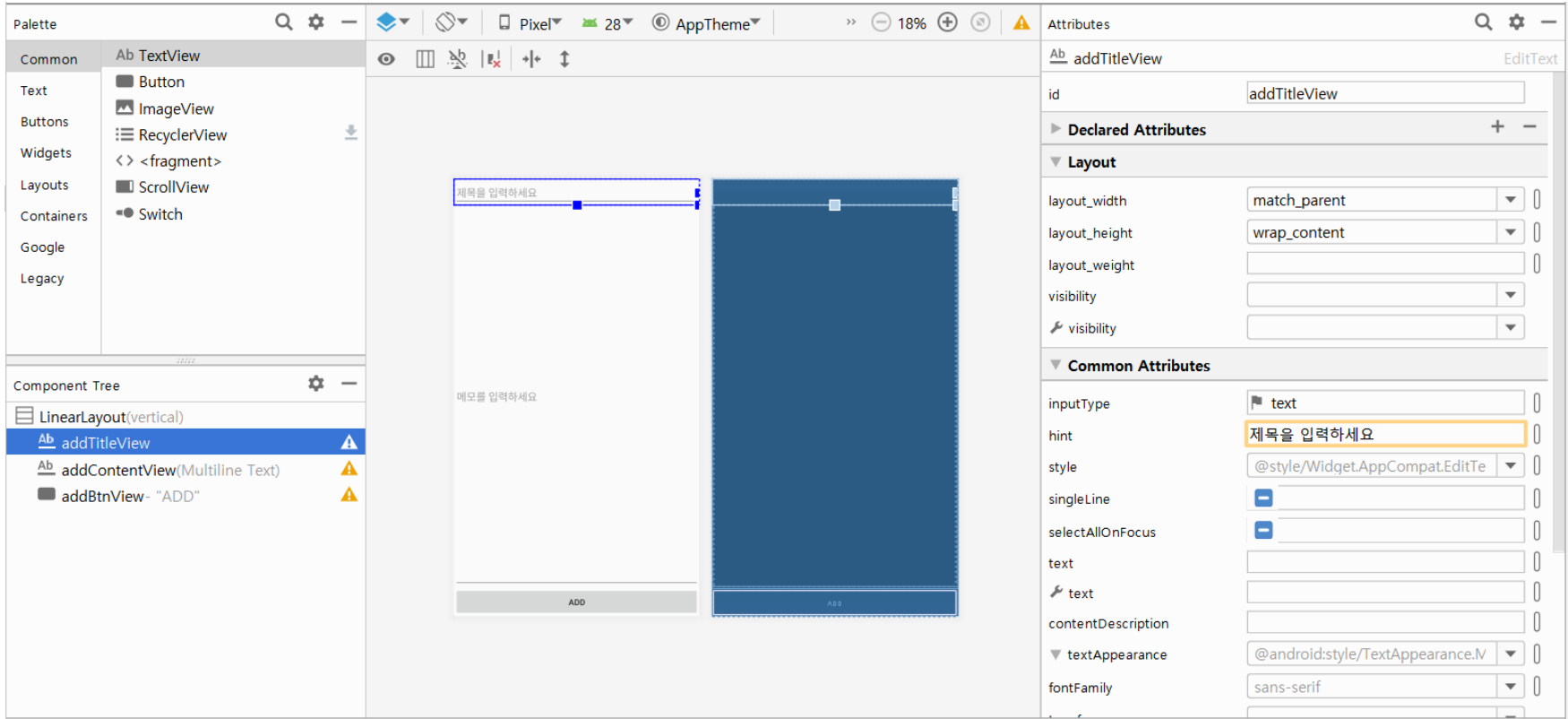




# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_main.xml 구성

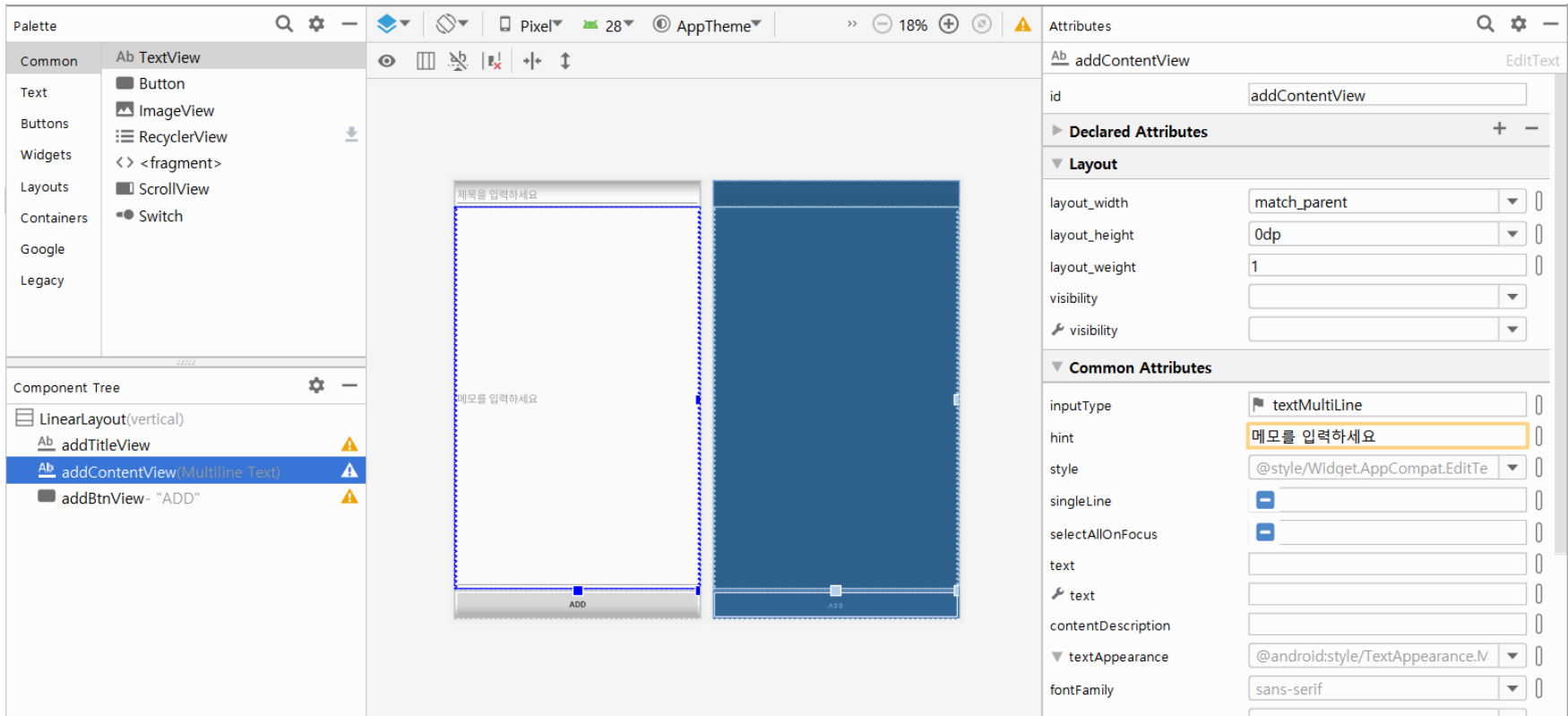
### ▶ 메모의 제목을 입력 받을 에디트 텍스트 추가



# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_main.xml 구성

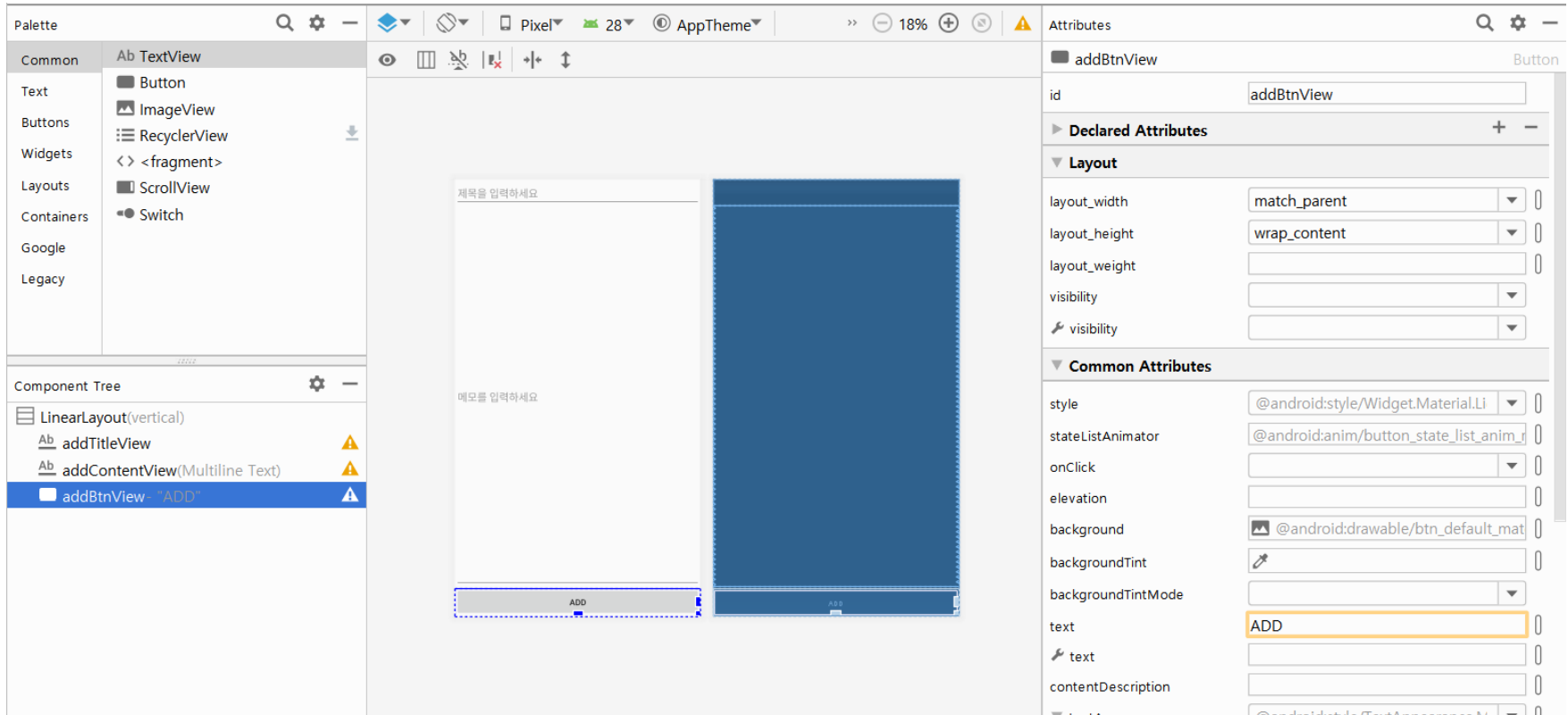
### ▶ 메모 내용을 입력 받을 에디트 텍스트 추가



# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_main.xml 구성

### ▶ 메모가 완성되면 저장할 수 있도록 버튼 추가



# SQLite를 이용한 간단한 메모장 앱 만들기

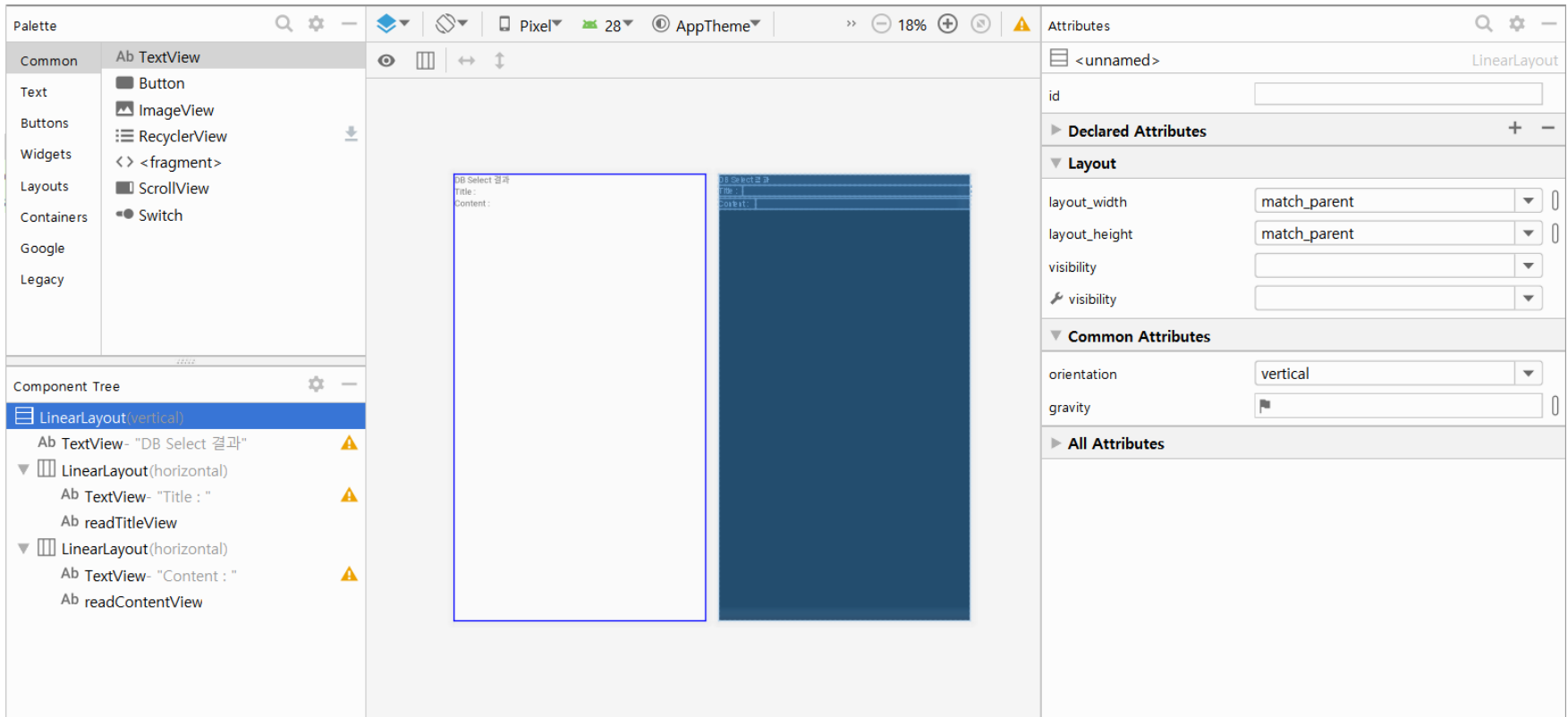
## ▶ activity\_main.xml 코드

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <EditText
        android:id="@+id/addTitleView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="제목을 입력하세요"/>
    <EditText
        android:id="@+id/addContentView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:inputType="textMultiLine"
