CHAP 12. 파일과 데이터베이스

파일과 데이터베이스

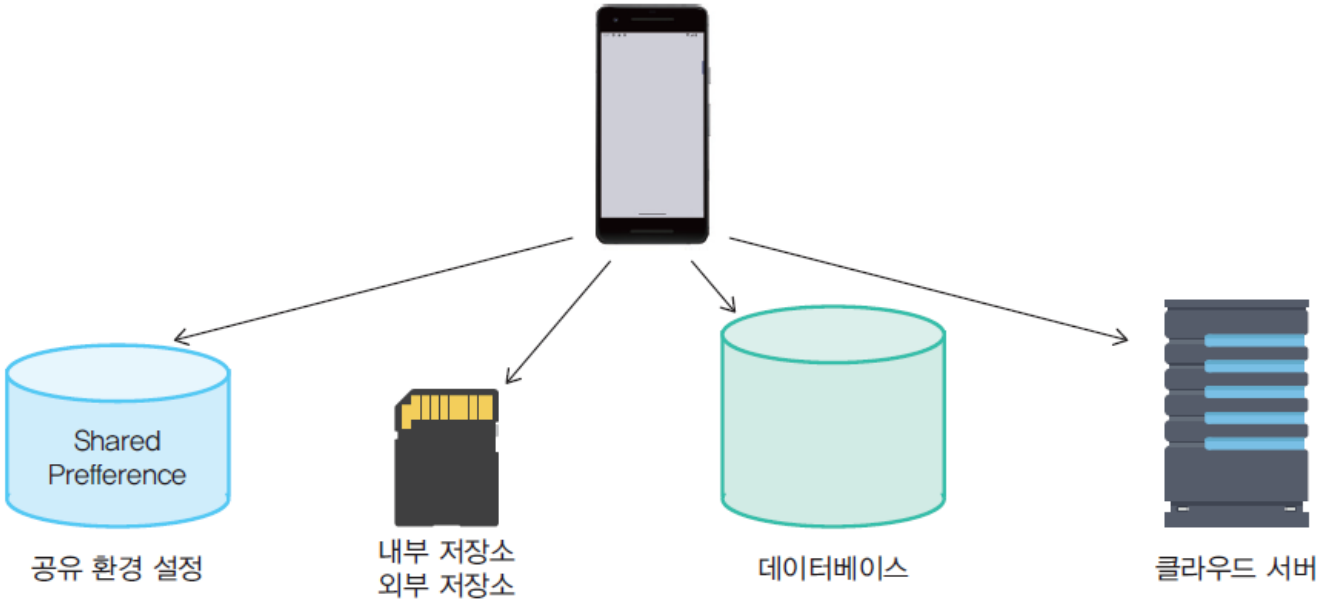

왜 이리 저장이 안 되는 거지?
금방 잊어먹는단 말야.
차분히 생각해봐. 안드로이드가 데이터를 저장하는 방법으로 파일과 데이터베이스가 있는데 모두 앱의 데이터를 영구적으로 설정하는 방법이지.

12장의 목표

- 영화 정보를 데이터베이스에 저장하는 앱을 작성해보자.

데이터를 저장하는 방법

- 안드로이드는 애플리케이션이 데이터를 저장하는 몇 가지 옵션을 제공한다.

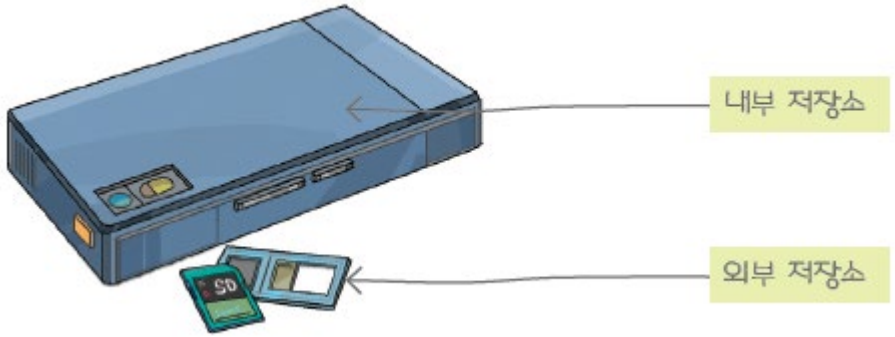


데이터를 저장하는 방법

방법	설명
앱별 저장소	앱 전용으로 사용하는 데이터를 내부 저장소 내의 전용 디렉토리 또는 외부 저장소 내의 전용 디렉토리에 파일 형태로 저장한다. 다른 앱이 액세스해서는 안 되는 민감한 정보라면 내부 저장소를 사용하여야 한다.
공유 저장소	문서, 사진, 동영상처럼 다른 앱과 공유하려는 파일을 저장한다.
환경설정	사적이고 기초적인 데이터를 키-값(key-value) 쌍으로 저장한다.
데이터베이스	SQLite나 Room 라이브러리를 사용하여 데이터베이스에 저장한다.
네트워크 연결	데이터를 클라우드 서버에 저장한다.

스마트폰 내부의 저장 공간

- 내부 저장소: 모든 장치에서 존재하고, 항상 사용할 수 있으므로 더 안정적으로 데이터를 보관할수 있다.
- 외부저장소: SD 카드와 같은 이동식 볼륨은 외부 저장소의 일부라고 생각할 수 있다.



스마트폰 내부의 저장 공간

- 내부 저장소: 내부 저장소에는 파일을 저장하는 디렉토리와 캐시 데이터를 저장하는 디렉토리가 포함되어 있다. 안드로이드 시스템은 다른 앱에서 이러한 디렉토리에 액세스하는 것을 방지하고, Android 10(API 수준 29) 이상에서는 이러한 디렉토리가 암호화된다. getFilesDir() 또는 getCacheDir()

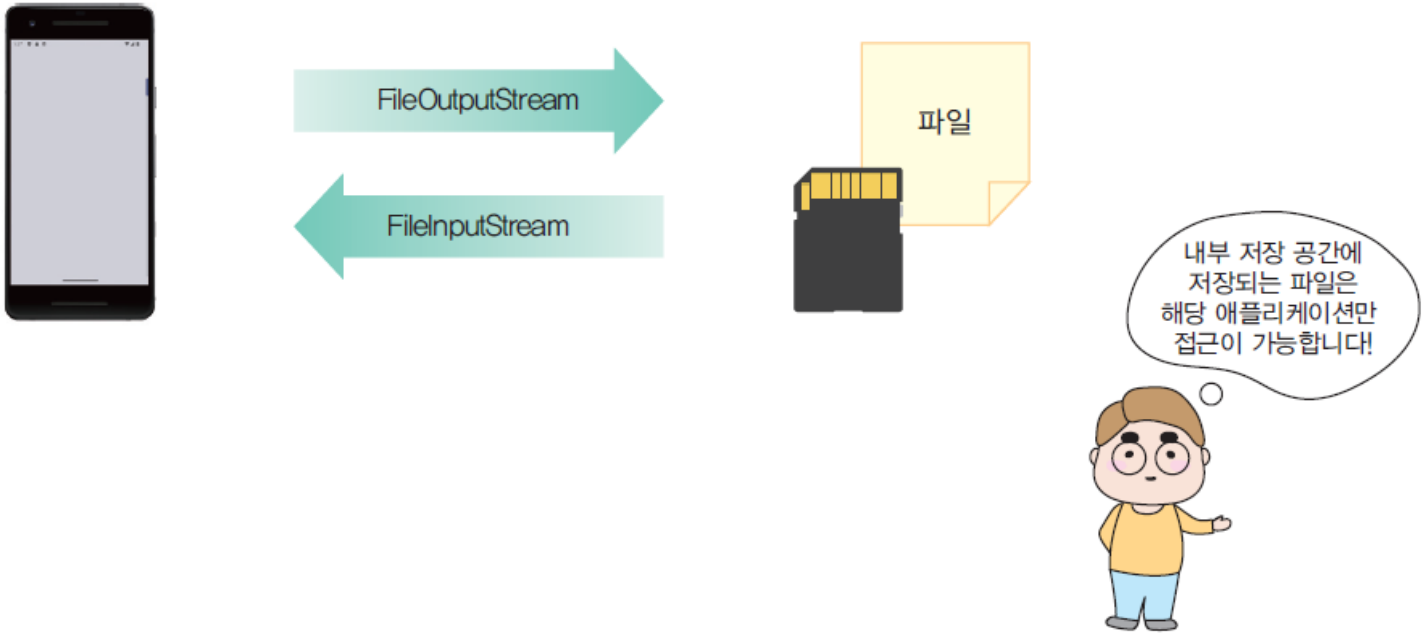
- 외부저장소: 다른 앱과 공유하는 파일을 만들려면 앱에서 이러한 파일을 외부 저장소의 공유 저장 공간 부분에 저장해야 한다. 다른 앱에 적절한 권한이 있는 경우 이러한 디렉터리에 액세스할 수 있다. getExternalFilesDir() 또는getExternalCacheDir()

외부 저장소 사용 권한

- 안드로이드에서 외부 저장소를 사용하려면 외부 저장소에 관한 읽기 및 쓰기 액세스권한(READ_EXTERNAL_STORAGE 및 WRITE_EXTERNAL_STORAGE)을 정의하여야 한다.

내부 공간에 파일 만들기

- 애플리케이션은 장치의 내부 저장 공간에 파일을 저장할 수 있다.
- 내부 저장 공간에 저장되는 파일은 해당 애플리케이션만 접근이 가능하다.
- 사용자가 애플리케이션을 제거하면 이들 파일들도 제거된다.

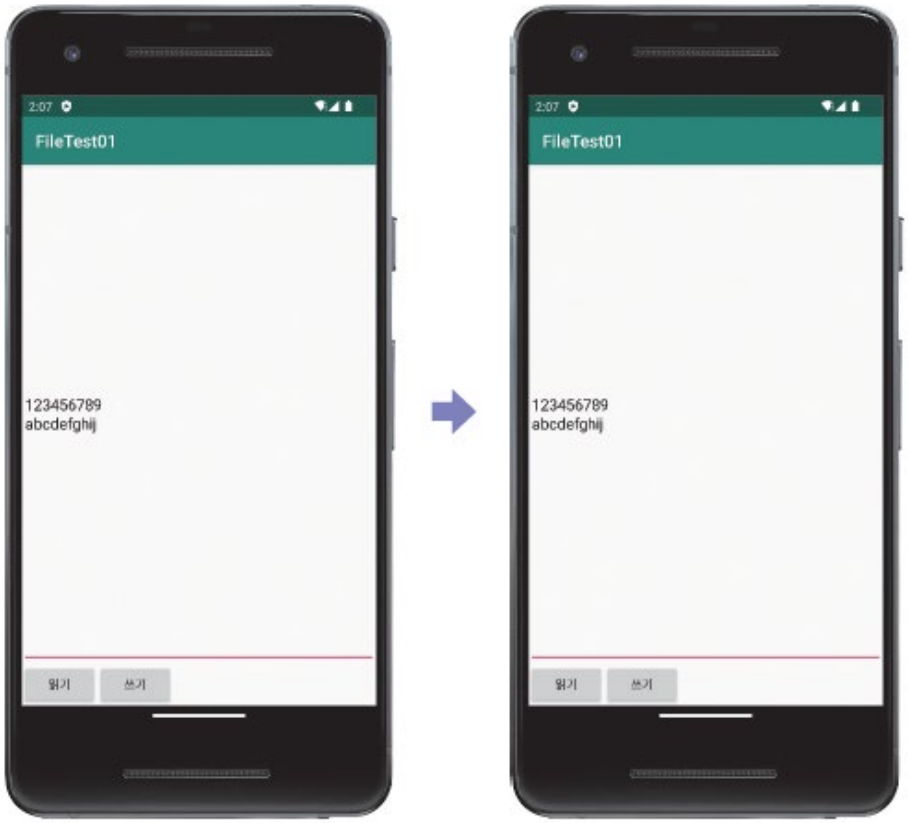


파일을 읽고 쓰는 절차

- 파일을 읽고 쓰려면 먼저 Context 클래스의 openFileInput( )이나 openFileOutput( )을 호출하여 FileInputStream이나 FileOutputStream 객체를 얻는다.
- 이들 객체의 read( ), write( )를 사용하면 데이터를 읽거나 쓸 수 있다.

예제: 내부 공간에 파일 만들기

- 사용자가 에디트 뷰에 입력한 텍스트를 파일에 저장하고 버튼을 누르면 이것을 읽어오는 애플리케이션 **FileTest01**을 작성하여 보자

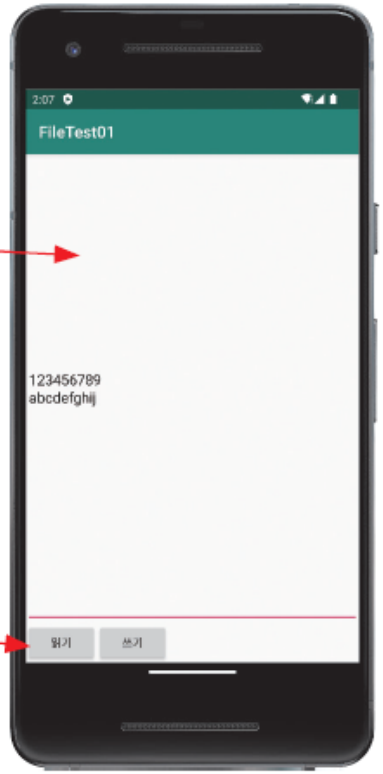


예제: 내부 공간에 파일 만들기: UI 설계

activity_main.xml

```
<LinearLayout ...
  <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent" >
      android:layout_weight="1"
      <EditText ...
          android:singleLine="false" />
  </LinearLayout>
  <LinearLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" >
      <Button ...          android:text="읽기" />
      <Button ...          android:text="쓰기" />
  </LinearLayout>
</LinearLayout>
```

가중치를 1로 하여서 남은 공간을 다 차지한다.