        android:scrollbars="vertical"
        android:hint="메모를 입력하세요"/>
    <Button
        android:id="@+id/addBtnView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ADD" />
</LinearLayout>
```

# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_read\_db.xml 구성

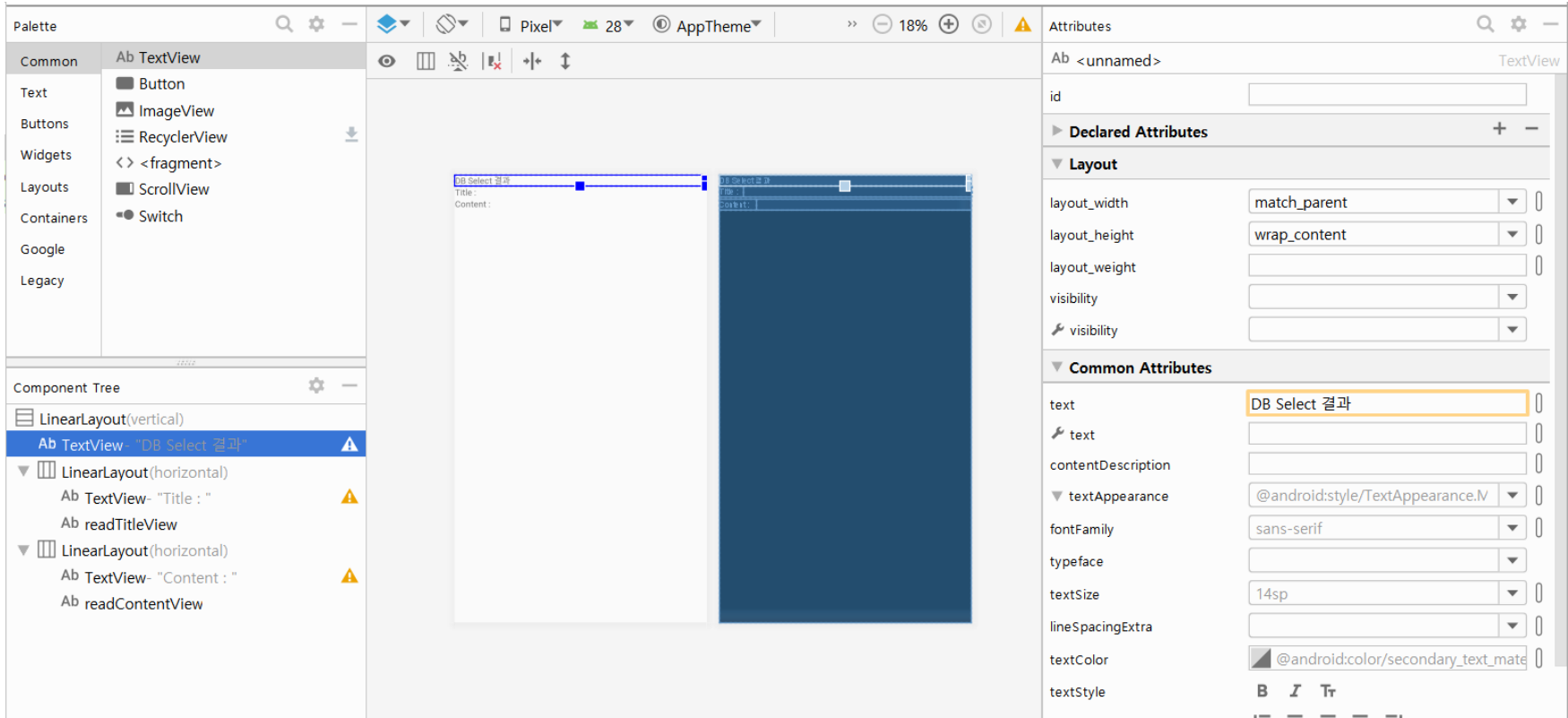
### ▶ 간단하게 LinearLayout 사용



# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_read\_db.xml 구성

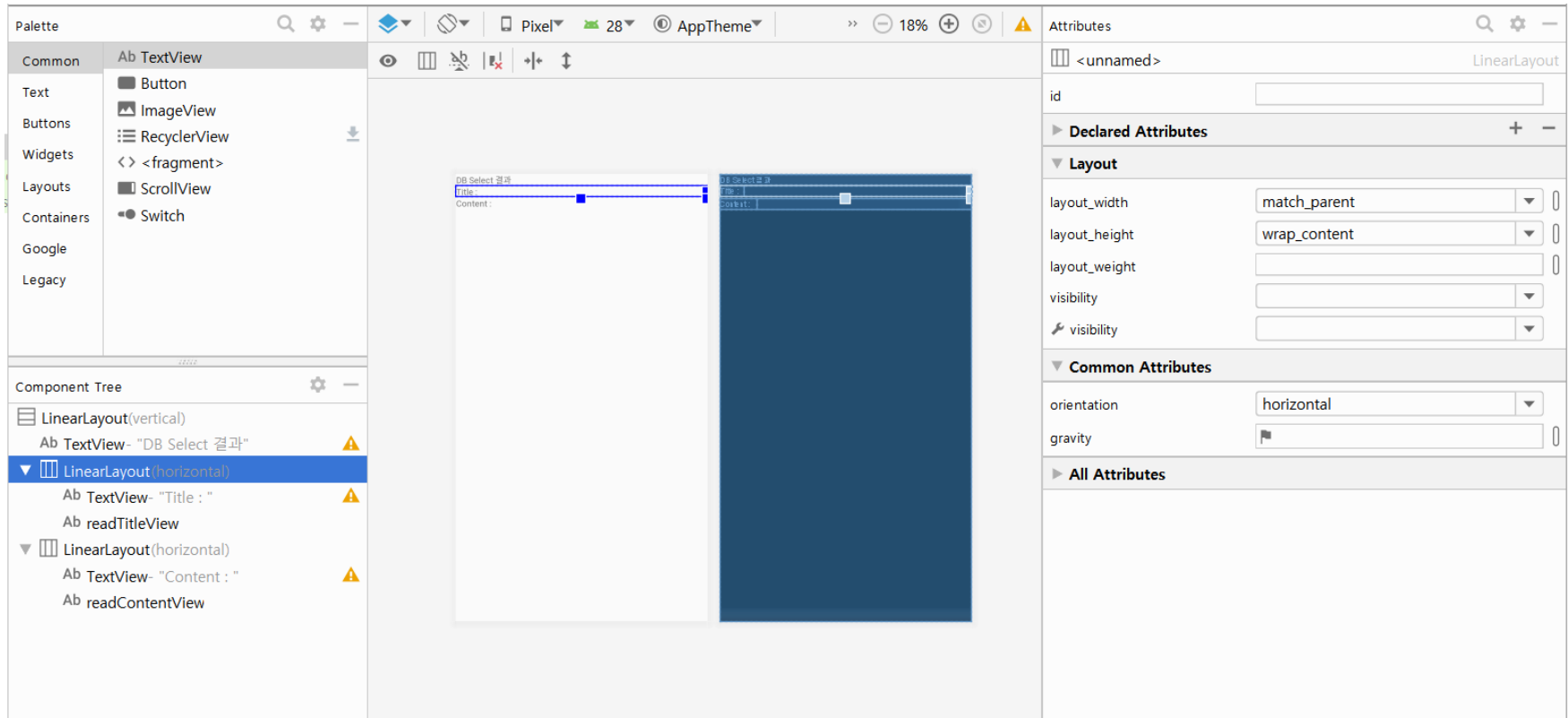
### ▶ 화면 설명이 출력될 텍스트뷰 추가



# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_read\_db.xml 구성

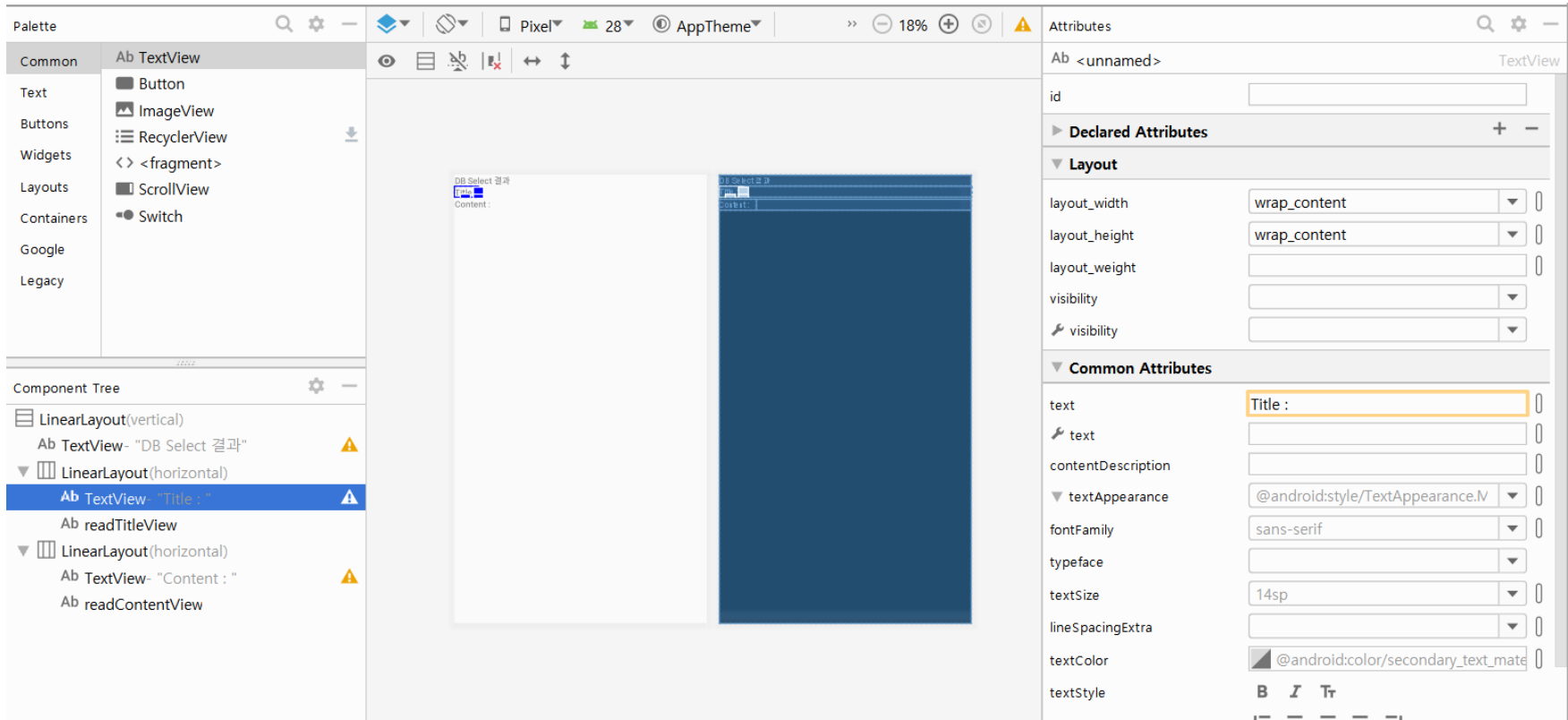
### ▶ 제목이 출력될 부분의 레이아웃 구성



# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_read\_db.xml 구성

### ▶ 제목이 출력될 부분의 레이아웃 구성

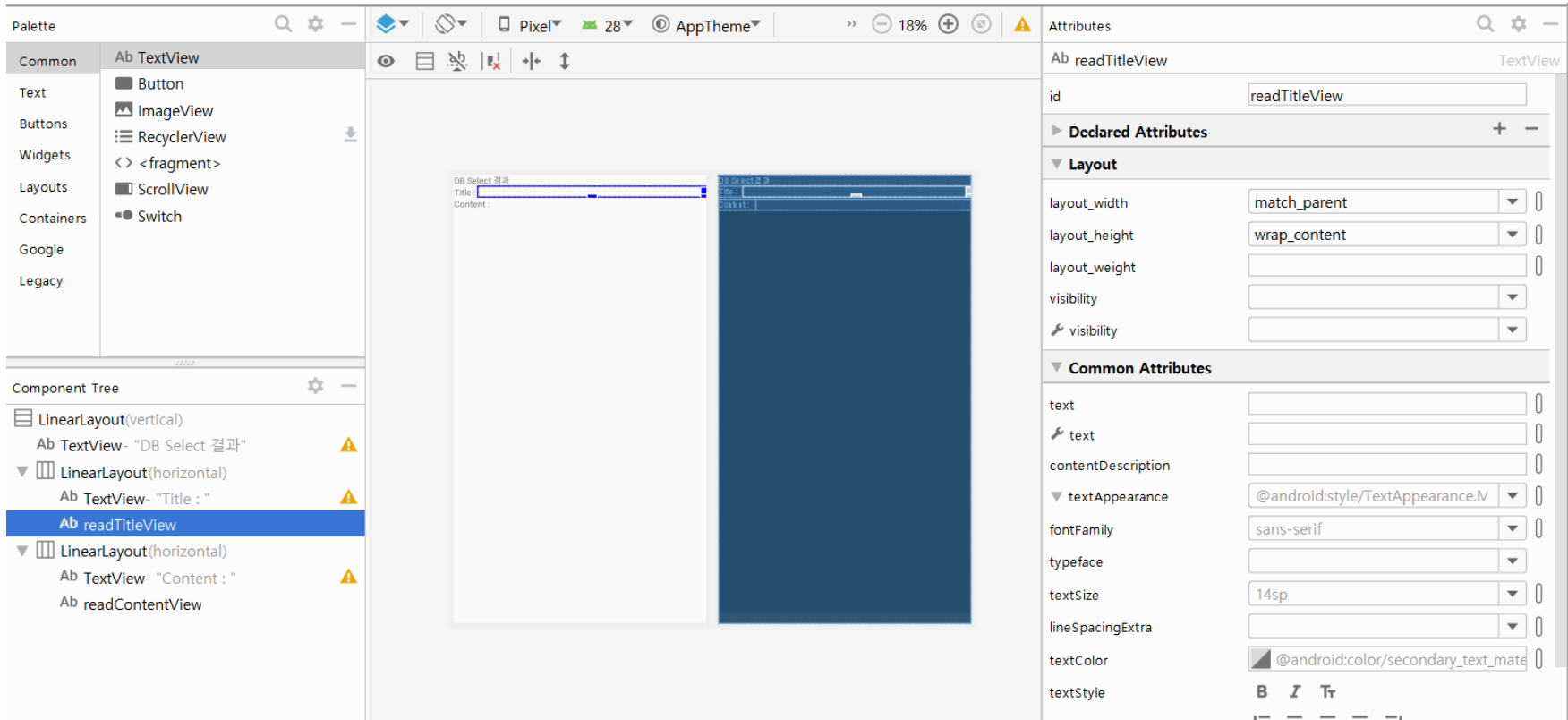




# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_read\_db.xml 구성