내부 공간에 파일 만들기

FileTest01.java

```
package kr.co.company.filetest01;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class FileTest01 extends AppCompatActivity {
  String FILENAME = "test.txt";
  EditText edit;

  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      edit = (EditText) findViewById(R.id.EditText01);
      Button readButton = (Button) findViewById(R.id.read);
      readButton.setOnClickListener(new View.OnClickListener() {
          public void onClick(View v) {
```

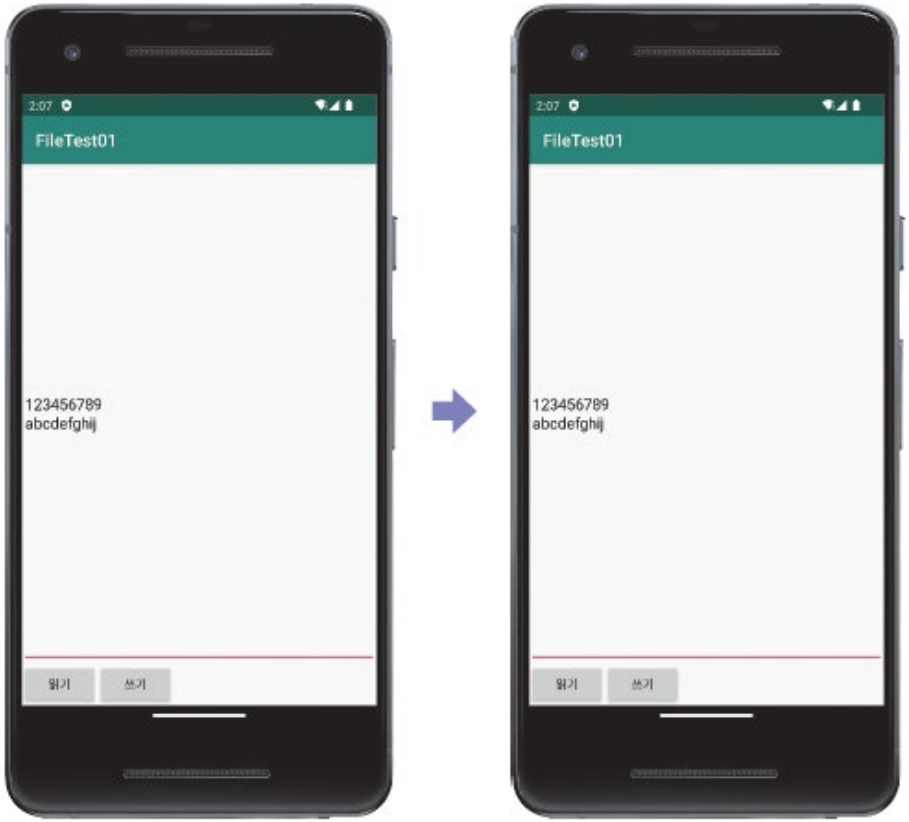
```
            try {
                FileInputStream fis = openFileInput(FILENAME);
                byte[] buffer = new byte[fis.available()];
                fis.read(buffer);
                edit.setText(new String(buffer));
                fis.close();
            } catch (IOException e) {
            }
        }
    });
    Button writeButton = (Button) findViewById(R.id.write);
    writeButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            try {
                FileOutputStream fos = openFileOutput(FILENAME,
                        Context.MODE_PRIVATE);
                fos.write(edit.getText().toString().getBytes());
                fos.close();
            } catch (IOException e) {
            }
        }
    });
  }
}
```

사용자가 "읽기" 버튼을 누르면 openFileInput() 메소드를 이용하여서 test.txt 파일을 오픈한다. 이 메소드는 파일 입력 스트림 객체를 반환하고 이 객체의 read() 메소드를 이용하여서 파일에 저장된 텍스트를 바이트 배열로 읽는다.

파일 입출력은 도중에 오류가 발생할 경우가 아주 많으므로 try/catch 블록으로 감싸주었다. 현재는 오류가 발생하면 아무것도 하지 않지만, 오류 메시지를 출력하는 편이 좋다.

먼저 에디트뷰에 텍스트를 입력하고 "쓰기" 버튼을 누르면 test.txt 파일을 openFileOutput() 메소드를 이용하여서 생성하였다. 이 메소드는 파일 출력 스트림 객체를 반환하고 이 객체의 write() 메소드를 이용하여서 에디트 뷰의 텍스트를 바이트 배열로 변환하여서 스트림에 기록한다.

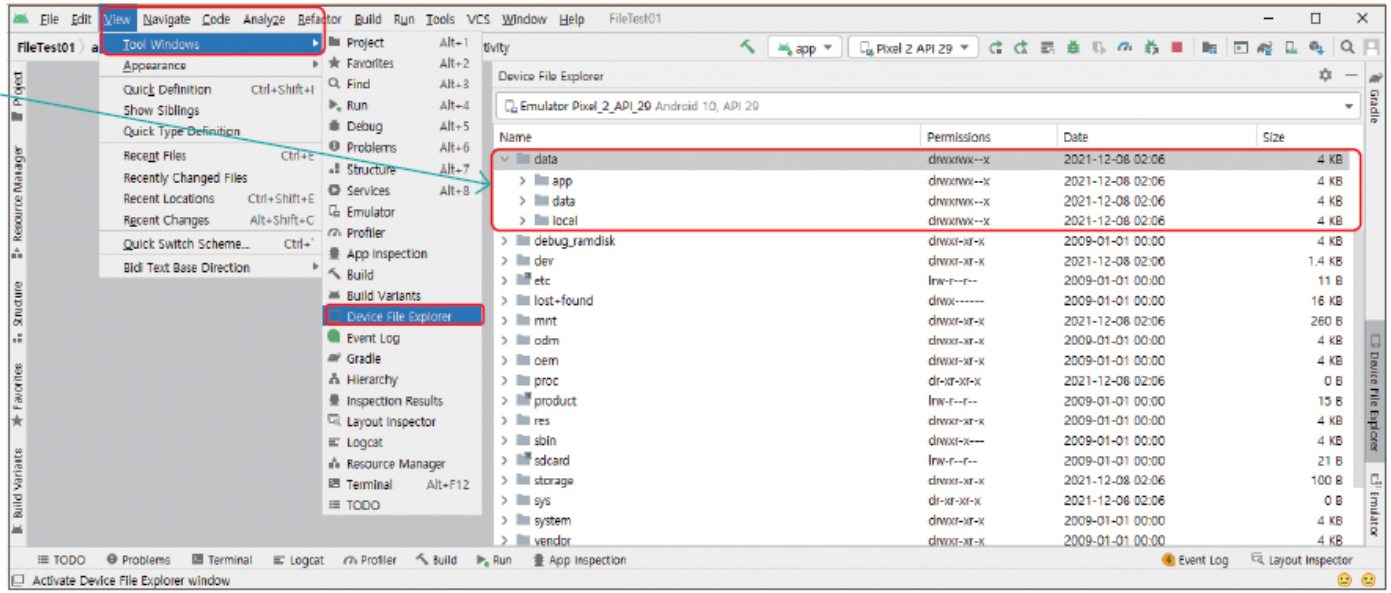
실행 결과


2:07
FileTest01
123456789
abcdefghij
읽기
쓰기
2:07
FileTest01
123456789
abcdefghij
읽기
쓰기

Device File Explorer

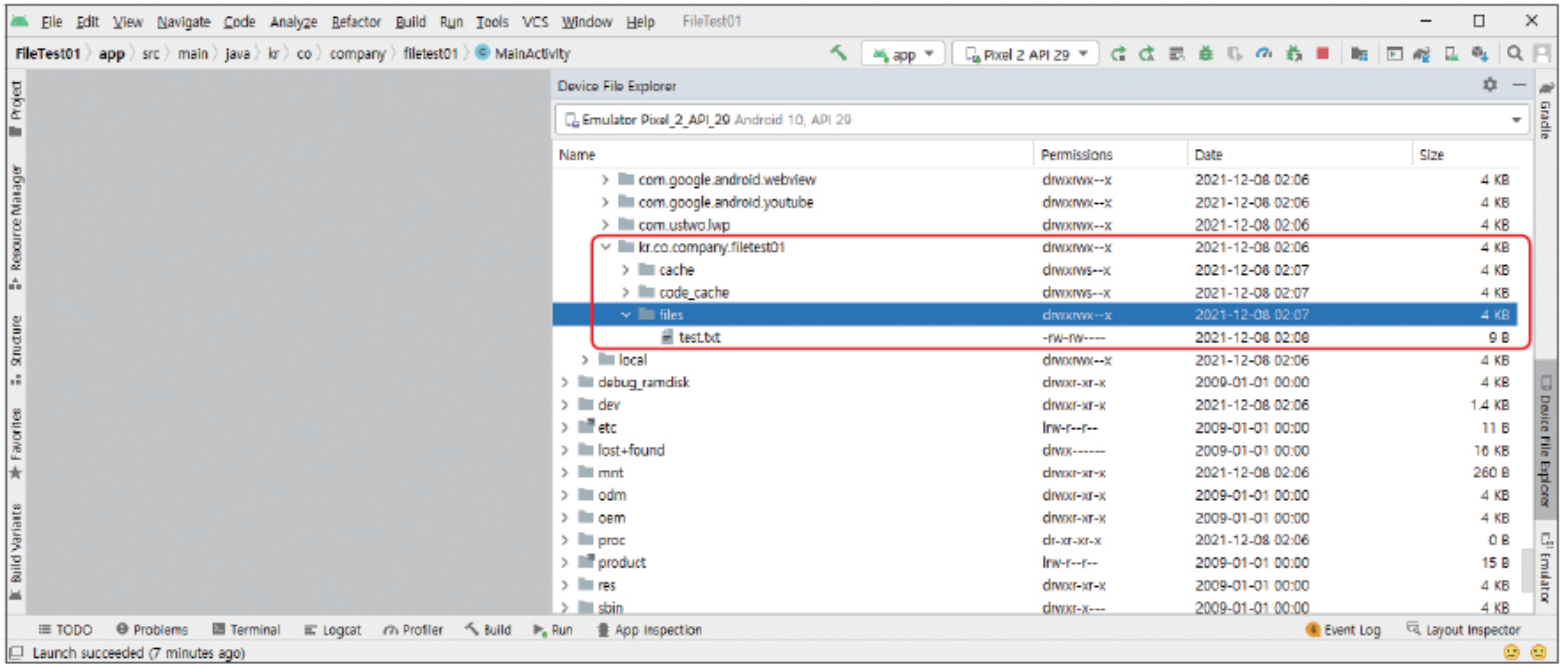
- [View] → [Tool Windows] → [Device File Explorer]를 클릭하여 Device File Explorer를 연다.

앱의 데이터들이 저장되는 디렉토리이다.



Device File Explorer

- 우리가 앞의 예제에서 생성한 test.txt 파일은 어디에 만들어졌을까? /data/data/kr.co.company.filetest01/files/에 만들어진 것을 확인할 수 있다.



내부저장소에서 유용한 메소드들

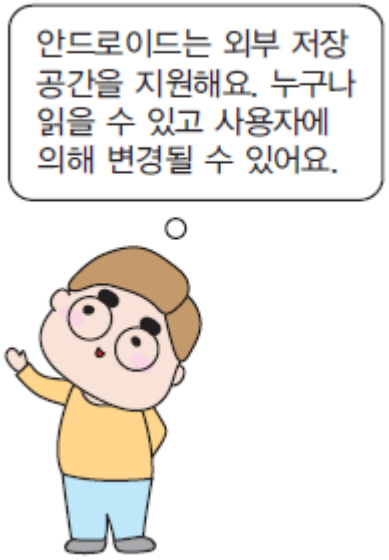
메소드	설명
getFilesDir()	내부 파일들이 저장되는, 파일 시스템 디렉토리의 절대 경로를 반환한다.
getDir()	내부 저장소의 디렉토리를 생성하거나 오픈한다.
deleteFile()	내부 저장소에 저장된 파일을 삭제한다.
fileList()	애플리케이션이 현재 저장한 파일 리스트를 반환한다.

Coding Challenge:
메모 앱 만들기

- 메모를 저장하는 파일 이름을 입력받는다. 메모의 내용도 입력받아서 내부 저장소에 메모를 파일로 저장한다.

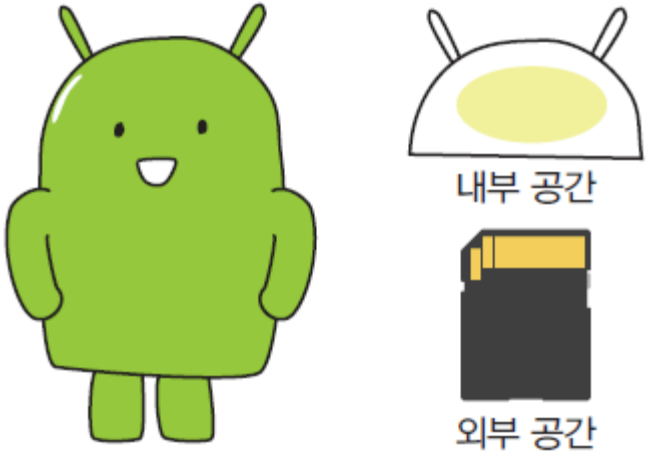
외부 공간에 사적 데이터 저장하기

- 외부 저장 공간은 착탈이 가능한 SD 카드
- 외부 저장 공간에 저장된 파일들은 누구나 읽을 수 있으며 사용자에 의해서 변경될 수 있다.
- 데이터를 읽기 전에 외부 미디어가 장착되어 있는지를 확인



내부 공간과 외부 공간 비교

- 내부 저장소는 주로 장치 안에 내장된 메모리이고 외부 저장소는 마이크로 SD 카드로서 착탈이 가능하다.
- 외부 저장소에도 앱별 데이터를 저장할 수 있다. 또 앱이 다른 앱과 공유해야 할 수 있는 이미지, 오디오, 비디오, 문서 등과 같은 데이터 파일도 저장이 가능하다



외부 미디어 장착 여부 검사

```
String state = Environment.getExternalStorageState();
if(state.equals(Environment.MEDIA_MOUNTED)) {
  // 미디어에 쓰고 읽을 수 있다.
} else if(state.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
  // 미디어를 읽을 수 있다.
} else {
  // 쓰거나 읽을 수 없다.
}
```

외부 저장소에 데이터를 저장하는 메소드

- getExternalStoragePublicDirectory( ): 이것은 파일을 공개적으로 저장하기 위해 권장되는 방법이며, 이러한 파일은 시스템에서 앱을 제거해도 삭제되지 않는다. 예를 들어서 카메라가 촬영한 이미지는 카메라를 제거한 후에도 계속 사용할 수 있다.

- getExternalFilesDir(String type): 이 메소드는 앱에만 해당하는 사적 데이터를 저장하는 데 사용된다. 앱을 제거하면 데이터가 제거된다.