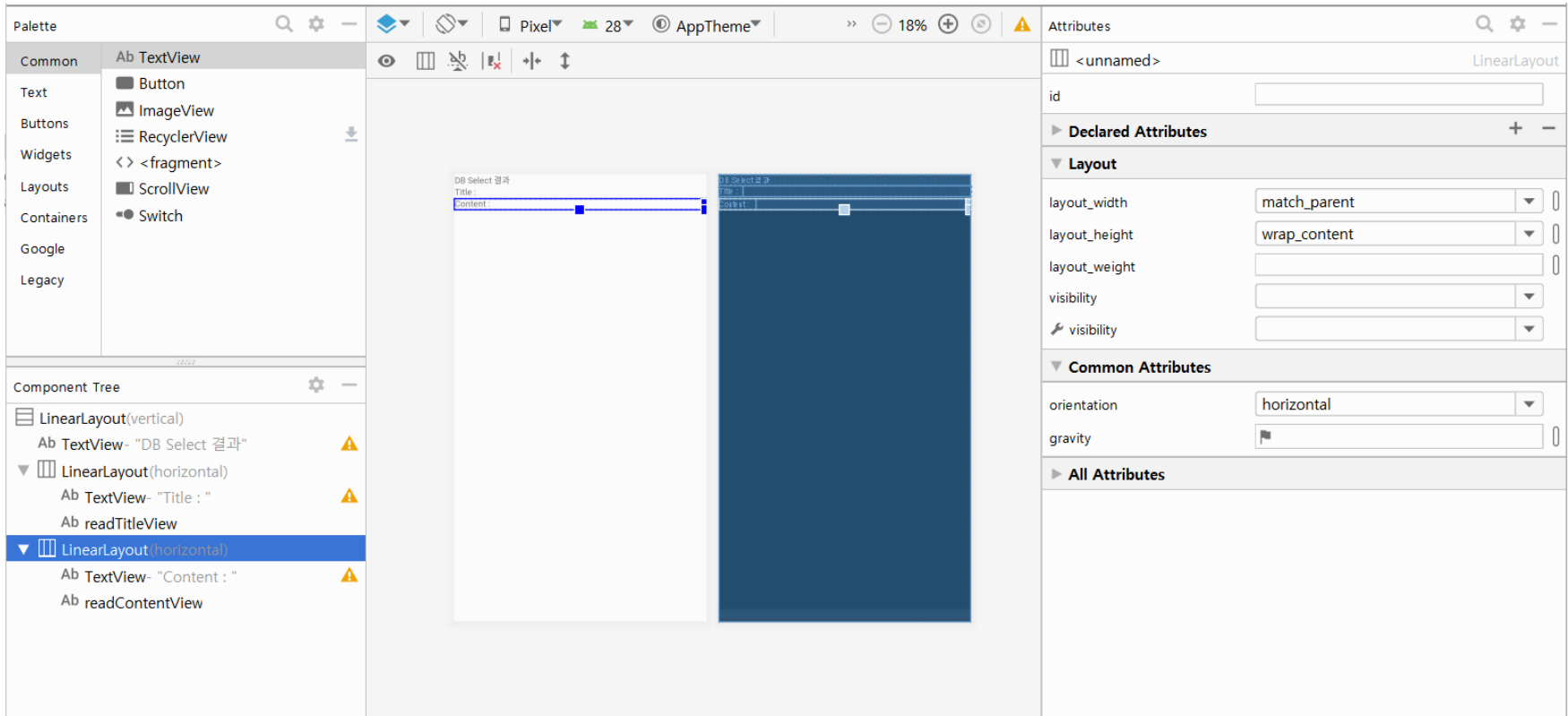
### ▶ 제목이 출력될 부분의 레이아웃 구성



# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_read\_db.xml 구성

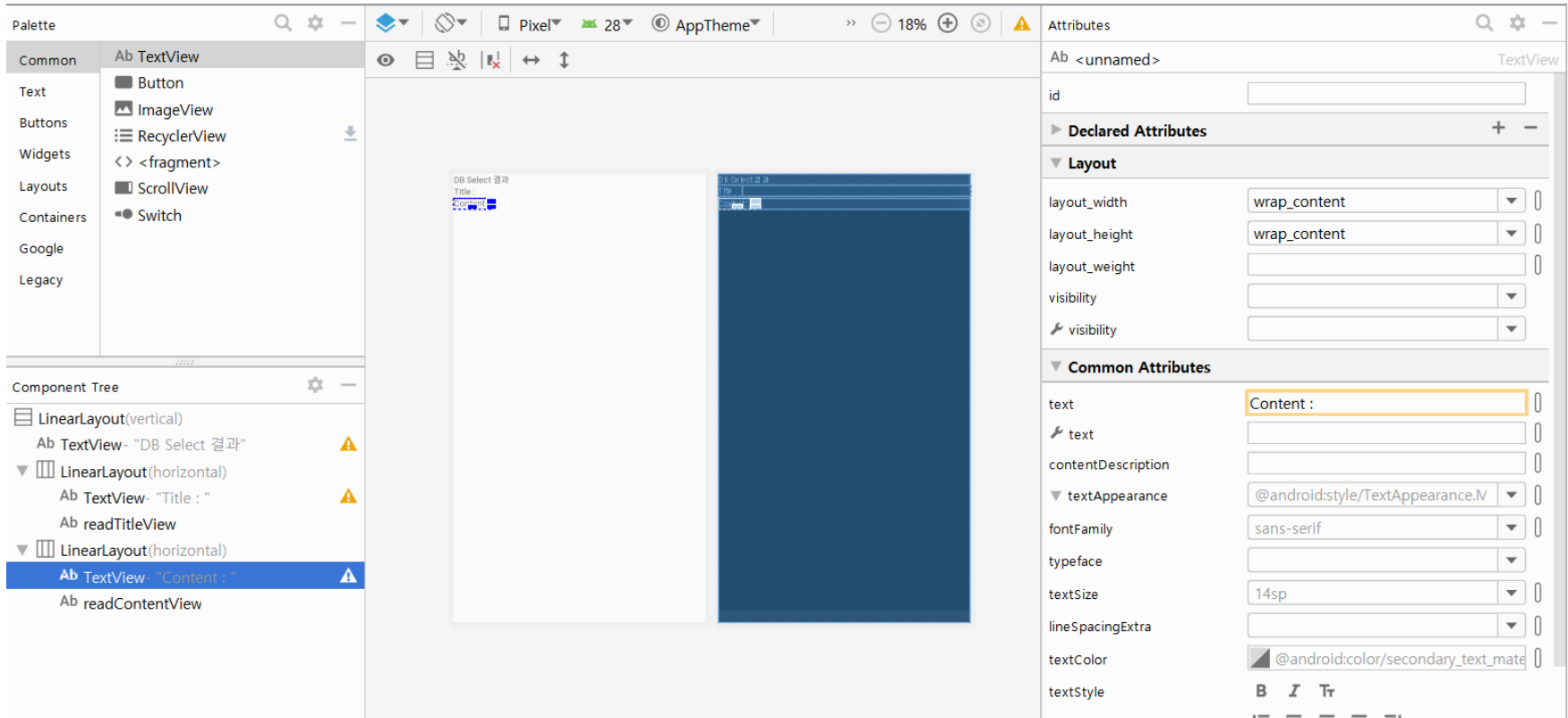
### ▶ 내용이 출력될 부분의 레이아웃 구성



# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_read\_db.xml 구성

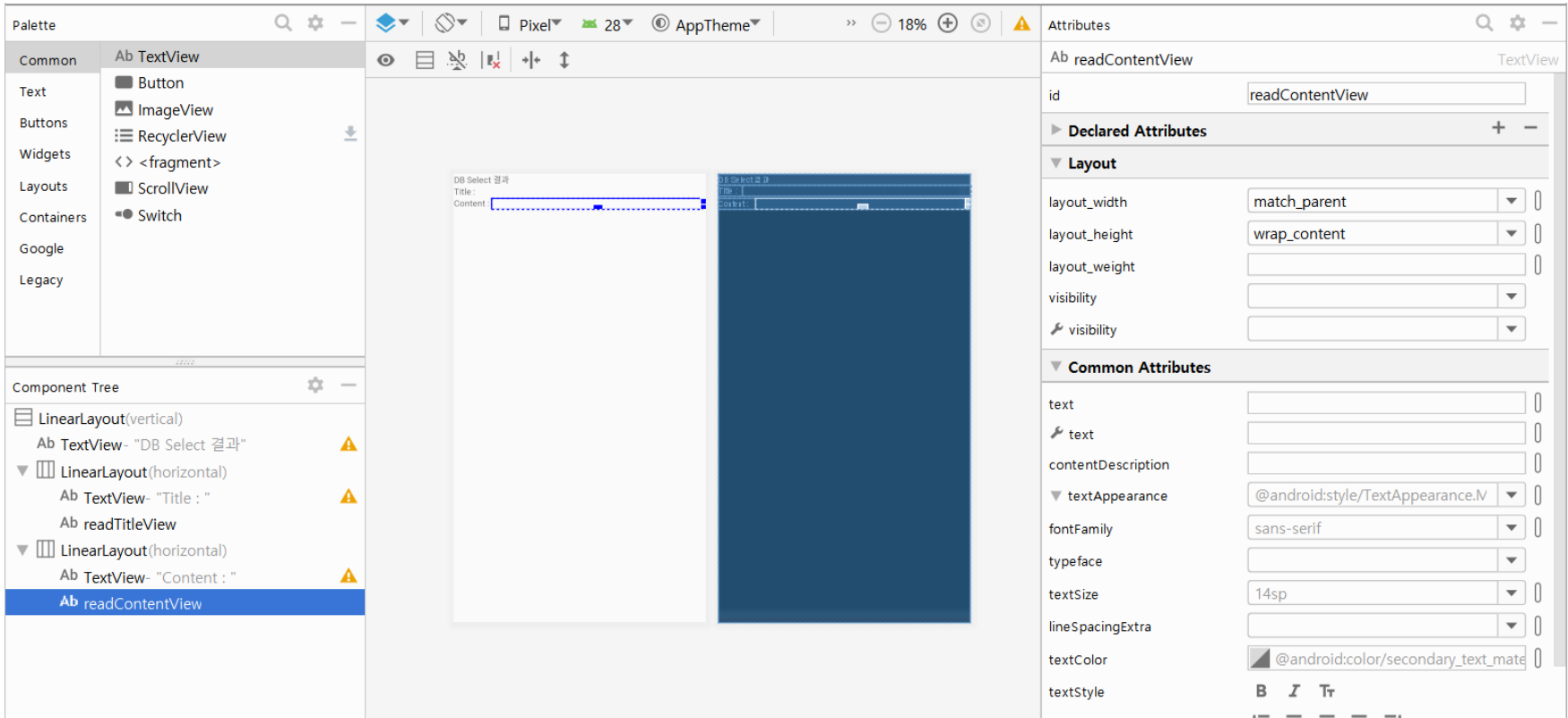
### ▶ 내용이 출력될 부분의 레이아웃 구성



# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_read\_db.xml 구성

### ▶ 내용이 출력될 부분의 레이아웃 구성



# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ activity\_read\_db.xml 코드(1)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="DB Select 결과" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Title : "/>
        <TextView
            android:id="@+id/readTitleView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </LinearLayout>
```

# SQLite를 이용한 간단한 메모장 앱 만들기

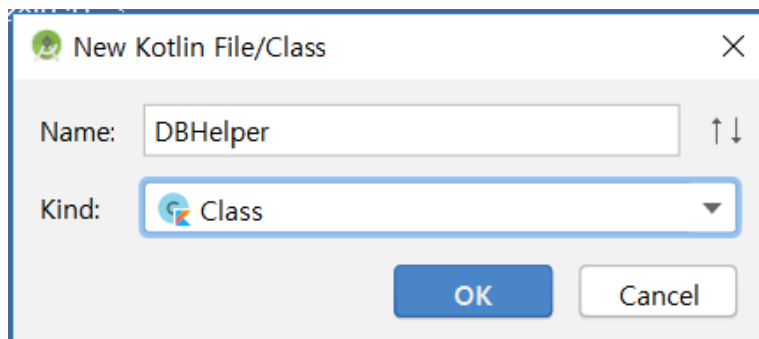
## ▶ activity\_read\_db.xml 코드(2)

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Content : "/>
    <TextView
        android:id="@+id/readContentView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
</LinearLayout>
```

# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ DBHelper 클래스 작성

- ▶ SQLiteOpenHelper 클래스를 상속받는 DBHelper 클래스는 액티비티가 아니므로
- ▶ File - New - kotlin File / Class 로 생성



# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ DBHelper 클래스 작성

▶ tb\_memo 테이블을 생성하고 \_id, title, content 컬럼 생성

▶ id가 레코드를 구분할 수 있는 기본 키이며 레코드를 추가할 때마다 자동증가

```
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DBHelper(context: Context) : SQLiteOpenHelper(context, "memodb", null, 1) {
    override fun onCreate(db: SQLiteDatabase) {
        val memoSQL = "create table tb_memo " +
            "(_id integer primary key autoincrement," +
            "title," +
            "content)"
        db.execSQL(memoSQL)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL("drop table tb_memo")
        onCreate(db)
    }
}
```



# SQLite를 이용한 간단한 메모장 앱 만들기

## ▶ MainActivity 클래스 작성

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        addBtnView.setOnClickListener{  
            val title = addTitleView.getText().toString()  
            val content = addContentView.getText().toString()  
            val helper = DBHelper(this)  
            val db = helper.writableDatabase  
            db.execSQL("insert into tb_memo (title, content) values (?,?)",  
                arrayOf<String>(title, content))  
            db.close()  
            val intent = Intent(this, ReadDBActivity::class.java)  
            startActivity(intent)  
        }  
    }  
}
```