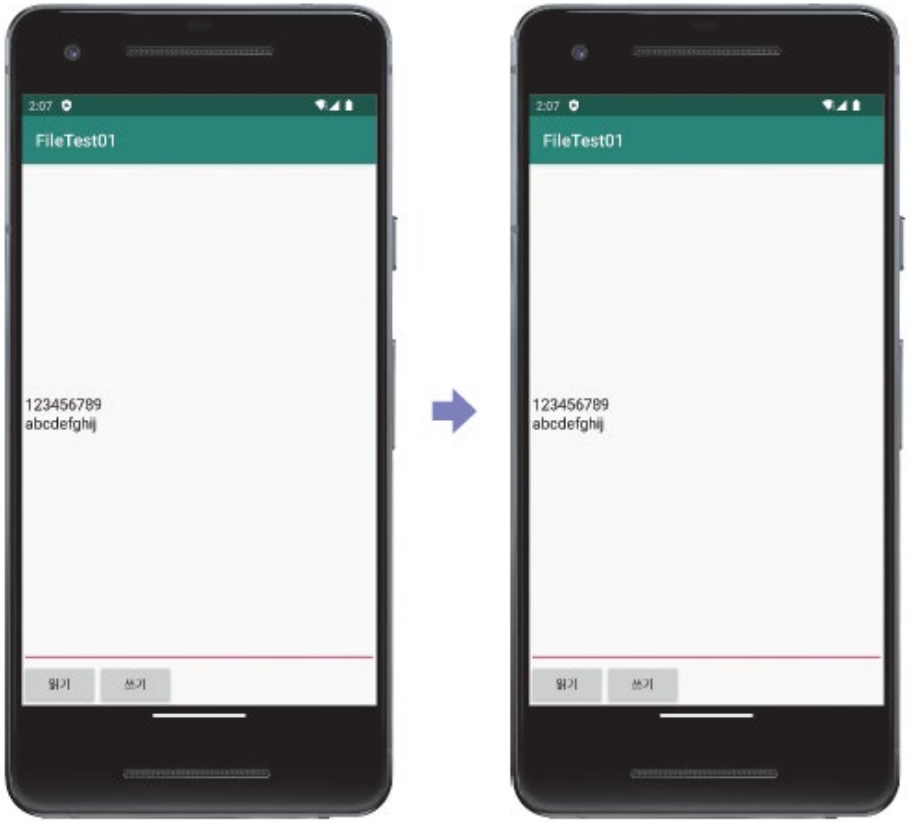
매니페스트 파일

- 외부 저장소의 파일을 읽거나 쓰려면 앱이 READ_EXTERNAL_STORAGE 또는 WRITE_EXTERNAL_STORAGE 시스템 권한을 획득해야 한다

```
<manifest ...>
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
  ...
</manifest>
```

예제: 외부 공간에 파일 만들기

- 텍스트 데이터를 외부 저장소의 사적 공간에 저장

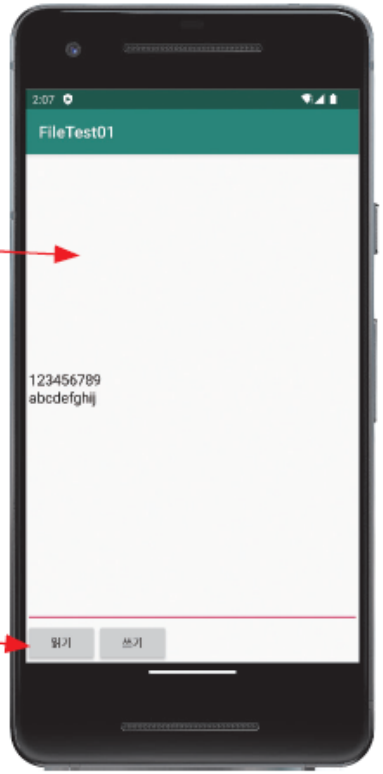


예제: 외부 공간에 파일 만들기

activity_main.xml

```
<LinearLayout ...
  <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent" >
      android:layout_weight="1"
      <EditText ...
          android:singleLine="false" />
  </LinearLayout>
  <LinearLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" >
      <Button ...          android:text="읽기" />
      <Button ...          android:text="쓰기" />
  </LinearLayout>
</LinearLayout>
```

가중치를 1로 하여서 남은 공간을 다 차지한다.



예제: 외부 공간에 파일 만들기 예제

MainActivity.java

```
package kr.co.company.filetest02;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class MainActivity extends AppCompatActivity {
  String FILENAME = "test.txt";
  EditText edit;

  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      String state = Environment.getExternalStorageState();
      if(state.equals(Environment.MEDIA_MOUNTED)==false){
            Toast.makeText(this, "외부 스토리지 실패", Toast.LENGTH_SHORT).show();
      }
      edit = (EditText) findViewById(R.id.EditText01);
      Button readButton = (Button) findViewById(R.id.read);
```

외부 저장 공간 체크

예제: 외부 공간에 파일 만들기 예제

외부 저장 공간 파일 읽기

```
readButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        File file = new File(getExternalFilesDir(null), FILENAME);
        try {
            InputStream is;
            is = new FileInputStream(file);
            byte[] buffer = new byte[is.available()];
            is.read(buffer);
            edit.setText(new String(buffer));
            is.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
Button writeButton = (Button) findViewById(R.id.write);
writeButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
```

예제: 외부 공간에 파일 만들기 예제

```
            File file = new File(getExternalFilesDir(null), FILENAME);
            try {
                OutputStream os = new FileOutputStream(file);
                os.write(edit.getText().toString().getBytes());
                os.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
  }
}
```

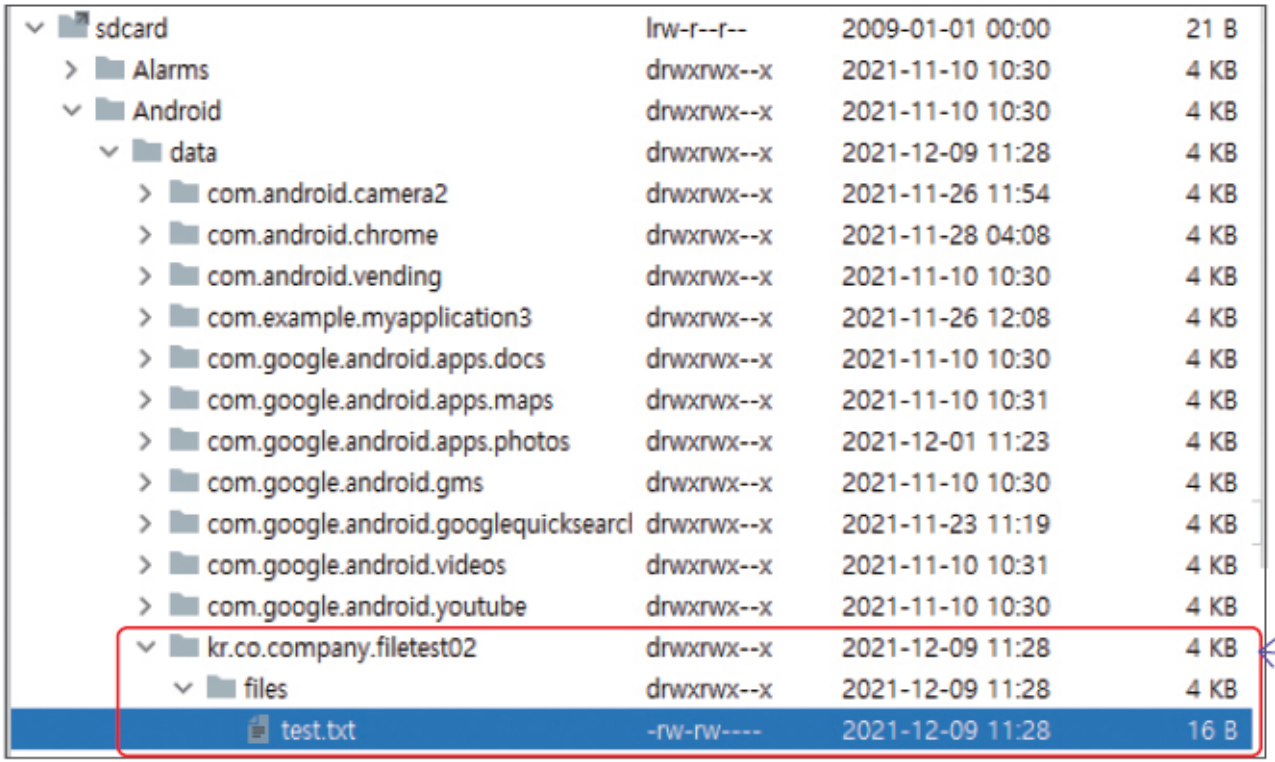
외부 저장 공간 파일 쓰기

매니페스트 파일

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    ...
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    ...
</manifest>
```

실행 결과


sdcard lrw-r--r-- 2009-01-01 00:00 21 B
Alarms drwxrwx--x 2021-11-10 10:30 4 KB
Android drwxrwx--x 2021-11-10 10:30 4 KB
data drwxrwx--x 2021-12-09 11:28 4 KB
com.android.camera2 drwxrwx--x 2021-11-26 11:54 4 KB
com.android.chrome drwxrwx--x 2021-11-28 04:08 4 KB
com.android.vending drwxrwx--x 2021-11-10 10:30 4 KB
com.example.myapplication3 drwxrwx--x 2021-11-26 12:08 4 KB
com.google.android.apps.docs drwxrwx--x 2021-11-10 10:30 4 KB
com.google.android.apps.maps drwxrwx--x 2021-11-10 10:31 4 KB
com.google.android.apps.photos drwxrwx--x 2021-12-01 11:23 4 KB
com.google.android.gms drwxrwx--x 2021-11-10 10:30 4 KB
com.google.android.googlequicksearcl drwxrwx--x 2021-11-23 11:19 4 KB
com.google.android.videos drwxrwx--x 2021-11-10 10:31 4 KB
com.google.android.youtube drwxrwx--x 2021-11-10 10:30 4 KB
kr.co.company.filetest02 drwxrwx--x 2021-12-09 11:28 4 KB
files drwxrwx--x 2021-12-09 11:28 4 KB
test.txt -rw-rw---- 2021-12-09 11:28 16 B
여기에 사적인 파일이 저장 되어 있다.

외부 저장소에 공유 데이터 저장하기

- 사용자가 앱을 통하여 획득한 새로운 미디어 파일들은 원칙적으로 다른 앱들도 접근할 수 있는 공용 디렉터리에 저장되어야 한다.
- 공용 디렉토리에 저장해야만 사용자가 앱을 삭제하더라도 사진과 같은 사용자 데이터가 삭제되지 않는다.
- 앱은 플랫폼의 MediaStore API를 사용하여 이 콘텐츠에 액세스할 수 있다.

공용 디렉토리

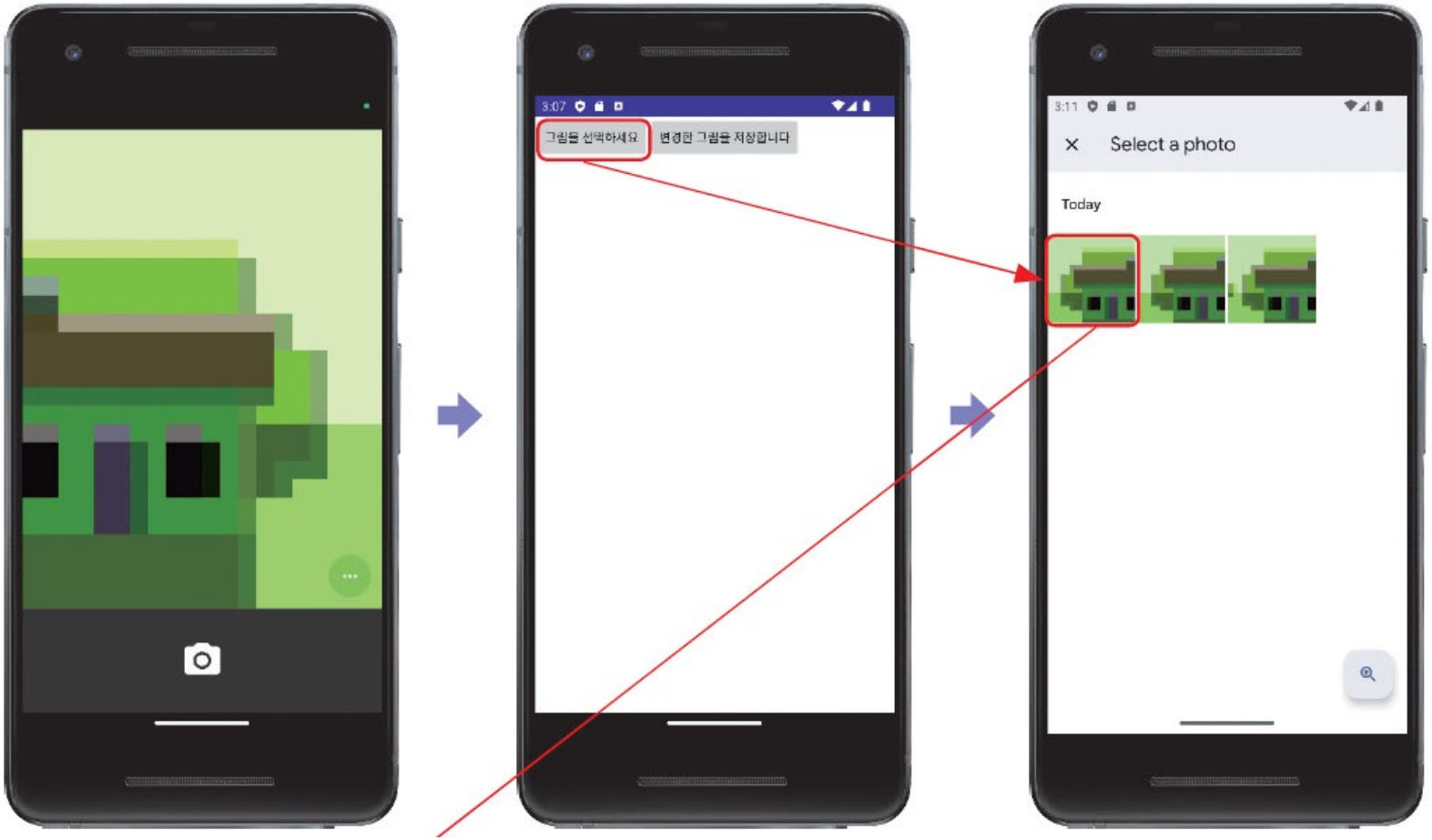
- 공용 디렉토리는 외부 저장 공간의 루트에 위치하며, Music/, Pictures/, Ringtones/와 같은 이름을 사용한다.
  - 이미지: 사진 및 스크린샷을 포함하며, DCIM/ 및 Pictures/ 디렉토리에 저장된다.
  - 동영상: DCIM/, Movies/, Pictures/ 디렉토리에 저장된다.
  - 오디오 파일: Alarms/, Audiobooks/, Music/ 디렉토리에 저장한다.
  - 다운로드한 파일: Download/ 디렉토리에 저장된다.

외부 저장소에 공유 데이터 저장하기

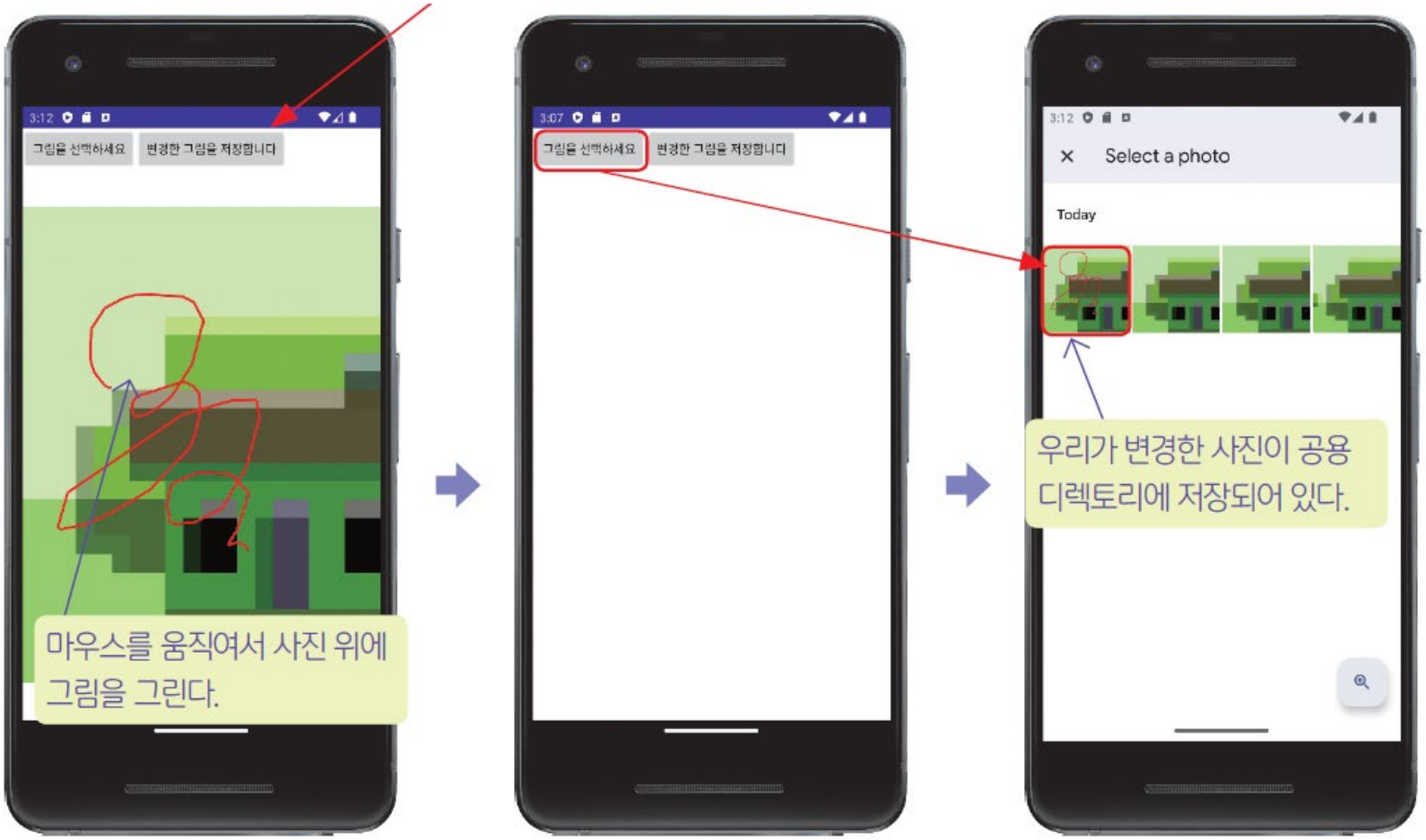
- 안드로이드 버전이 Q 버전 이상이면 공용 데이터를 저장할 때, 콘텐트 제공자를 이용하는 것이 바람직하다.

- 안드로이드 버전이 Q 버전 미만이면 getExternalStoragePublicDirectory()을 호출하여 공용 디렉토리를 얻은 후에 파일을 생성하면 된다.

예제: 이미지를 수정하여 외부 공용 저장소에 저장하기



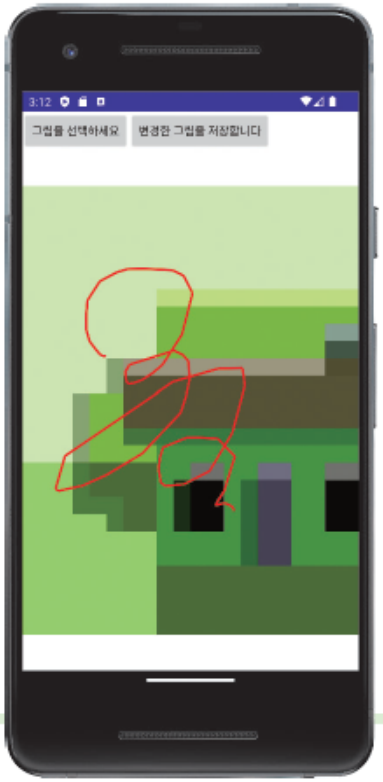
예제: 이미지를 수정하여 외부 공용 저장소에 저장하기



예제: 이미지를 수정하여 외부 공용 저장소에 저장하기

activity_main.xml