# SQLite를 이용한 간단한 메모장 앱 만들기

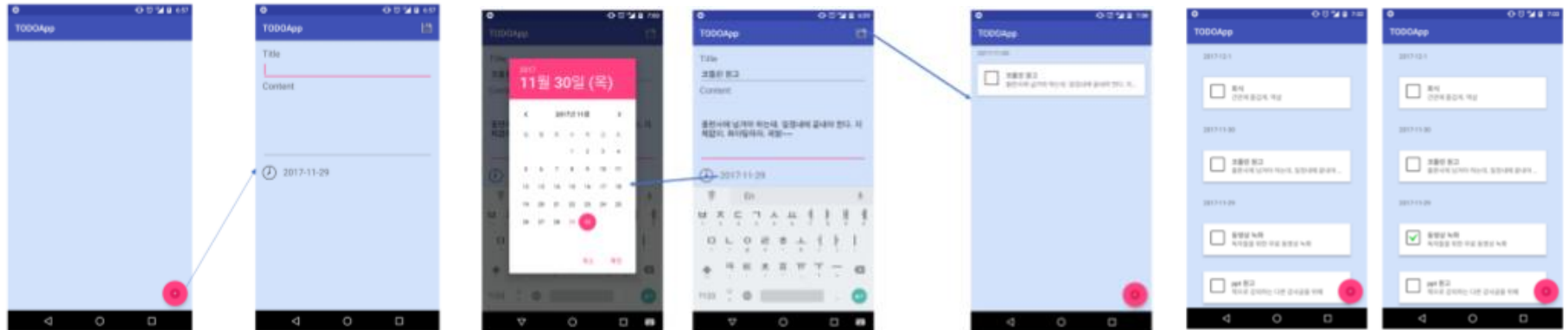
## ▶ ReadDBActivity 클래스 작성

```
class ReadDBActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_read_db)  
  
        val helper = DBHelper(this)  
        val db = helper.writableDatabase  
        val cursor= db.rawQuery("select title, content from tb_memo order by _id desc limit 1", null);  
        while (cursor.moveToNext()){  
            readTitleView.setText(cursor.getString(0));  
            readContentView.setText(cursor.getString(1));  
        }  
        db.close();  
    }  
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ 앱 소개

- ▶ 앱을 실행하면 빈상태의 화면이 나오고 버튼을 클릭하면 업무관련 내용을 입력하는 화면으로 전환
  - ▶ FloatingActionButton 사용
- ▶ 타이틀과 콘텐츠를 에디트텍스트에 입력
- ▶ 아래의 날짜 선택을 클릭하면 다이얼로그를 이용한 캘린더 뷰에서 날짜를 선택
- ▶모두 작성한 후 저장을 하면 내부 DB에 저장되며 MainActivity에 리사이클뷰로 출력
  - ▶ 다양한 업무를 입력했다면 다수의 목록으로 출력됨



# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ 필요 기술

- ▶ UI 프로그래밍
- ▶ RecyclerView
- ▶ FloatingActionButton
- ▶ 사용자 이벤트
- ▶ Menu
- ▶ Intent
- ▶ DBMS

## ▶ DB 구성

테이블명 : TB_TODO	
컬럼명	설명
<b>_id</b>	Primary key, auto increment
<b>Title</b>	Todo 제목
<b>Content</b>	Todo 내용
<b>Date</b>	Todo 날짜
<b>Completed</b>	처리한 내용인지에 대한 식별, 0 – 미처리, 1 – 처리

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ 프로젝트 생성 및 설정

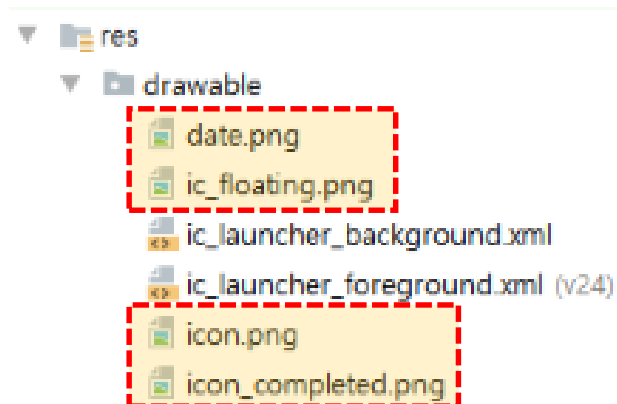
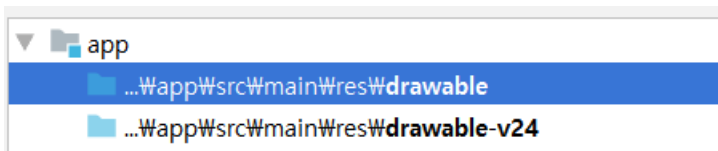
- ▶ 프로젝트명 : DBTodoApp
- ▶ 안드로이드 API 21(android 5.0)버전으로 설정
- ▶ gradle에 의존성 추가
  - ▷ implementation 'com.android.support:recyclerview-v7:26.1.0'
  - ▷ implementation 'com.android.support:design:26.1.0'

## ▶ 액티비티 추가

- ▶ 할 일을 입력하는 액티비티인 AddTodoActivity를 추가

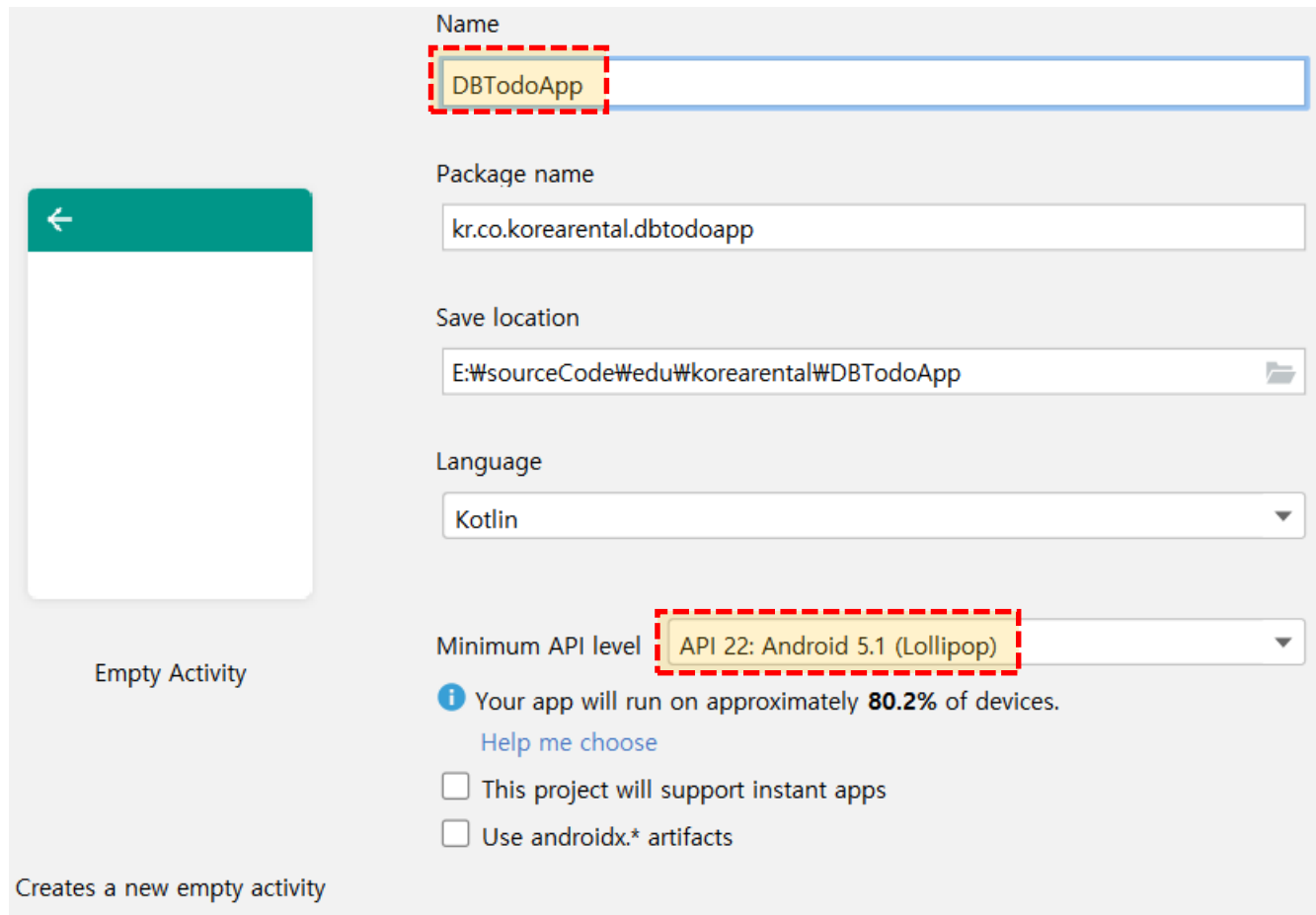
## ▶ 리소스 추가

- ▶ 강의사이트에서 다운로드하여 추가
- ▶ 이미지 4개



# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ 프로젝트 생성 및 설정



Name  
DBToDoApp

Package name  
kr.co.korearental.dbtodoapp

Save location  
E:\sourceCode\Wedu\korearental\DBToDoApp

Language  
Kotlin

Minimum API level  
API 22: Android 5.1 (Lollipop)

**i** Your app will run on approximately **80.2%** of devices.  
[Help me choose](#)

☐ This project will support instant apps

☐ Use androidx.\* artifacts

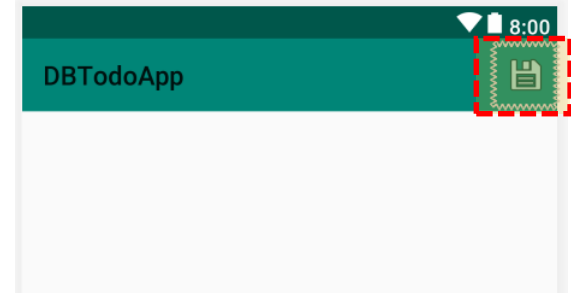
Empty Activity

Creates a new empty activity

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ 메뉴 구성

- ▶ AddToDoActivity에서 사용자가 할 일을 입력한 후 입력한 내용이 DB에 저장하는 버튼 추가
- ▶ 옵션 메뉴로 처리
- ▶ 메뉴는 대부분 리소스화하여 XML로 작성
  - ▷ res - menu에 저장되므로 해당 폴더를 생성
  - ▷ res에서 우 클릭 - new - Android resource directory
  - ▷ resource type을 menu로 설정
    - 디렉터리명은 자동으로 설정됨



### New Resource Directory

Directory name: menu

Resource type: menu

Source set: main



# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ 메뉴 구성

### ▶ 메뉴 폴더에 menu\_add 이름의 xml 파일 추가

New Resource File ×

File name:

Source set:

Directory name:

### ▶ 디자인 구성

▶ `android:icon="@android:drawable/ic_menu_save"` #기본 제공 이미지

▶ `showAsAction="always"` #메뉴 상단 바에 아이콘 표시(ch05웹브라우저만들기 참고)

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
  <item
    android:id="@+id/menu_add"
    android:title="save"
    android:icon="@android:drawable/ic_menu_save"
    app:showAsAction="always"
  />
</menu>
```



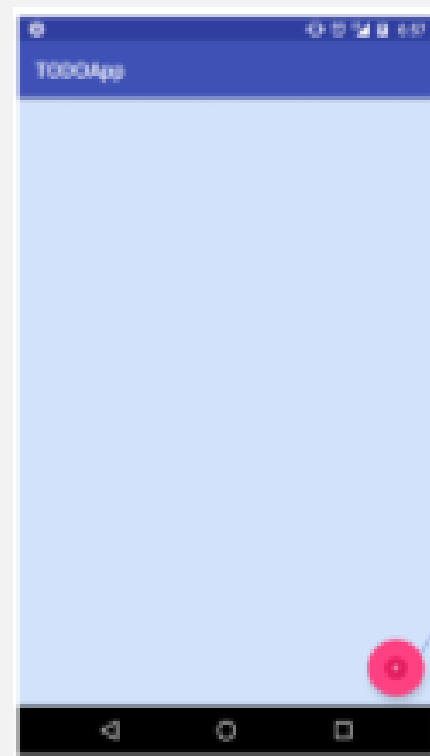
# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ UI 구성

### ▶ activity\_main.xml

#### ▶ RelativeLayout, RecyclerView, FloatingActionButton(ch04스톱워치앱 참고)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#d2e2fb" />
    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:src="@drawable/ic_floating"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_marginBottom="24dp"
        android:layout_marginRight="24dp"
        app:fabSize="normal"
        app:rippleColor="#FFFFFF"
    />
</RelativeLayout>
```



# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ UI 구성

### ▶ activity\_add\_todo.xml(1)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:background="#d2e2fb">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Title"
        android:textSize="20dp"/>

    <EditText
        android:id="@+id/addTitleEditView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"/>
```

Title

Content

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ UI 구성

### ▶ activity\_add\_todo.xml(2)

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Content"
    android:textSize="20dp"/>

<EditText
    android:id="@+id/addContentEditView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:lines="6"/>
```

Title

Content

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ UI 구성

### ▶ activity\_add\_todo.xml(3)

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="16dp">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/date"
        android:maxWidth="30dp"
        android:maxHeight="30dp"
        android:adjustViewBounds="true"/>
    <TextView
        android:id="@+id/addDateView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:textSize="20dp"
        android:clickable="true"
        android:layout_marginLeft="16dp"/>
</LinearLayout>
</LinearLayout>
```

Title

Content

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ UI 구성

- ▶ 리사이클뷰의 항목을 위한 레이아웃 파일 작성
- ▶ 날짜가 나오는 xml 파일을 layout 폴더에 추가

### ▷ item\_header.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/itemHeaderView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ UI 구성

- ▶ 리사이클뷰의 항목을 위한 레이아웃 파일 작성
- ▶ 할일의 제목과 내용이 나오는 항목 추가

### ▶ item\_main.xml(1)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp">

    <ImageView
        android:id="@+id/completedIconView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/icon"
        android:maxLength="32dp"
        android:maxLength="32dp"
        android:adjustViewBounds="true"
        android:clickable="true"/>
```

2019-05-15



ttyt  
ffffg

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ UI 구성

- ▶ 리사이클뷰의 항목을 위한 레이아웃 파일 작성
- ▶ 할일의 제목과 내용이 나오는 항목 추가

### ▷ item\_main.xml

```
<TextView
    android:id="@+id/itemTitleView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="15dp"
    android:textStyle="bold"
    android:layout_marginLeft="16dp"
    android:layout_toRightOf="@id/completedIconView"
    android:ellipsize="end"
    android:maxLines="1"/>

<TextView
    android:id="@+id/itemContentView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/itemTitleView"
    android:layout_alignLeft="@id/itemTitleView"
    android:ellipsize="end"
    android:maxLines="1"
/>
</RelativeLayout>
```

2019-05-15

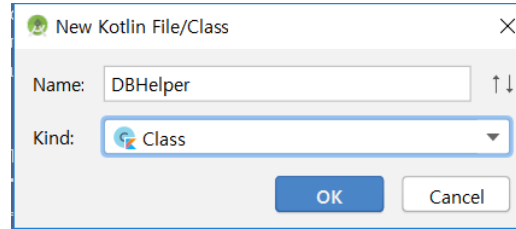


ttyt  
ffffg

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ DBHelper 작성

### ▶ DBHelper 클래스 생성



```
package kr.co.korearental.dbtodoapp

import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DBHelper(context: Context) : SQLiteOpenHelper(context, "tododb", null, 2) {
    override fun onCreate(db: SQLiteDatabase) {
        val memoSQL = "create table tb_todo " +
            "(_id integer primary key autoincrement," +
            "title," +
            "content," +
            "date," +
            "completed)"//0 - none, 1 - completed
        db.execSQL(memoSQL)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL("drop table tb_todo")
        onCreate(db)
    }
}
```



# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ MainActivity 작성(1)

### ▶ 필요한 클래스와 함수를 먼저 선언

```
class MainActivity : AppCompatActivity() {  
  
    var list: MutableList<ItemVO> = mutableListOf()    //DB에서 가져온 데이터 저장  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
    }  
  
    private fun selectDB(){  
        //DB에서 데이터를 가져오는 코드 추가  
    }  
  
    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  
        //할일 입력화면에서 첫화면으로 돌아올때 처리해주는 코드  
    }  
    class HeaderViewHolder(view: View) : RecyclerView.ViewHolder(view) {  
        //날짜 뷰와 데이터를 연결  
    }  
    class DataViewHolder(view: View) : RecyclerView.ViewHolder(view) {  
        //할일 정보 뷰와 데이터를 연결  
    }  
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ MainActivity 작성(2)

```
    inner class MyAdapter(val list: MutableList<ItemVO>) :  
RecyclerView.Adapter<RecyclerView.ViewHolder>() {  
        //리사이클러뷰를 위한 어댑터  
    }  
  
    inner class MyDecoration() : RecyclerView.ItemDecoration() {  
        //아이템의 상세 속성을 설정하는 클래스  
    }  
}  
//할일 정보를 표현하는 클래스  
abstract class ItemVO {  
  
}  
  
class HeaderItem(var date: String) : ItemVO() {  
}  
  
class DataItem(var id: Int, var title: String, var content: String, var completed: Boolean = false) :  
ItemVO() {  
  
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ 항목 데이터 클래스 작성

### ▶ 할일 데이터를 표현하기 위한 클래스

▷ 할일 아이템은 날짜와 할일 내용의 두가지 타입으로 구성

▷ 데이터도 두가지 타입으로 나누어 표현

```
abstract class ItemVO {
    abstract val type: Int
    companion object {
        val TYPE_HEADER = 0
        val TYPE_DATA = 1
    }
}

class HeaderItem(var date: String) : ItemVO() {
    override val type: Int
        get() = ItemVO.TYPE_HEADER
}

class DataItem(var id: Int, var title: String, var content: String, var completed: Boolean = false) :
    ItemVO() {
    override val type: Int
        get() = ItemVO.TYPE_DATA
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ 항목 데이터 구성

```
private fun selectDB(){
    list = mutableListOf()
    val helper = DBHelper(this)
    val db = helper.readableDatabase
    val cursor = db.rawQuery("select * from tb_todo order by date desc", null)

    var preDate: Calendar? = null
    while (cursor.moveToNext()) {
        val dbdate=cursor.getString(3)
        val date = SimpleDateFormat("yyyy-MM-dd").parse(dbdate)
        val currentDate = GregorianCalendar()
        currentDate.time = date

        if(!currentDate.equals(preDate)) {
            val headerItem = HeaderItem(dbdate)
            list.add(headerItem)
            preDate=currentDate
        }

        val completed= cursor.getInt(4) != 0
        val dataItem = DataItem(cursor.getInt(0), cursor.getString(1), cursor.getString(2),
completed)
        list.add(dataItem)
    }

    Log.d("KT_LOG","list size ${list.size}")
    recyclerView.layoutManager = LinearLayoutManager(this)
    recyclerView.adapter = MyAdapter(list)
    recyclerView.addItemDecoration(MyDecoration())
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ 뷰홀더 작성

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    if(requestCode==10 && resultCode== Activity.RESULT_OK){
        selectDB()
    }
}

class HeaderViewHolder(view: View) : RecyclerView.ViewHolder(view) {
    val headerView = view.itemHeaderView
}

class DataViewHolder(view: View) : RecyclerView.ViewHolder(view) {
    val completedIconView = view.completedIconView
    val itemTitleView = view.itemTitleView
    val itemContentView = view.itemContentView
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ adapter 작성

```
inner class MyAdapter(val list: MutableList<ItemVO>) : RecyclerView.Adapter<RecyclerView.ViewHolder>()
{
    override fun getItemViewType(position: Int): Int {
        return list.get(position).type
    }

    override fun onCreateViewHolder(parent: ViewGroup?, viewType: Int): RecyclerView.ViewHolder {
        if (viewType == ItemVO.TYPE_HEADER) {
            //parent?.context : parent가 not null이면 context가 리턴되고 null이면 null이 리턴된다.
            val inflater = LayoutInflater.from(parent?.context)
            return HeaderViewHolder(inflater.inflate(R.layout.item_header, parent, false))
        } else {
            val inflater = LayoutInflater.from(parent?.context)
            return DataViewHolder(inflater.inflate(R.layout.item_main, parent, false))
        }
    }
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ adapter 작성

```
//각 항목을 구성하는 메소드
//날짜나 할일에 따라 다르게 처리
override fun onBindViewHolder(holder: RecyclerView.ViewHolder?, position: Int) {
    val itemVO = list.get(position)
    if (itemVO.type == ItemVO.TYPE_HEADER) {
        //as .. type casting
        val viewHolder = holder as HeaderViewHolder
        val headerItem = itemVO as HeaderItem
        viewHolder.headerView.setText(headerItem.date)
    } else {
        val viewHolder = holder as DataViewHolder
        val dataItem = itemVO as DataItem
        viewHolder.itemTitleView.setText(dataItem.title)
        viewHolder.itemContentView.setText(dataItem.content)

        if(dataItem.completed){
            viewHolder.completedIconView.setImageResource(R.drawable.icon_completed)
        }else {
            viewHolder.completedIconView.setImageResource(R.drawable.icon)
        }
    }
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ adapter 작성

```
viewHolder.completedIconView.setOnClickListener{
    val helper = DBHelper(this@MainActivity)
    //inner 라는 예약어가 클래스에 추가되어 있어야 한다.
    val db=helper.writableDatabase

    if(dataItem.completed){
        db.execSQL("update tb_todo set completed=? where _id=?", arrayOf(0,
dataItem.id))
        viewHolder.completedIconView.setImageResource(R.drawable.icon)
    }else {
        db.execSQL("update tb_todo set completed=? where _id=?", arrayOf(1,
dataItem.id))
        viewHolder.completedIconView.setImageResource(R.drawable.icon_completed)
    }
    dataItem.completed = !dataItem.completed
    db.close()
}
}

override fun getItemCount(): Int {
    return list.size
}
}
```



# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ itemDecoration 작성

```
inner class MyDecoration() : RecyclerView.ItemDecoration() {
    override fun getItemOffsets(outRect: Rect?, view: View?, parent: RecyclerView?, state:
RecyclerView.State?) {
        super.getItemOffsets(outRect, view, parent, state)
        val index = parent!!.getChildAdapterPosition(view)
        Log.d("KT_LOG", "index $index.... list size: ${list.size}")
        val itemVO = list.get(index)
        if (itemVO.type == ItemVO.TYPE_DATA) {
            view!!.setBackgroundColor(0xFFFFFFFF.toInt())
            ViewCompat.setElevation(view, 10.0f)
        }
        outRect!!.set(20, 10, 20, 10)
    }
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ onCreate()

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    selectDB()  
  
    fab.setOnClickListener{  
        val intent = Intent(this, AddTodoActivity::class.java)  
        startActivityForResult(intent, 10)  
    }  
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ AddTodoActivity 작성

```
class AddTodoActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_add_todo)  
  
        val date = Date()  
        val sdfFormat = SimpleDateFormat("yyyy-MM-dd")  
        addDateView.text=sdfFormat.format(date)  
  
        addDateView.setOnClickListener{  
            val c = Calendar.getInstance()  
            val year = c.get(Calendar.YEAR)  
            val month = c.get(Calendar.MONTH)  
            val day = c.get(Calendar.DAY_OF_MONTH)  
  
            val dateDialog = DatePickerDialog(this, object : DatePickerDialog.OnDateSetListener {  
                override fun onDateSet(view: DatePicker, year: Int, monthOfYear: Int, dayOfMonth: Int)  
                {  
                    addDateView.text = "$year-${monthOfYear+1}-${dayOfMonth}"  
                }  
            }, year, month, day).show()  
        }  
  
    }  
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

## ▶ AddTodoActivity 작성

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.menu_add, menu)
    return super.onCreateOptionsMenu(menu)
}
override fun onOptionsItemSelected(item: MenuItem?): Boolean {
    if(item?.itemId==R.id.menu_add){
        if(addTitleEditText.text.toString() != null && addContentEditText.text.toString() != null)
        {
            val helper = DBHelper(this)
            val db = helper.writableDatabase

            val contentValues=ContentValues()
            contentValues.put("title", addTitleEditText.text.toString())
            contentValues.put("content", addContentEditText.text.toString())
            contentValues.put("date", addDateView.text.toString());
            contentValues.put("completed", 0)

            db.insert("tb_todo", null, contentValues)

            db.close()

            setResult(Activity.RESULT_OK)

            finish()
        }else {
            Toast.makeText(this, "모든 데이터가 입력되지 않았습니다.", Toast.LENGTH_SHORT).show()
        }
    }
    return super.onOptionsItemSelected(item)
}
```

# SQLite와 RecyclerView를 이용한 Todo 리스트 앱

---

▶ 결과 확인

# MSSQL과 연동하는 앱

## ▶ 앱 설명

- ▶ 외부 데이터베이스와 연결하는 앱을 작성
  - ▶ 본 실습에서 사용할 DBMS는 MSSQL
- ▶ 본 실습을 위해 세팅된 MSSQL은 서버로 동작하여 안드로이드 앱을 이용해 원격으로 데이터를 가져옴
- ▶ DBMS에서 데이터를 가져와 안드로이드 앱의 리스트뷰에 출력



서버



데이터  
가져오기



클라이언트

# MSSQL과 연동하는 앱

## ▶ 네트워크 접속 권한 부여하기

- ▶ 서버를 접속하는 것은 네트워크를 통해 통신하는 행위이기 때문에 앱에 네트워크 접속권한을 부여해 주어야함
- ▶ AndroidManifest.xml 을 수정하여 접속권한 부여가능
  - ▶ `<uses-permission android:name="android.permission.INTERNET" />`

# MSSQL과 연동하는 앱

## ▶ 메니페스트 권한

### ▶ 해당 앱 실행 시 인터넷 접속 허용

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

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="kr.co.korearental.mssqltestapp">
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        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>
```



# MSSQL과 연동하는 앱

## ▶ JDBC 라이브러리 추가

▶ 프로젝트에 MSSQL 서버에 접속할 수 있도록 JDBC 라이브러리인 JTDS를 추가

▷ implementation 'net.sourceforge.jtds:jtds:1.3.1'

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"  
    implementation 'com.android.support:appcompat-v7:28.0.0'  
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.2'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
    implementation 'net.sourceforge.jtds:jtds:1.3.1'  
}
```

# MSSQL과 연동하는 앱

## ▶ 코드작성

- ▶ 안드로이드 어플리케이션은 네트워크에 접속을 하기 위해선 주 Thread를 사용하면 안된다.
- ▶ MSSQL 서버에 접속하는 것도 네트워크에 접속하는 행위이기 때문에 AsyncTask 클래스를 사용하여 접속
- ▶ AsyncTask의 기본적인 형태, MainActivity 클래스 내부에 작성할 것
- ▶ doInBackground()가 호출되어 실행이 완료되면 자동으로 onPostExecute()가 실행

```
inner class MyAsyncTask : AsyncTask<String, Void, ArrayList<String>>() {  
  
    override fun doInBackground(vararg params: String): ArrayList<String> {  
        //DBMS에 접속을 하기위한 코드 작성  
    }  
  
    override fun onPostExecute(list: ArrayList<String>) {  
        // doInBackground 에서의 작업이 끝나면 실행, 이곳에서 데이터를 리스트에 저장  
    }  
}
```

# MSSQL과 연동하는 앱

## ▶ MSSQL 서버의 정보

### ▶ DB 서버 정보

▷ DATA SOURCE = 115.71.52.30:14336

▷ USER ID = icsmanager

▷ PASSWORD = gksrnrfpsxkfanstnwls

▷ DATABASE = HomeWork

▷ Table명 = MeasData

### ▶ HomeWork.MeasData 테이블 정보

▷ TestDate

▷ 통신사

▷ Frequency

▷ MeasuredValue

# MSSQL과 연동하는 앱

## ▶ MSSQL 서버의 정보

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The left pane shows the server hierarchy for '115.71.52.30,14336 (SQL Server 10.50.1600.1 - icsmanager)'. The central pane shows the query results for the query 'select \* from HomeWork.MeasData'. The right pane shows the '속성' (Properties) window for the selected query.