```
<LinearLayout>
    <LinearLayout
    android:orientation="horizontal">

    <Button
        android:id="@+id/choose"
        android:onClick="choose"
        android:text="그림을 선택하세요" />
    <Button
        android:id="@+id/save"
        android:onClick="save"
        android:text="변경한 그림을 저장합니다" />
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"></ImageView>
</LinearLayout>
```



예제: 이미지를 수정하여 외부 공용 저장소에 저장하기

MainActivity.java

```
package kr.co.company.publicimage;
...
public class MainActivity extends Activity implements View.OnTouchListener {
    ImageView imageView;
    Bitmap orgBitmap;
    Bitmap changedBitmap;
    Canvas canvas;
    Paint paint;
    Matrix matrix;
    float x1=0, y1=0, x2=0, y2=0;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        imageView = (ImageView) this.findViewById(R.id.imageView);
        imageView.setOnTouchListener(this);
    }
```

버튼이 눌리면 암시적인 인텐트 ACTION_PICK을 시작한다.

```
    public void choose(View v) {
        Intent choosePictureIntent = new Intent(Intent.ACTION_PICK, android.provider.MediaStore.
Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(choosePictureIntent, 0);
    }
```

```
public void save(View v) {
    OutputStream fos;

    if (changedBitmap == null) return;
    try {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
            ContentValues contentValues = new ContentValues(3);
            contentValues.put(MediaStore.Images.Media.DISPLAY_NAME, "My Pictures");
            Uri imageFileUri = getContentResolver().insert(MediaStore.Images.Media.EXTERNAL_
CONTENT_URI, contentValues);
            fos = getContentResolver().openOutputStream(imageFileUri);
        } else {
            String imagesDir = Environment.getExternalStoragePublicDirectory(Environment.
DIRECTORY_PICTURES).toString();
            File image = new File(imagesDir, "My Picture");
            fos = new FileOutputStream(image);
        }
        changedBitmap.compress(Bitmap.CompressFormat.JPEG, 90, fos);
    } catch (Exception e) {
        Log.v("EXCEPTION", e.getMessage());
    }
}
```

저장 버튼이 눌리면 컨텐츠 제공자를 이용하여 공용 디렉토리에 변경된 이미지를 저장한다.

예제: 이미지를 수정하여 외부 공용 저장소에 저장하기

```
    protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
        super.onActivityResult(requestCode, resultCode, intent);
        if (resultCode == RESULT_OK) {
            Uri uri = intent.getData();
            try {
                BitmapFactory.Options bmpFactoryOptions = new BitmapFactory.Options();
                orgBitmap = BitmapFactory.decodeStream(getContentResolver().
openInputStream(uri), null, bmpFactoryOptions);
                changedBitmap = Bitmap.createBitmap(orgBitmap.getWidth(), orgBitmap.getHeight(),
orgBitmap.getConfig());
                canvas = new Canvas(changedBitmap);
                paint = new Paint();
                paint.setColor(Color.RED);
                paint.setStrokeWidth(10);
                matrix = new Matrix();
                canvas.drawBitmap(orgBitmap, matrix, paint);
                imageView.setImageBitmap(changedBitmap);
                imageView.setOnTouchListener(this);
            } catch (Exception e) {
                Log.v("ERROR", e.toString());
            }
        }
    }
```

공용 디렉토리에서 이미지가 선택되면 이미지를 읽어서 이미지 뷰에 표시한다. 그림을 그릴 수 있도록 캔버스 객체에 이미지를 그린다.

예제

```
public boolean onTouch(View v, MotionEvent event) {
    int action = event.getAction();
    switch (action) {
            case MotionEvent.ACTION_DOWN:
                x1 = event.getX();
                y1 = event.getY();
                break;
            case MotionEvent.ACTION_MOVE:
                x2 = event.getX();
                y2 = event.getY();
                canvas.drawLine(x1, y1, x2, y2, paint);
                imageView.invalidate();
                x1 = x2;
                y1 = y2;
                break;
            case MotionEvent.ACTION_UP:
                x2 = event.getX();
                y2 = event.getY();
                canvas.drawLine(x1, y1, x2, y2, paint);
                imageView.invalidate();
                break;
            case MotionEvent.ACTION_CANCEL:
                break;
            default:
                break;
        }
        return true;
    }
}
```

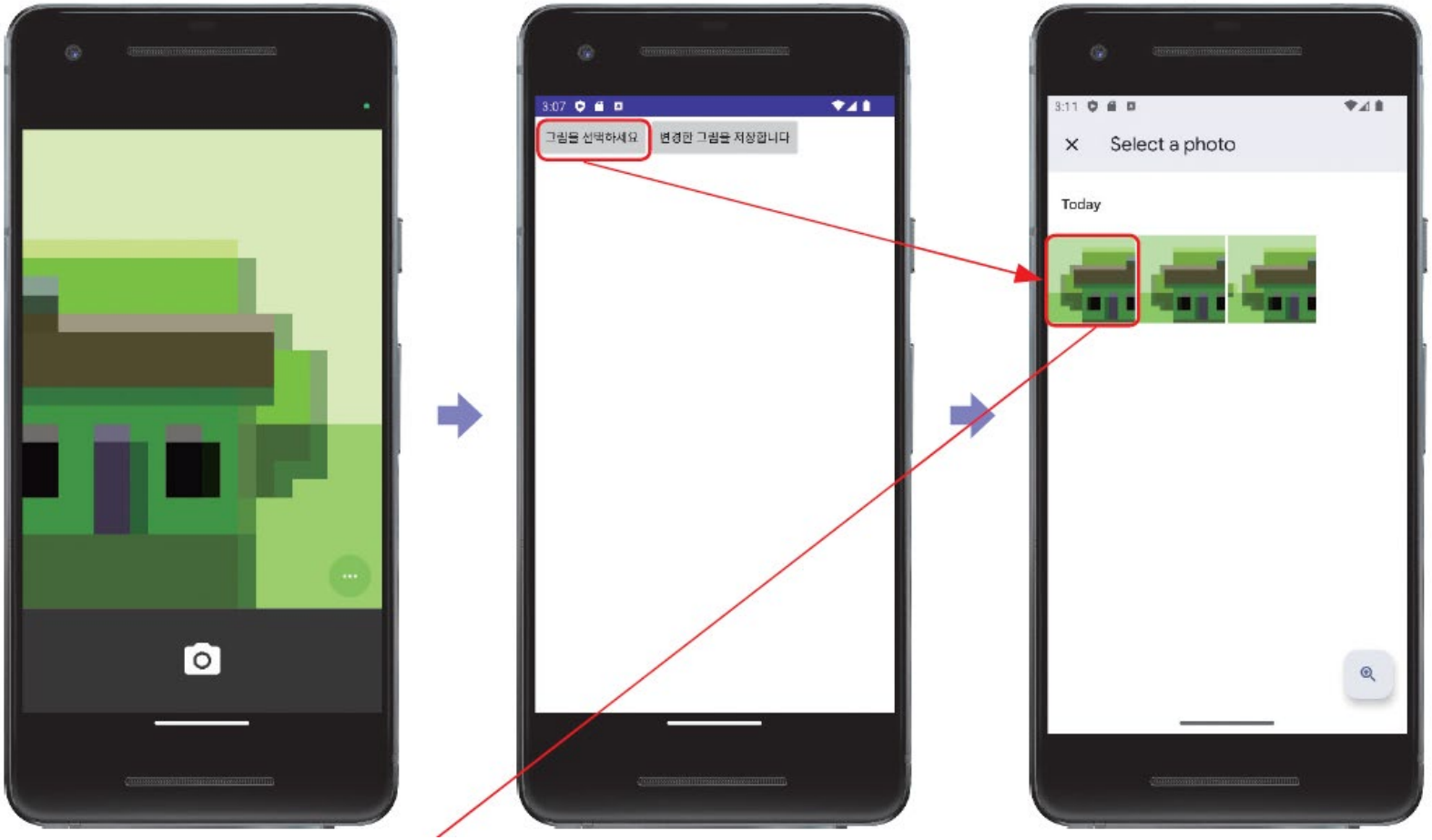
터치 이벤트가 발생하면 이미지 위에 빨간색 직선을 연결하여 그린다.

예제: 이미지를 수정하여 외부 공용 저장소에 저장하기

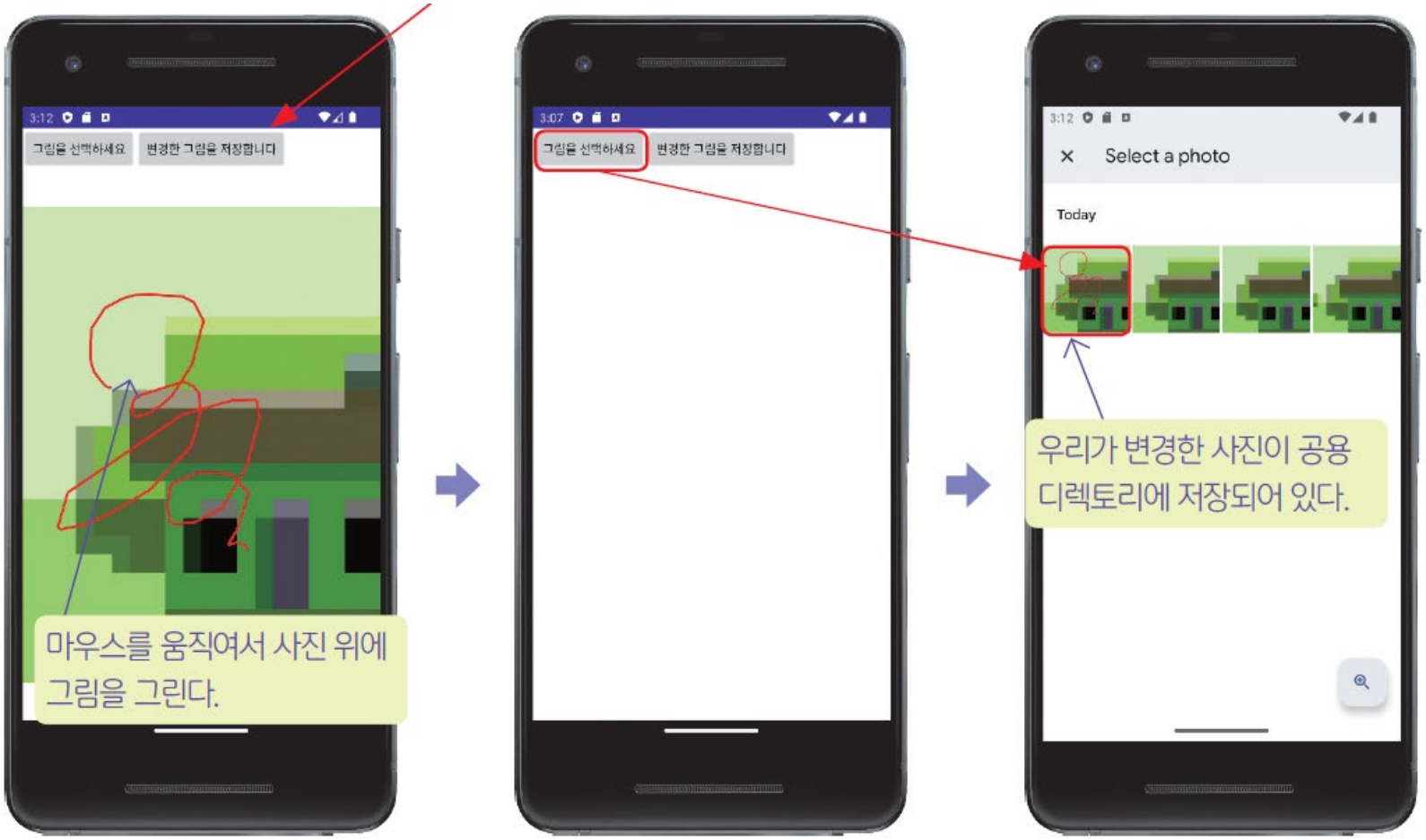
AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    ...
    <uses-permission
      android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
      android:name="android.permission.READ_EXTERNAL_STORAGE" />
  ...
</manifest>
```