**Query Results:**

	TestDate	통신사	Frequency	MeasuredValue
1	2019-02-27 05:57:35.000	SKT	1800MHz	14
2	2019-02-27 05:57:37.000	SKT	1800MHz	47
3	2019-02-27 05:57:38.000	SKT	1800MHz	19
4	2019-02-27 05:57:39.000	SKT	1800MHz	47
5	2019-02-27 05:57:40.000	SKT	1800MHz	21
6	2019-02-27 11:46:04.000	KT	1800MHz	22
7	2019-02-27 11:46:07.000	KT	1800MHz	24
8	2019-02-27 11:46:10.000	KT	1800MHz	23
9	2019-02-27 11:46:13.000	KT	1800MHz	47
10	2019-02-27 11:46:16.000	KT	1800MHz	8
11	2019-02-27 11:46:18.000	KT	1800MHz	40
12	2019-02-27 11:46:20.000	KT	1800MHz	4
13	2019-02-27 11:46:22.000	KT	1800MHz	12
14	2019-02-27 11:46:24.000	KT	1800MHz	34
15	2019-02-28 05:10:33.000	SKT	800M	1
16	2019-02-28 05:13:53.000	KT	2600M	1
17	2019-02-28 05:13:55.000	KT	2600M	2
18	2019-02-28 05:13:56.000	KT	2600M	3
19	2019-02-28 05:13:57.000	KT	2600M	4
20	2019-02-28 05:13:58.000	KT	2600M	5
21	2019-02-28 05:58:38.000	SKT	1800M	5
22	2019-02-28 05:58:40.000	SKT	1800M	4
23	2019-02-28 05:58:42.000	SKT	1800M	9
24	2019-02-28 05:59:01.000	SKT	1800M	8
25	2019-03-04 05:30:33.000	SKT	900M	55
26	2019-03-04 10:32:50.000	SKT	2100M	2
27	2019-03-04 12:01:31.000	KT	1800MHz	38

**Server Properties (속성):**

- 현재 연결 매개 변수: 현재 연결
- 연결 이름: 115.71.52.30,14
- 연결 정보: Azure Active, SPID: 56, 로그인 이름: icsmanager, 반환된 연결: 166, 서버 버전: 10.50.1600, 서버 이름: 115.71.52.30,14, 세션 추적 ID: 연결 경과 시: 00:00:00.351, 연결 상태: 열림, 연결 시작 시: 2019-05-07 오전, 연결 암호화: 암호화되지 않음, 연결 완료 시: 2019-05-07 오전, 표시 이름: 115.71.52.30,14
- 집계 상태: 경과된 시간: 00:00:00.351, 반환된 행 수: 166, 상태: 열림, 시작 시간: 2019-05-07 오전, 연결 실패: 완료 시간: 2019-05-07 오전, 이름: 115.71.52.30,14, 연결 이름입니다.

준비 | 115.71.52.30,14336(10.50 RTM) | icsmanager (56) | HomeWork | 00:00:00 | 166개의 행

# MSSQL과 연동하는 앱

## ▶ 코드작성

### ▶ MainActivity의 멤버변수 작성

```
private var mTask: MyAsyncTask? = null //Background에서 DB접속을 위한 MyAsyncTask 변수
private var listView: ListView? = null //데이터를 출력할 ListView 변수
private var adapter: ArrayAdapter<String>? = null //ListView 변수와 붙일 Adapter
private var query = "select * from HomeWork.MeasData" //데이터 요청을 보낼 요청문
```

### ▶ onCreate() 작성

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

    listView = findViewById(R.id.listView)
    adapter = ArrayAdapter(this, android.R.layout.simple_list_item_1)
    listView?.adapter = adapter

    mTask = MyAsyncTask()
    mTask?.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, "") //Background thread 실행
}
```

# MSSQL과 연동하는 앱

## ▶ 코드작성

### ▶ MyAsyncTask Class에 MSSQL 서버에 접속을 하고 리스트뷰에 출력하는 코드 작성

```
inner class MyAsyncTask : AsyncTask<String, Void, ArrayList<String>>() {  
    override fun doInBackground(vararg params: String): ArrayList<String> {  
        val list = ArrayList<String>()  
        var reset: ResultSet?  
        var conn: Connection?  
  
        try {  
            Class.forName("net.sourceforge.jtds.jdbc.Driver") //사용하려는 라이브러리 이름  
            conn =  
DriverManager.getConnection("jdbc:jtds:sqlserver://115.71.52.30:14336;databaseName=Homework",  
"icsmanager", "gksrnrfpsxkfanstnwls")  
            //접속정보를 바탕으로 접속을 위한 코드  
            val stmt = conn!!.createStatement() //서버에 접속  
  
            reset = stmt.executeQuery(query) //데이터 가져오기
```

# MSSQL과 연동하는 앱

## ▶ 코드작성

### ▶ MyAsyncTask Class에 MSSQL 서버에 접속을 하고 리스트뷰에 출력하는 코드 작성

```
while (reset!!.next()) { //서버에서 MeasData를 전부 불러온다
    if (isCancelled) break //스레드 종료

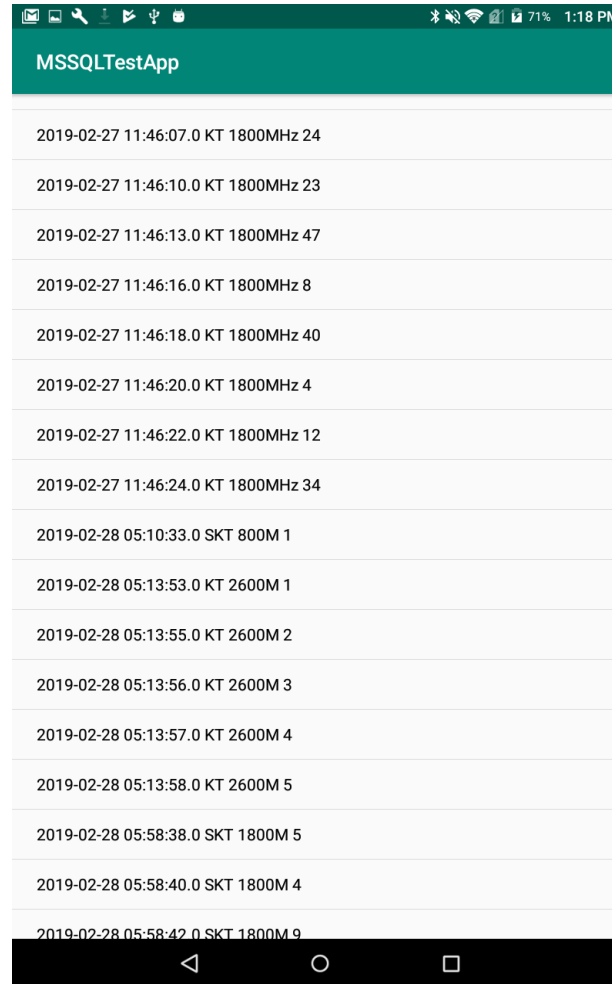
    val str = reset.getString(1) + " " + reset.getString(2) + " " + reset.getString(3) + " "
+ reset.getString(4)
    //하나의 데이터에 4개의 정보를 갖고 있음
    list.add(str) //리스트뷰에 넣을 문자열 리스트
}
conn.close() //접속종료
} catch (e: Exception) {
    Log.w("Error connection", "" + e.message)
}

return list
}

override fun onPostExecute(list: ArrayList<String>) {
    adapter?.clear()
    adapter?.addAll(list) //문자열 리스트를 리스트뷰에 삽입
    adapter?.notifyDataSetChanged()
}
```

# MSSQL과 연동하는 앱

## ▶ 실행화면



The screenshot displays the MSSQLTestApp interface on a mobile device. The app has a teal header bar with the title "MSSQLTestApp". Below the header is a list of data records, each containing a timestamp, a carrier code, and a frequency value. The records are separated by horizontal lines. The status bar at the top shows various icons and the time 1:18 PM. The bottom of the screen shows the Android navigation bar.

MSSQLTestApp		
2019-02-27 11:46:07.0	KT	1800MHz 24
2019-02-27 11:46:10.0	KT	1800MHz 23
2019-02-27 11:46:13.0	KT	1800MHz 47
2019-02-27 11:46:16.0	KT	1800MHz 8
2019-02-27 11:46:18.0	KT	1800MHz 40
2019-02-27 11:46:20.0	KT	1800MHz 4
2019-02-27 11:46:22.0	KT	1800MHz 12
2019-02-27 11:46:24.0	KT	1800MHz 34
2019-02-28 05:10:33.0	SKT	800M 1
2019-02-28 05:13:53.0	KT	2600M 1
2019-02-28 05:13:55.0	KT	2600M 2
2019-02-28 05:13:56.0	KT	2600M 3
2019-02-28 05:13:57.0	KT	2600M 4
2019-02-28 05:13:58.0	KT	2600M 5
2019-02-28 05:58:38.0	SKT	1800M 5
2019-02-28 05:58:40.0	SKT	1800M 4
2019-02-28 05:58:42.0	SKT	1800M 9



## 실습예제 2

- ▶ KTServerApp프로젝트를 구현하시오.
  - ▶ 한국렌탈 서버 정보를 이용하여 해당 테이블의 정보를 모두 가져옴
  - ▶ 리사이클러뷰를 사용하여 모든 레코드 정보를 출력
  - ▶ 해당 정보를 효율적으로 보여줄 수 있도록 item 구성(xml 파일)
  - ~~▶ 데이터를 저장하는 버튼을 생성하고 저장하는 화면 작성~~

# 데이터 검색하여 출력하기

- ▶ 앞의 실습 예제를 응용하여 서버에 있는 데이터를 다양한 방법으로 검색하여 가져오는 예제
  - ▶ 자동완성 텍스트 뷰, Spinner
- ▶ 새로운 프로젝트 생성
  - ▶ 프로젝트 명 : SearchDataApp
  - ▶ 안드로이드 버전 : 5.0 or 5.1
- ▶ JDBC 라이브러리 추가
  - ▶ 프로젝트에 MSSQL 서버에 접속할 수 있도록 JDBC 라이브러리인 JTDS를 추가
    - ▶ implementation 'net.sourceforge.jtds:jtds:1.3.1'
- ▶ 메니페스트 권한
  - ▶ 해당 앱 실행 시 인터넷 접속 허용
    - ▶ <uses-permission android:name="android.permission.INTERNET" />

# 데이터 검색하여 출력하기

## ▶ 리사이클러뷰로 출력할 아이템 형식 디자인(1)

- ▶ res - layout에 item\_main.xml 추가
- ▶ 릴레이티브 레이아웃에 텍스트뷰 4개 추가

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp">

    <TextView
        android:id="@+id/itemDate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="15dp"
        android:textStyle="bold"
        android:layout_marginLeft="16dp"
        android:ellipsize="end"
        android:maxLines="1"/>

    <TextView
        android:id="@+id/itemCarrier"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/itemDate"
        android:layout_alignLeft="@id/itemDate"
        android:ellipsize="end"
        android:maxLines="1"/>
```