예제: 이미지를 수정하여 외부 공용 저장소에 저장하기

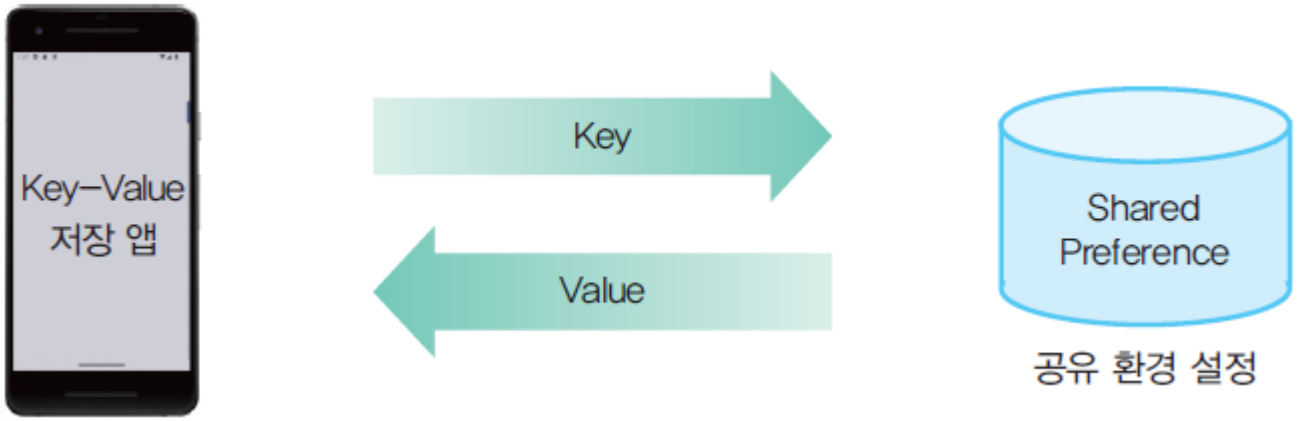


예제: 이미지를 수정하여 외부 공용 저장소에 저장하기



공유 프레퍼런스

- 애플리케이션의 환경 설정
- 기초적인 자료형을 키-값 쌍으로 저장하고 복원할 수 있는 방법
- 저장된 데이터는 사용자 애플리케이션이 종료되더라도 저장



공유 프레퍼런스를 얻는 방법

- getSharedPreferences(name, mode)
  - 이름으로 식별되는 여러 개의 프레퍼런스 파일이 필요하면 이 메소드를 사용한다.
- getPreferences(mode)
  - 하나의 프레퍼런스 파일만 필요하다면 이것을 사용한다. 이 파일은 액티비티마다 하나만 존재하므로 파일 이름이 필요 없다.

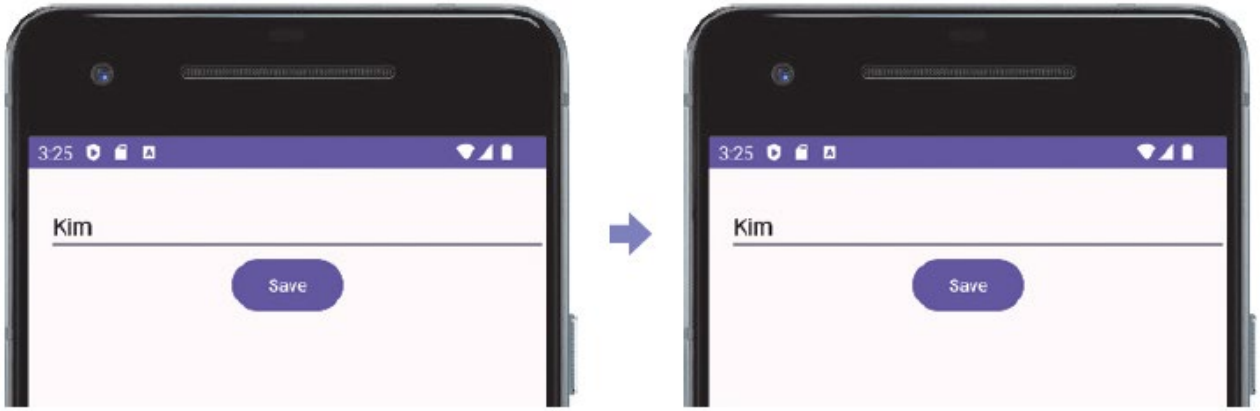
프레퍼런스 파일에서 값을 읽거나 쓰는 메소드

- getBoolean(String key, boolean defValue)
- getInt(String key, int defValue)
- getString(String key, String defValue)

- putBoolean(String key, boolean value)
- putInt(String key, int value)
- putString(String key, String value)

예제: 문자열을 프레퍼런스에 기록하고 읽는다.

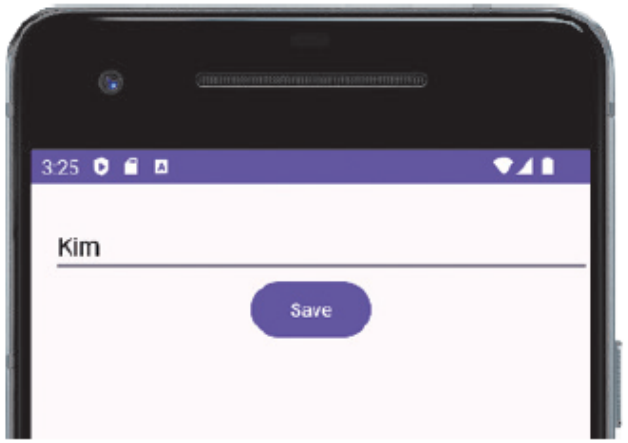
- 문자열을 입력하고 BACK를 누르더라도 문자열이 없어지지 않는 예제



예제: 사용자 인터페이스 작성

activity_main.xml

```
<androidx.constraintlayout.widget.ConstraintLayout >
    <EditText
        android:id="@+id/editText"
        android:hint="Enter your name" />

    <Button
        android:id="@+id/saveButton"
        android:text="Save" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



예제: 액티비티 작성

PrefTest01.java

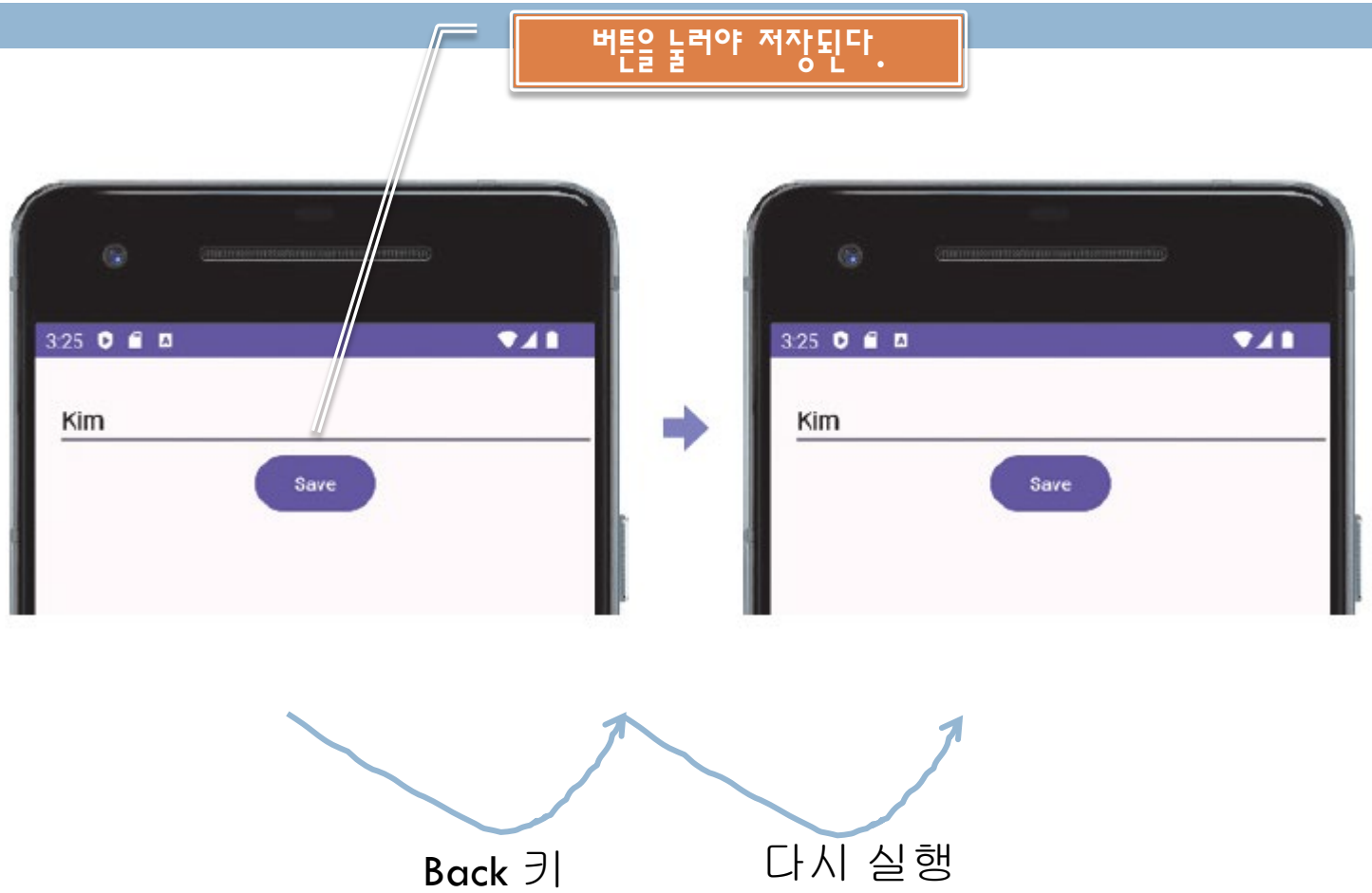
```
public class MainActivity extends AppCompatActivity {
    private EditText editText;

    private Button saveButton;

    private String sharedPrefFile = "my_settings";
    private SharedPreferences sharedPreferences;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        sharedPreferences = getSharedPreferences(sharedPrefFile, Context.MODE_PRIVATE);

        saveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                saveSettings();
            }
        });

        // 앱이 시작될 때 저장된 설정을 읽어옵니다.
        loadSettings();
    }

    private void saveSettings() {
        String username = editText.getText().toString();
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString("username", username);
        editor.apply();
    }

    private void loadSettings() {
        String username = sharedPreferences.getString("username", "");
        editText.setText(username);
    }
}
```

공유 프레퍼런스에 이름을 저장한다.

공유 프레퍼런스에서 이름을 읽는다.

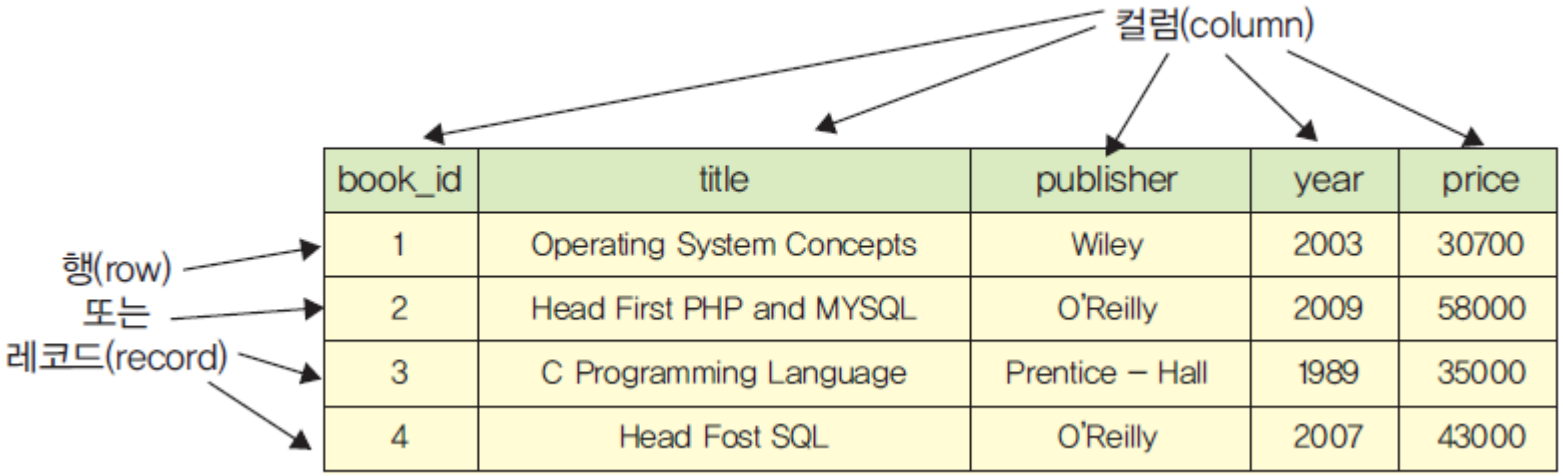
실행 결과



안드로이드에서의 데이터베이스

- **SQLite**
- www.sqlite.org
- SQLite는 안드로이드 와 아이폰을 비롯한 많은 모바일 장치에서 사용되는 데이터베이스
- SQL을 거의 완전하게 지원

데이터베이스



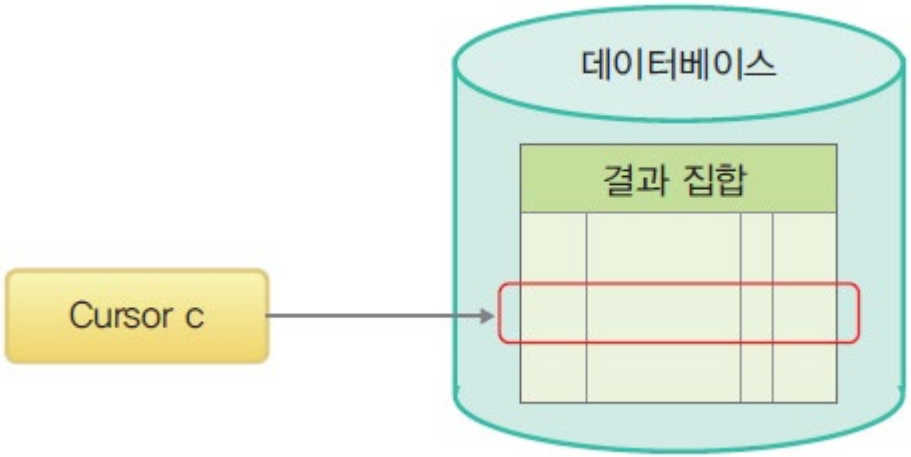
book_id	title	publisher	year	price
1	Operating System Concepts	Wiley	2003	30700
2	Head First PHP and MYSQL	O'Reilly	2009	58000
3	C Programming Language	Prentice – Hall	1989	35000
4	Head Fost SQL	O'Reilly	2007	43000

SQL

구분	명령어	설명
데이터 정의 명령어 (Data Definition Language)	CREATE	사용자가 제공하는 컬럼 이름을 가지고 테이블을 생성한다. 사용자는 컬럼의 데이터 타입도 지정해야 한다. 데이터 타입은 데이터베이스에 따라 달라진다. 이미 테이블이 만들어져 있는 경우가 많기 때문에 CREATE TABLE은 통상적으로 DML보다 적게 사용된다.
	ALTER	테이블에서 컬럼을 추가하거나 삭제한다.
	DROP	테이블의 모든 레코드를 제거하고 테이블의 정의 자체를 데이터베이스로부터 삭제하는 명령어이다.
	USE	어떤 데이터베이스를 사용하는지 지정한다.
데이터 조작 명령어 (Data Manipulation Language)	SELECT	데이터베이스로부터 데이터를 쿼리하고 출력한다. SELECT 명령어들은 결과 집합에 포함시킬 컬럼을 지정한다. SQL 명령어 중에서 가장 자주 사용된다.
	INSERT	새로운 레코드를 테이블에 추가한다. INSERT는 새롭게 생성된 테이블을 채우거나 새로운 레코드를 이미 존재하는 테이블에 추가할 때 사용된다.
	DELETE	지정된 레코드를 테이블로부터 삭제한다.
	UPDATE	테이블에서 레코드에 존재하는 값을 변경한다.

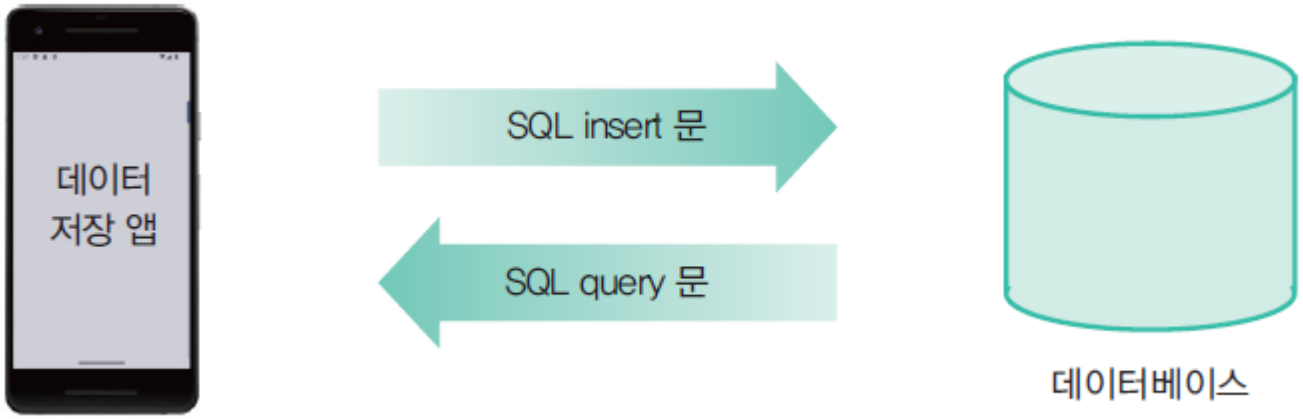
결과집합

- 쿼리의 조건을 만족하는 레코드들의 집합이 결과 집합(result set)이다.
- 결과 집합에서 사용자는 커서를 사용하여 한 번에 한 레코드씩 데이터에 접근할 수 있다



데이터베이스 사용하기

- SQLite는 정식 데이터베이스로서 어떤 형태의 데이터도 효율적으로 저장할 수 있다. 앱에서는 SQL 명령어를 이용하여 데이터베이스에 데이터를 저장하거나 원하는 조건의 데이터를 가져올 수 있다.

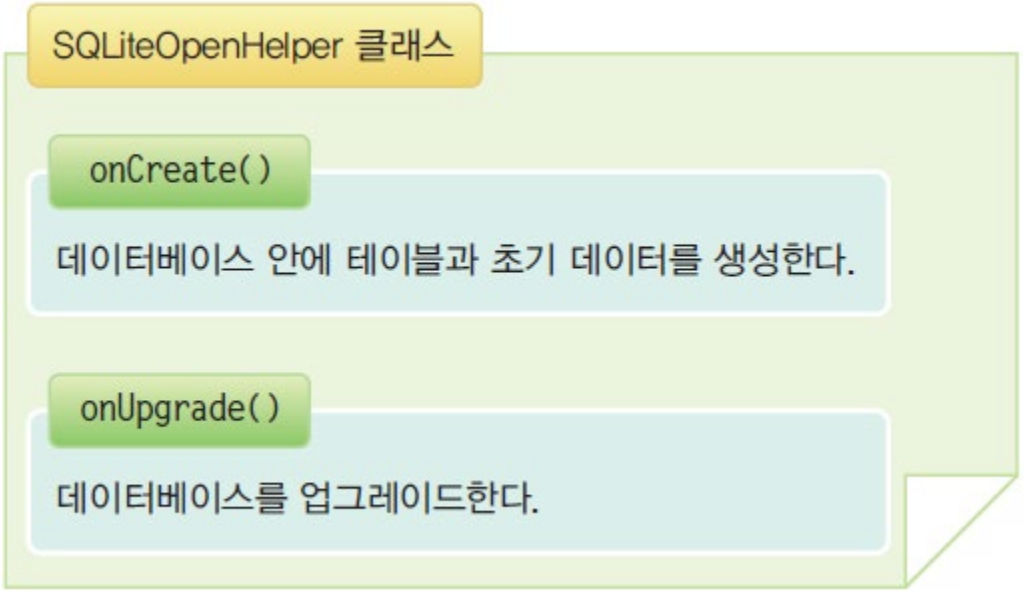


데이터베이스를 사용하는 2가지 방법

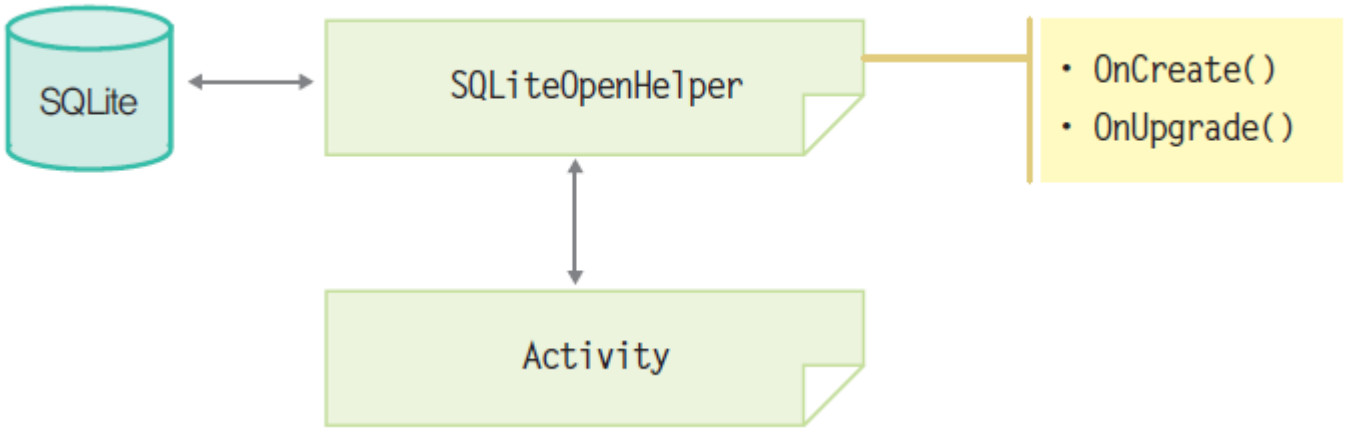
- SQLiteOpenHelper를 사용하는 방법
- openOrCreateDatabase( ) 메소드로 데이터베이스 객체를 직접 생성하는 방법

SqLiteOpenHelper 클래스

- SQLiteOpenHelper 클래스는 데이터베이스를 감싸고 있는 도우미 클래스이다. SQLiteOpen-Helper 클래스를 사용하면 데이터베이스가 생성되거나 버전이 업그레이드될 때에 호출되는 콜백 메소드를 개발자가 정의하여서 적절한 처리를 할 수 있다.



SQLiteOpenHelper 클래스



SqLiteOpenHelper 클래스

· SQLiteOpenHelper(Context context, String name, SQLiteDatabase. CursorFactory factory, int version)

SQLiteOpenHelper 생성자이다. 매개변수 context에는 데이터베이스를 생성하는 액티비티를 전달한다. name은 데이터베이스 파일 이름이다. factory는 커서를 지정하는 매개변수로서 null을 전달하면 표준 커서가 사용된다. version은 데이터베이스의 버전이다.

전체 구조

```
class DBHelper extends SQLiteOpenHelper {

  public dbHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
  }

@Override
  public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE contacts ( _id INTEGER PRIMARY KEY
        AUTOINCREMENT, name TEXT, tel TEXT)");
  }
@Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

SQLiteOpenHelper 클래스의 onCreate() 메소드를 재정의해준다. 데이터베이스가 처음으로 생성될 때에 호출된다. SQL 명령어들을 이용하여서 데이터베이스의 테이블을 생성하고 초기화하면 된다.

SqLiteOpenHelper 클래스

```
        db.execSQL("DROP TABLE IF EXISTS contact");
        onCreate(db);
    }
}
```

onUpgrade()는 데이터베이스가 업그레이드될 필요가 있을 때 호출된다. 데이터베이스의 버전이 올라가면 호출되므로 기존의 테이블을 삭제하고 새로 만들어주면 된다. 일반적으로는 기존의 테이블을 버리고 새로운 테이블을 생성한다.

SQL을 사용하여서 데이터 추가, 삭제, 쿼리하기

메소드	설명
`void execSQL(String sql)`	SELECT 명령을 제외한 모든 SQL 문장을 실행한다. 예를 들어서 CREATE TABLE, DELETE, INSERT 등을 실행한다.
`Cursor rawQuery (String sql, String[] selectionArgs)`	SELECT 명령어를 사용하여 쿼리를 실행하려면 rawQuery()를 사용하면 된다. 쿼리의 결과는 Cursor 객체로 반환된다. Cursor 객체는 쿼리에 의하여 생성된 행들을 가리키고 있다. Cursor는 데이터베이스에서 결과를 순회하고 데이터를 읽는데 사용되는 표준적인 메커니즘이다.

예제 데이터베이스

- 예를 들어서 다음과 같은 데이터베이스 테이블을 가정하자.