# 데이터 검색하여 출력하기

## ▶ 리사이클러뷰로 출력할 아이템 형식 디자인(2)

▶ res - layout에 item\_main.xml 추가

▶ 릴레이티브 레이아웃에 텍스트뷰 4개 추가

```
<TextView
    android:id="@+id/itemFrequency"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/itemCarrier"
    android:layout_alignLeft="@id/itemCarrier"
    android:ellipsize="end"
    android:maxLines="1"/>

<TextView
    android:id="@+id/itemMeasuredValue"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/itemFrequency"
    android:layout_alignLeft="@id/itemFrequency"
    android:ellipsize="end"
    android:maxLines="1"/>
</RelativeLayout>
```

# 데이터 검색하여 출력하기

## ▶ 메인 액티비티 디자인(1)

### ▶ 리니어 레이아웃 내부에 자동완성텍스트뷰, 버튼, 스피너, 리사이클뷰 구성

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent" android:orientation="vertical"
    tools:context=".MainActivity">
    <AutoCompleteTextView
        android:id="@+id/autoComplete"
        android:hint="통신사"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
    <android.support.v7.widget.AppCompatButton
        android:id="@+id/searchBtn"
        android:text="검색"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
```

# 데이터 검색하여 출력하기

## ▶ 메인 액티비티 디자인(2)

### ▶ 리니어 레이아웃 내부에 자동완성텍스트뷰, 버튼, 스피너, SwipeRefreshLayout, 리싸이클러뷰

```
<Spinner
    android:id="@+id/spinnerView"
    android:layout_width="match_parent" android:layout_height="wrap_content"></Spinner>

<android.support.v4.widget.SwipeRefreshLayout
    android:id="@+id/swipeRefreshLo"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_marginTop="8dp"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</android.support.v4.widget.SwipeRefreshLayout>
</LinearLayout>
```

# 데이터 검색하여 출력하기

## ▶ MyAsyncTask 구현

▶ file - new - kotlin file/class

```
import android.os.AsyncTask
import android.util.Log
import java.sql.Connection
import java.sql.DriverManager
import java.sql.ResultSet

class MyAsyncTask : AsyncTask<String, Void, ArrayList<DataItem>>>(){
    interface OnUpdateListener {
        fun onUpdate(result: ArrayList<DataItem>)
    }
    var listener: OnUpdateListener? = null

    override fun doInBackground(vararg params: String): ArrayList<DataItem> {
        val list = ArrayList<DataItem>()
        var reset: ResultSet?
        var conn: Connection?
```

# 데이터 검색하여 출력하기

## ▶ MyAsyncTask 구현

### ▶ file - new - kotlin file/class

```
try {
    Class.forName("net.sourceforge.jtds.jdbc.Driver")
    conn =
DriverManager.getConnection("jdbc:jtds:sqlserver://115.71.52.30:14336;databaseName=Homework",
"icsmanager", "gksrnrfpsxkfanstnwls")
    val stmt = conn!!.createStatement()

    //쿼리 입력
    reset = stmt.executeQuery(params[0])

    while (reset!!.next()) {
        if (isCancelled) break

        //DataItem class에 데이터를 저장
        val dataItem = DataItem(reset.getString(1), reset.getString(2), reset.getString(3),
reset.getString(4))
        list.add(dataItem)
    }
    conn.close()
} catch (e: Exception) {
    Log.w("Error connection", "" + e.message)
}
return list
}
```



# 데이터 검색하여 출력하기

## ▶ MyAsyncTask 구현

▶ file - new - kotlin file/class

```
override fun onPostExecute(list: ArrayList<DataItem>) {  
    if (listener != null) {  
        listener?.onUpdate(list)  
    }  
}
```

# 데이터 검색하여 출력하기

## ▶ MainActivity 구현(1)

```
class MainActivity : AppCompatActivity() {
    private var mTask: MyAsyncTask? = null
    var list: MutableList<DataItem> = mutableListOf()

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

        searchBtn.setOnClickListener {
            mTask = MyAsyncTask()
            mTask?.listener = object: MyAsyncTask.OnUpdateListener {
                override fun onUpdate(result: ArrayList<DataItem>) {
                    recyclerView.layoutManager = LinearLayoutManager(this@MainActivity)
                    list = result
                    recyclerView.adapter = MyAdapter(list)
                }
            }
            mTask?.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, "select * from HomeWork.MeasData
where 통신사 = '" + autoComplete.text + "'")
        }
    }
}
```

# 데이터 검색하여 출력하기

## ▶ MainActivity 구현(2)

```
mTask = MyAsyncTask()
mTask?.listener = object: MyAsyncTask.OnUpdateListener {
    override fun onUpdate(result: ArrayList<DataItem>) {
        recyclerView.layoutManager = LinearLayoutManager(this@MainActivity)
        list = result
        var temp = list.distinctBy { it.Carrier }.map { it.Carrier }
        val autoText = temp.toTypedArray()
        val textAdapter = ArrayAdapter(this@MainActivity, android.R.layout.simple_list_item_1,
autoText)

        autoComplete.setAdapter(textAdapter)
        recyclerView.adapter = MyAdapter(list)
        recyclerView.addItemDecoration(MyDecoration())
    }
}
```

# 데이터 검색하여 출력하기

## ▶ MainActivity 구현(3)

```
        val spinnerAdapter = ArrayAdapter(this@MainActivity,
        android.R.layout.simple_spinner_item, autoText)
        spinnerAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

        spinnerView!!.adapter = spinnerAdapter
        spinnerView.onItemSelectedListener = object : AdapterView.OnItemSelectedListener{
            override fun onItemSelected(parent: AdapterView<*>?, view: View?, position: Int,
id: Long) {
                mTask = MyAsyncTask()
                mTask?.listener = object: MyAsyncTask.OnUpdateListener {
                    override fun onUpdate(result: ArrayList<DataItem>) {
                        recyclerView.layoutManager = LinearLayoutManager(this@MainActivity)
                        list = result
                        recyclerView.adapter = MyAdapter(list)
                    }
                }
                mTask?.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, "select * from
HomeWork.MeasData where 통신사 = '" + spinnerView.getItemAtPosition(position).toString() + "'")
            }

            override fun onNothingSelected(parent: AdapterView<*>?) {
            }
        }
    }
    mTask?.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, "select * from HomeWork.MeasData")
}
```

# 데이터 검색하여 출력하기

## ▶ MainActivity 구현(4)

```
class DataViewHolder(view: View) : RecyclerView.ViewHolder(view) {  
    val itemTestDate = view.itemDate  
    val itemCarrier = view.itemCarrier  
    val itemFrequency = view.itemFrequency  
    val itemMeasuredValue = view.itemMeasuredValue  
}
```

# 데이터 검색하여 출력하기

## ▶ MainActivity 구현(5)

```
inner class MyAdapter(val list: MutableList<DataItem>) :  
RecyclerView.Adapter<RecyclerView.ViewHolder>() {  
    //type이 하나뿐이라 0반환  
    override fun getItemViewType(position: Int): Int {  
        return 0  
    }  
  
    //item_main 로드  
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {  
        val inflater = LayoutInflater.from(parent?.context)  
        return DataViewHolder(inflater.inflate(R.layout.item_main, parent, false))  
    }  
  
    override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {  
        val dataItem = list.get(position)  
        val viewHolder = holder as DataViewHolder  
  
        viewHolder.itemTestDate.text = dataItem.TestDate  
        viewHolder.itemCarrier.text = dataItem.Carrier  
        viewHolder.itemFrequency.text = dataItem.Frequency  
        viewHolder.itemMeasuredValue.text = dataItem.MeasuredValue  
    }  
  
    override fun getItemCount(): Int {  
        return list.size  
    }  
}
```

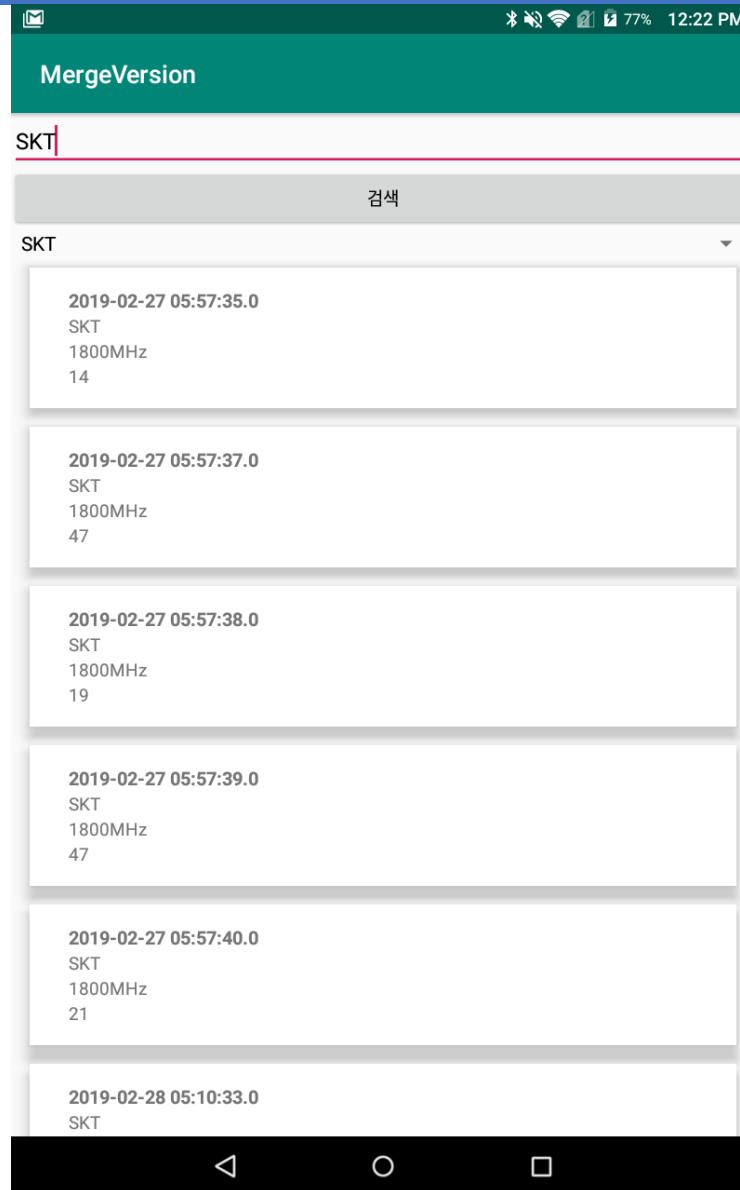
# 데이터 검색하여 출력하기

## ▶ MainActivity 구현(6)

```
    inner class MyDecoration() : RecyclerView.ItemDecoration() {
        override fun getItemOffsets(outRect: Rect, view: View, parent: RecyclerView, state:
RecyclerView.State) {
            super.getItemOffsets(outRect, view, parent, state)
            view!!.setBackgroundColor(0xFFFFFFFF.toInt())
            ViewCompat.setElevation(view, 10.0f)
            outRect!!.set(20, 10, 20, 10)
        }
    }
}
//DB의 데이터를 담을 class / 즉, 아이템을 담을 객체
class DataItem(var TestDate: String, var Carrier: String, var Frequency: String, var MeasuredValue:
String)
```

# 데이터 검색하여 출력하기

## ▶ 실행화면





# 연습문제

- ▶ 앞서 학습한 TodoList 프로젝트를 활용하여 서버에 데이터를 추가하는 액티비티와 코드를 작성하시오.

---

**Q & A**

---