ID	NAME	AGE	ADDRESS	SALARY
1	Kim	32	Busan	2000.00
2	Park	25	Seoul	1500.00

SQL을 사용하여서 데이터 추가, 삭제, 쿼리하기

```
SQLiteDatabase db = helper.getWritableDatabase();
db.execSQL("INSERT INTO CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)
      VALUES (1, 'Hong', 32, 'Seoul, Korea', 2000.00);")
```

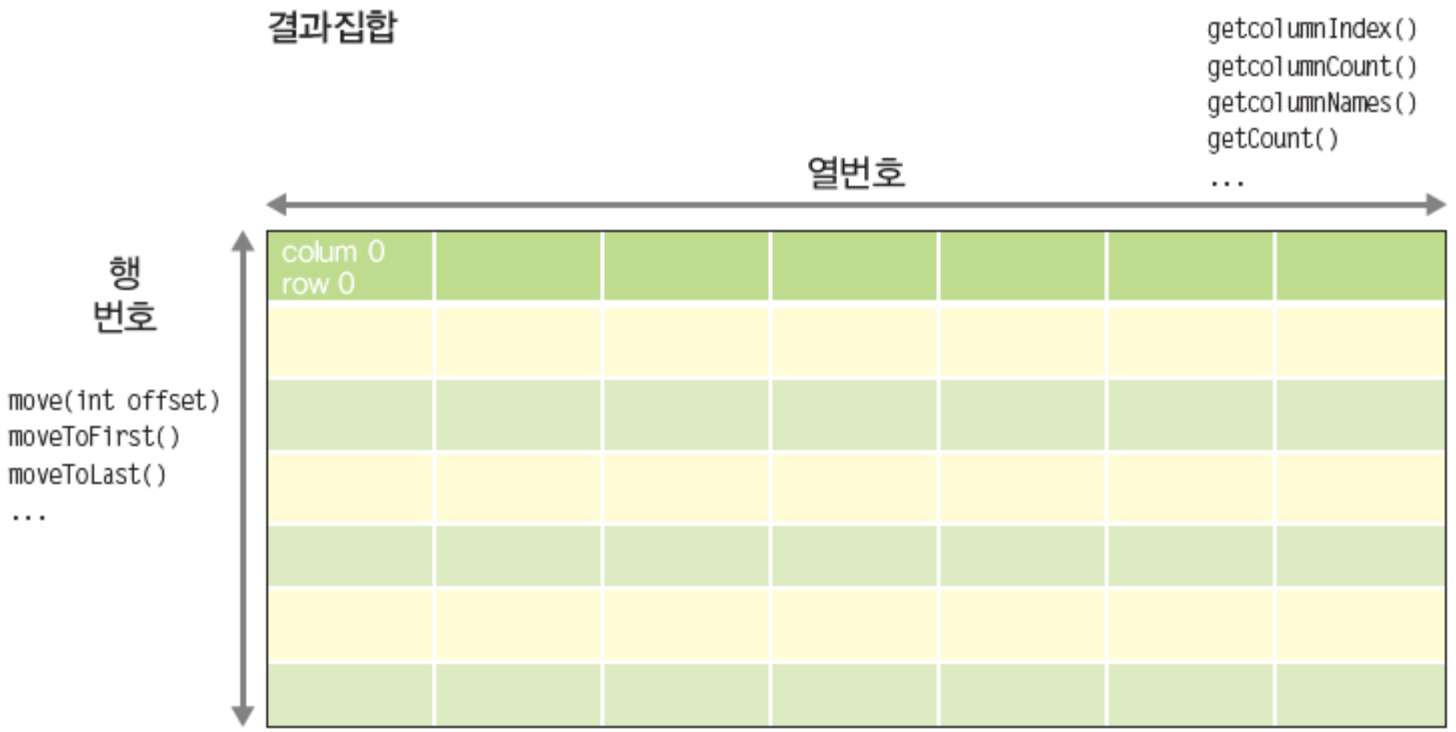
```
SQLiteDatabase db = helper.getWritableDatabase();
Cursor cursor = db.rawQuery("SELECT ID, NAME, SALARY FROM CUSTOMERS", null);
```

```
SQLiteDatabase db = helper.getWritableDatabase();
Cursor cursor = db.rawQuery("SELECT ID, NAME, SALARY
         FROM CUSTOMERS WHERE SALARY > 2000;", null);
```

SQL을 사용하여서 데이터 추가, 삭제, 쿼리하기

```
// 쿼리의 결과가 커서로 반환된다.
Cursor cursor = sqliteDB.rawQuery("SELECT NAME FROM CUSTOMERS", null);
// 만약 커서가 결과를 가지고 있으면
if (cursor != null) {
  // 커서를 첫 번째 레코드로 이동한다.
  if (cursor.moveToFirst()) {
      do {
          // 커서로부터 이름을 얻는다.
          String name = cursor.getString(cursor.getColumnIndex("NAME"));
          // 이름을 ArrayList인 list에 추가한다.
          list.add(name);
          // 다음 레코드로 간다.
      } while (cursor.moveToNext());
  }
}
```

결과 집합 메소드

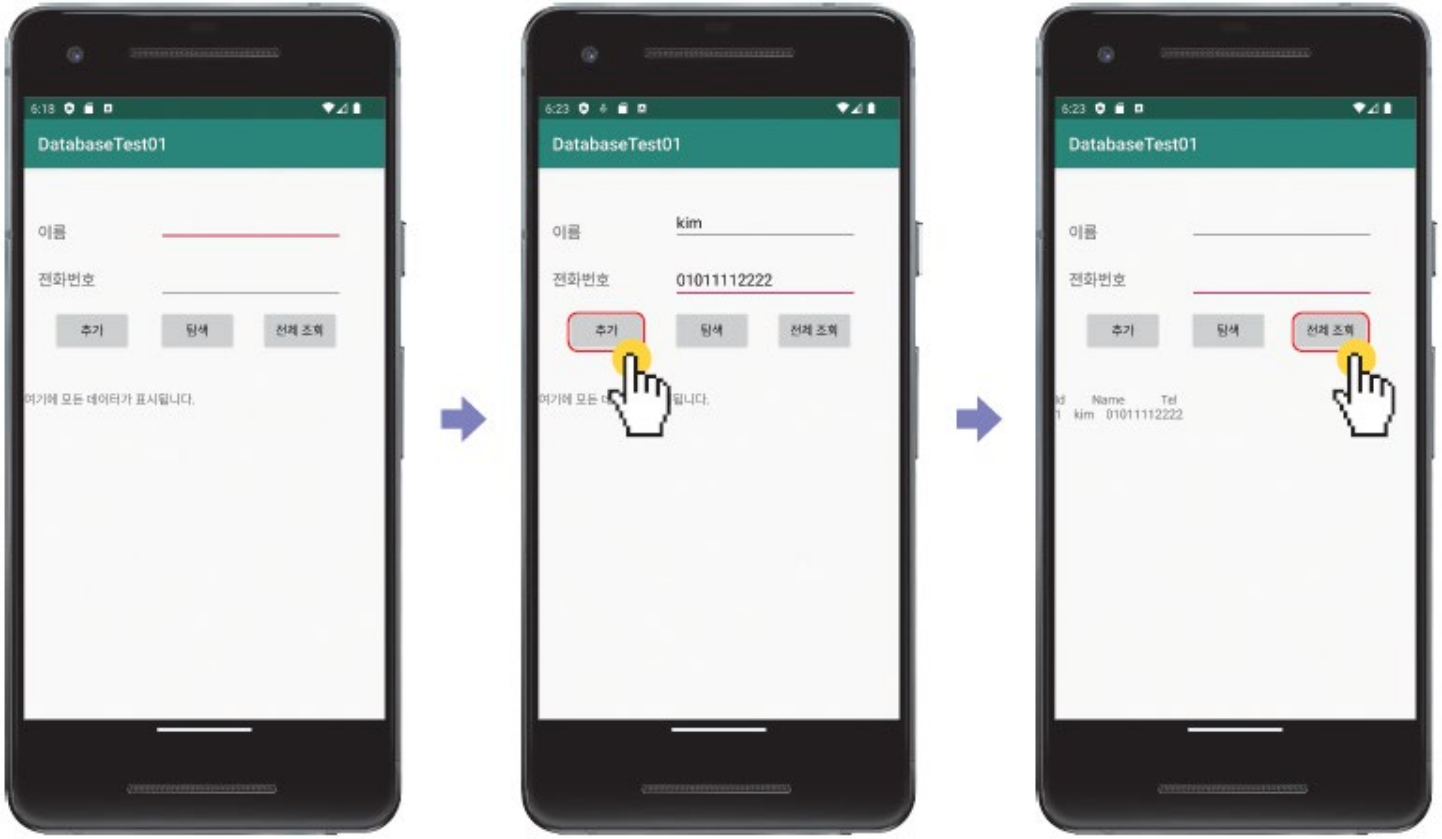

결과집합
getcolumnIndex()
getcolumnCount()
getcolumnNames()
getCount()
...
열번호
행
번호
move(int offset)
moveToFirst()
moveToLast()
...
colum 0
row 0

전용 메소드를 사용하여서 데이터 추가, 삭제, 쿼리하기

메소드	설명
`Cursor query(boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)`	
	query()는 쿼리를 실행하고 Cursor 객체를 반환한다. distinct : 만약 각 행이 유일하면 true이다. table : 쿼리의 대상이 되는 테이블이다. columns : 어떤 컬럼을 반환할 것인지를 결정한다. null은 모든 컬럼을 의미한다. selection : SQL WHERE에 해당되는 필터이다. null은 모든 행을 의미한다. selectionArgs : selection의 ?를 순서대로 대체한다. groupBy : SQL GROUP BY 절에 해당하는 필터이다. having : SQL HAVING 절에 해당하는 필터이다. orderBy : SQL ORDER BY 절에 해당하는 필터이다. limit : 반환되는 행의 개수를 제한한다.
`long insert(String table, String nullColumnHack, ContentValues values)`	
	table : 행을 추가하는 테이블 nullColumnHack : 만약 null이 아니면 NULL 값을 삽입하는 컬럼의 이름이 된다. values : 삽입되는 값이다.
`int delete(String table, String whereClause, String[] whereArgs)`	
	데이터베이스에서 조건에 맞는 행을 삭제한다.
`int update(String table, ContentValues values, String whereClause, String[] whereArgs)`	
	데이터베이스에서 조건에 맞는 행을 갱신한다.

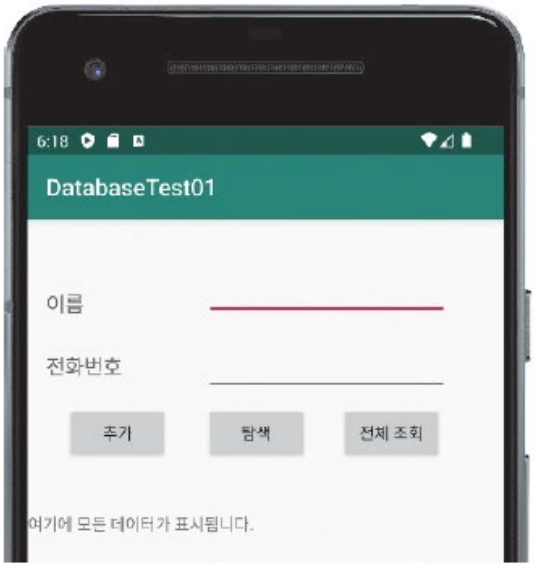
예제: 연락처 저장 앱

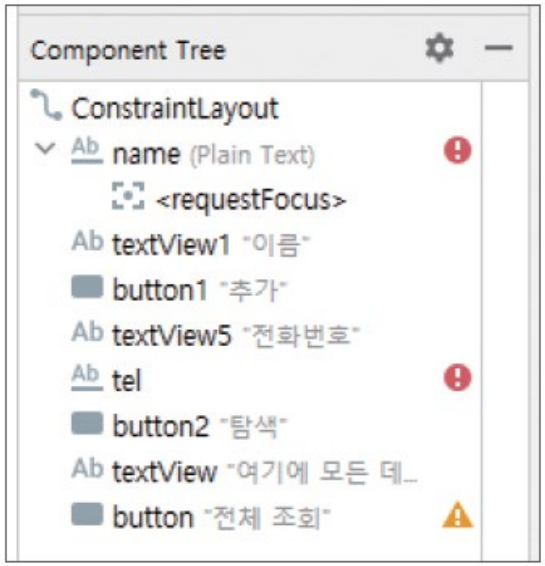
- 연락처를 저장하고 검색하는 간단한 애플리케이션를 작성하여 보자.



사용자 인터베이스 작성

- 다음과 같은 사용자 인터페이스를 작성한다.





코드 작성

DatabaseTest01Activity.java


```
package com.example.hello;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

class DBHelper extends SQLiteOpenHelper {
  private static final String DATABASE_NAME = "mycontacts.db";
  private static final int DATABASE_VERSION = 1;

  public DBHelper(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
  }

  public void onCreate(SQLiteDatabase db) {
      db.execSQL("CREATE TABLE contacts ( _id INTEGER PRIMARY KEY"+
                 " AUTOINCREMENT, name TEXT, tel TEXT);");
  }
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
      db.execSQL("DROP TABLE IF EXISTS contacts");
      onCreate(db);
  }
}
```

먼저 SQLiteOpenHelper 클래스를 상속받은 dbHelper 클래스가 정의되어 있다. 데이터베이스 파일 이름은 "mycontacts.db"가 되고 데이터베이스 버전은 1로 되어 있다. 만약 데이터베이스가 요청되었는데 데이터베이스가 없으면 onCreate()를 호출하여 데이터베이스 파일을 생성해준다.

onCreate()에서는 CREATE TABLE 명령어를 이용하여 데이터베이스 안에 테이블을 생성하여 준다. _id는 키필드인데 자동 증가된다는 것이 기술되어 있다. 이어서 name과 tel이라는 필드를 TEXT 형태로 정의하였다. 물론 _id 필드는 반드시 필요한 것은 아니지만, 데이터베이스에는 키 필드가 있어야 실행되는 연산들도 있다. 또 컨텐츠 공급자와 연결할 때도 키 필드는 필요하다. 따라서 가급적 포함시키는 것이 바람직하다.

코드 작성

```
public class DatabaseTest01Activity extends AppCompatActivity {
  DBHelper helper;
  SQLiteDatabase db;
  EditText edit_name, edit_tel;
  TextView edit_result;

  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      helper = new DBHelper(this);
      try {
          db = helper.getWritableDatabase();
      } catch (SQLiteException ex) {
          db = helper.getReadableDatabase();
      }
      edit_name = (EditText) findViewById(R.id.name);
      edit_tel = (EditText) findViewById(R.id.tel);
      edit_result = (TextView) findViewById(R.id.textView);
  }
```

onUpgrade()는 데이터베이스의 버전이 증가되었을 때 호출된다. 여러 가지 필요한 업그레이드 동작은 수행하는 것이 좋지만, 여기서는 무조건 기존의 테이블을 버리고 단순히 새로운 테이블을 다시 정의하였다.

액티비티의 onCreate()에서는 dbHelper 객체를 생성한다. 이어서 데이터베이스 객체를 얻기 위하여 getWritableDatabase()를 호출한다.

"추가" 버튼은 레코드를 추가할 때 사용하고 "탐색" 버튼은 이름을 가지고 데이터베이스에서 레코드를 검색할 때 사용된다.

"추가" 버튼이 눌려졌을 경우에 실행되는 코드를 살펴보자. 먼저 2개의 에디트 텍스트에서 문자열을 가져온다.

다음에는 데이터베이스에 레코드를 추가하는 SQL 문장을 작성하여 실행시키면 된다. 여기서는 간단히 데이터베이스 객체가 제공하는 execSQL() 메소드를 호출하였다.

SQL 문장 안에서 문자열을 만들 때는 "..."가 아닌 '...'을 사용한다는 것에 유의하자. 위의 문장이 실행되면 테이블에 새로운 레코드가 추가된다.

코드 작성

```
public void insert(View target) {
    String name = edit_name.getText().toString();
    String tel = edit_tel.getText().toString();
    db.execSQL("INSERT INTO contacts VALUES (null, '" + name + "', '" + tel
            + "');");
    Toast.makeText(getApplicationContext(), "성공적으로 추가되었음",
            Toast.LENGTH_SHORT).show();
    edit_name.setText("");
    edit_tel.setText("");
}
```

```
public void search(View target) {
    String name = edit_name.getText().toString();
    Cursor cursor;
    cursor = db.rawQuery("SELECT name, tel FROM contacts WHERE name='"
            + name + "';", null);
    while (cursor.moveToNext()) {
        String tel = cursor.getString(1);
        edit_tel.setText(tel);
    }
}
```

코드 작성

```
public void select_all(View target) {
      Cursor cursor;
         cursor = db.rawQuery("SELECT * FROM contacts", null);

      String s="Id          Name              Tel \r\n";
      while (cursor.moveToNext()) {
         s += cursor.getString(0) + "     ";
         s += cursor.getString(1) + "     ";
         s += cursor.getString(2) + "     \r\n";
      }
      edit_result.setText(s);

  }
```

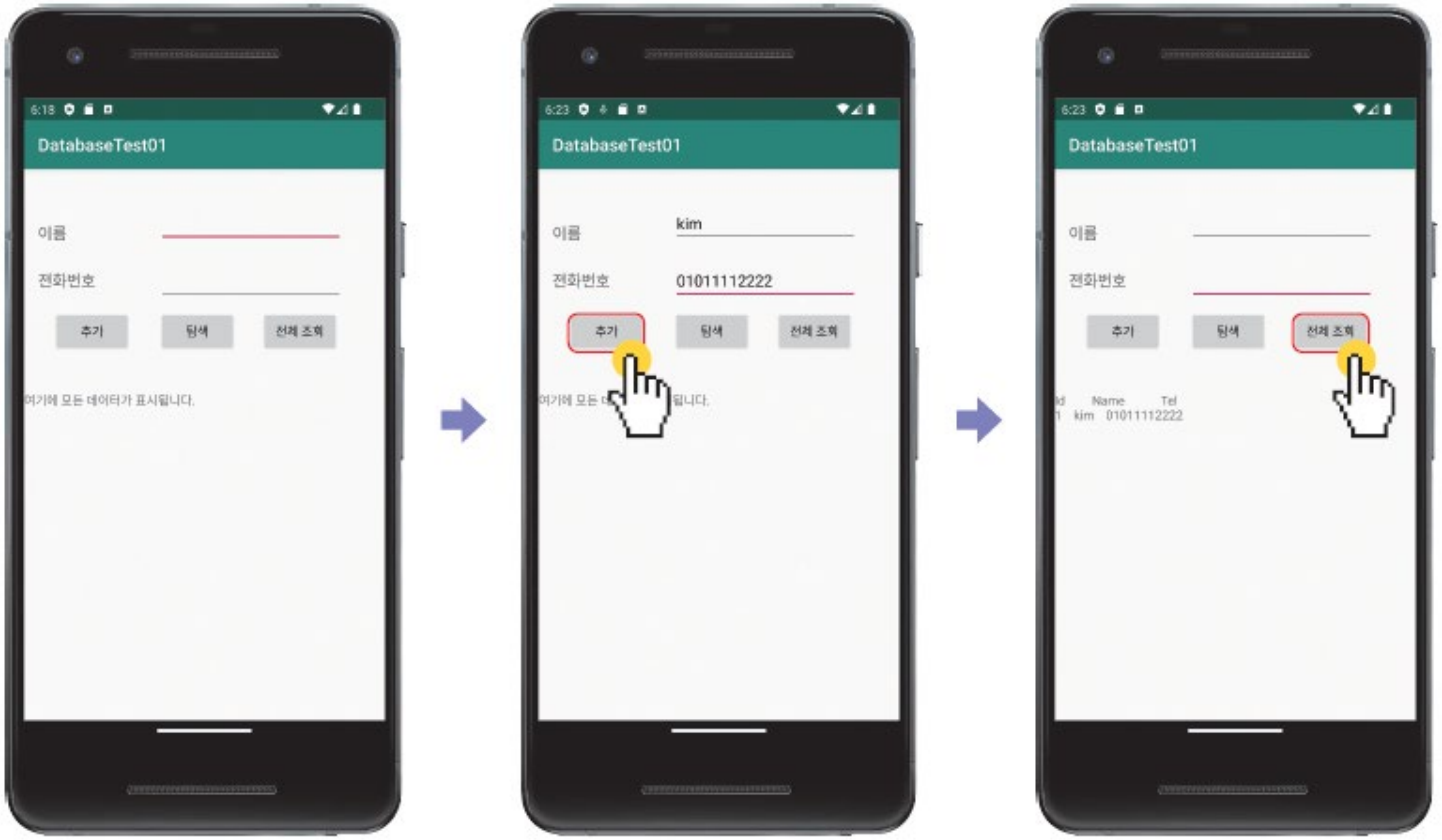
"search" 버튼이 눌려졌을 경우에 실행되는 코드를 살펴보자. 먼저 에디트 텍스트 "name"에서 찾고자 하는 사람의 이름을 가져온다.

다음으로 Cursor 참조 변수가 선언된다. Cursor 클래스는 쿼리의 결과로 생성되는 레코드의 집합을 순회하면서 데이터를 가져올 수 있는 클래스이다.

SQL에서 쿼리를 실행하는 명령어는 SELECT이다. 따라서 찾고자 하는 사람의 이름을 넣어서 SELECT 문장을 작성하고 이것을 rawQuery() 메소드로 실행시킨다.

검색 조건을 주고 싶으면 SELECT 문장에 WHERE 절을 추가하면 된다.

실행 결과

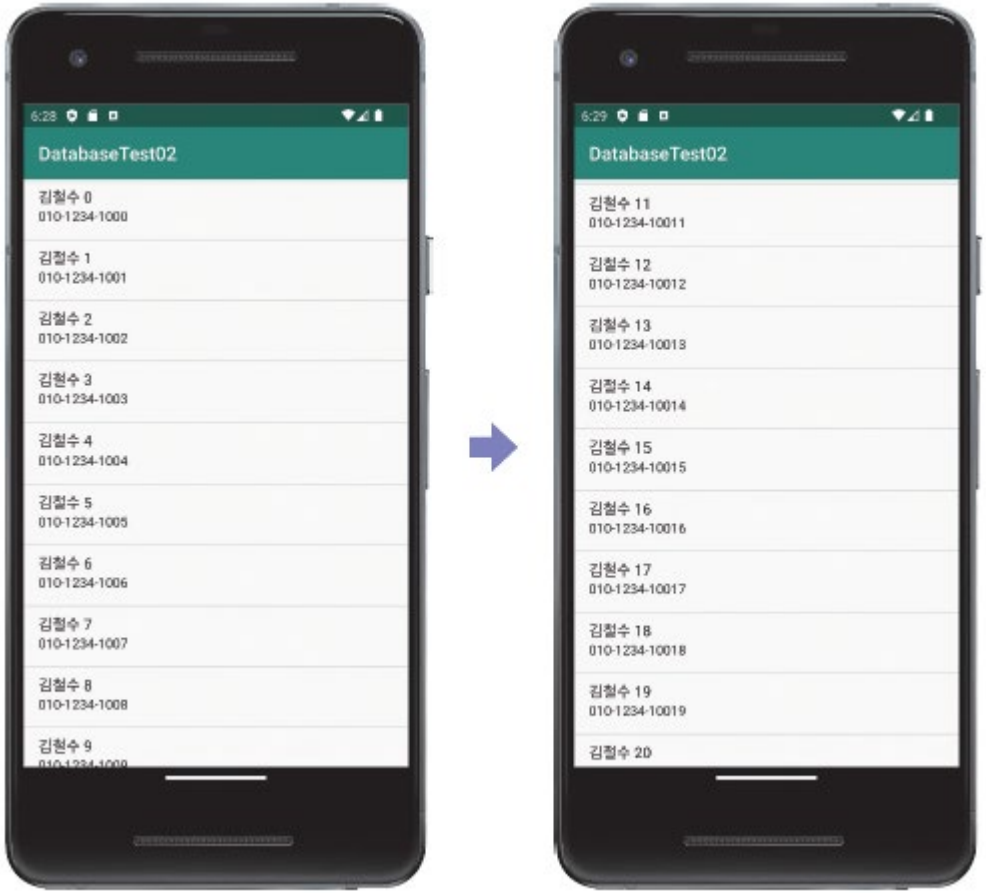

DatabaseTest01
이름
전화번호
추가
탐색
전체 조회
여기에 모든 데이터가 표시됩니다.
kim
01011112222
Name
Tel
1 kim 01011112222

데이터베이스와 어댑터

- SimpleCursorAdapter 객체는 데이터베이스와 화면을 연결한다.
- 데이터베이스에서 데이터를 읽어서 정해진 레이아웃으로 화면에 표시한다.

메소드	설명
`public` SimpleCursorAdapter(Context context, int layout, Cursor c, String[ ] from, int[ ] to)	
	layout은 데이터를 표시할 레이아웃 리소스를 가리킨다. c는 커서 객체이다. from은 문자열 배열로 화면에 표시하고 싶은 컬럼의 이름이다. to는 각 from 안의 컬럼이 표시되는 뷰들의 리스트이다. to 안에 들어 있는 리소스들은 모두 텍스트 뷰이어야 한다.

예제: 커서 어댑터를 이용한 예제



코드 작성

MainActivity.java

```
package kr.co.company.databasetest02;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

class dbHelper extends SQLiteOpenHelper {
  private static final String DATABASE_NAME = "mycontacts.db";
  private static final int DATABASE_VERSION = 1;

  public dbHelper(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
  }

  public void onCreate(SQLiteDatabase db) {
      db.execSQL("CREATE TABLE contacts ( _id INTEGER PRIMARY KEY AUTOINCREMENT, "
                  + "name TEXT, tel TEXT);");
      for (int i = 0; i < 10; i++) {
          db.execSQL("INSERT INTO contacts VALUES (null, " + "'김철수 " + i
                  + "'" + ", '010-1234-100" + i + "');");
      }
  }
```

이 애플리케이션에서는 dbHelper 클래스를 이용하여서 mycontacts.db라는 이름의 데이터베이스 파일을 생성한다. onCreate()에서 테이블을 생성한 후에, 반복 루프를 수행하면서 (김철수0, 010-1234-1000) ... (김철수9, 010-1234-1009)까지의 샘플 레코드를 테이블에 추가한다.

코드 작성

```
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS contacts");
        onCreate(db);
    }
}

public class MainActivity extends AppCompatActivity {
    dbHelper helper;
    SQLiteDatabase db;
    EditText edit_name, edit_tel;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

코드 작성

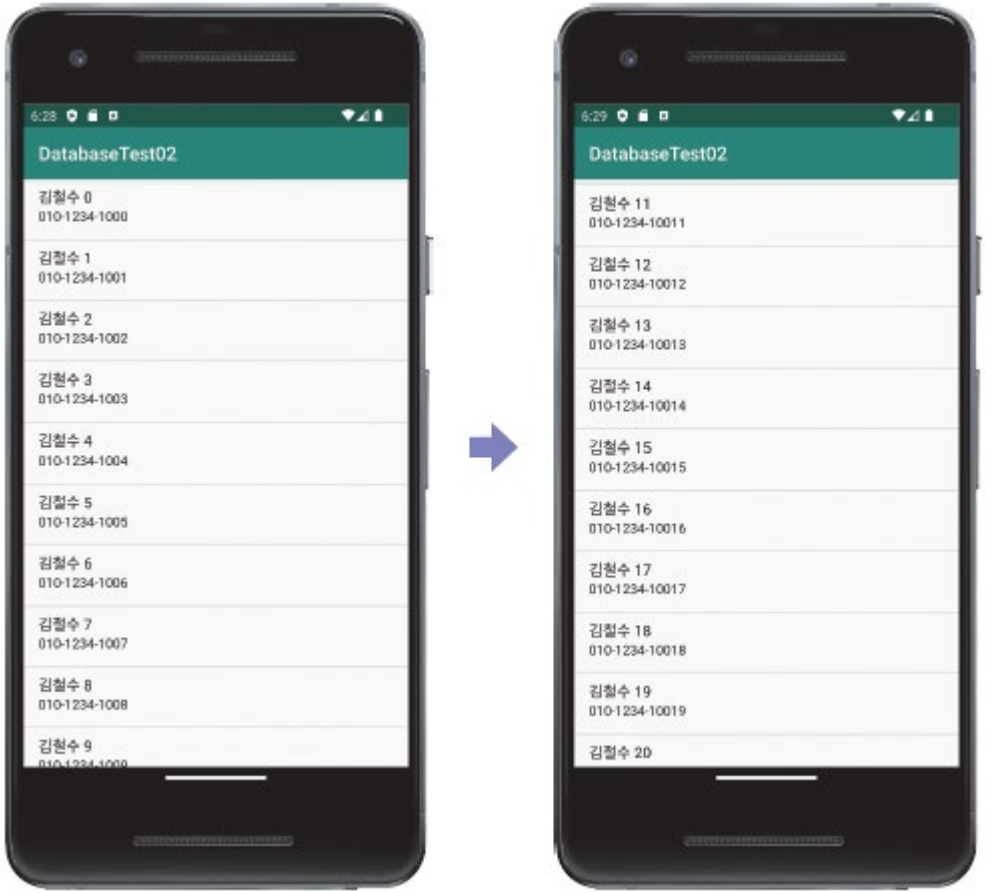
```
        helper = new dbHelper(this);
        db = helper.getWritableDatabase();
        Cursor cursor = db.rawQuery("SELECT * FROM contacts", null);
        startManagingCursor(cursor);

        String[] from = { "name", "tel" };
        int[] to = { android.R.id.text1, android.R.id.text2 };
        SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
                android.R.layout.simple_list_item_2, cursor, from, to);
        ListView list = (ListView) findViewById(R.id.list);
        list.setAdapter(adapter);
    }
}
```

액티비티에서는 dbHelper 객체를 생성하고 SELECT 명령어를 이용하여서 테이블의 모든 레코드를 커서 객체로 가져온다.

startManagingCursor(cursor) 문장은 액티비티가 커서 객체를 관리하도록 한다. 즉 액티비티의 생애주기와 커서의 생애주기를 일치시키는 것이다.

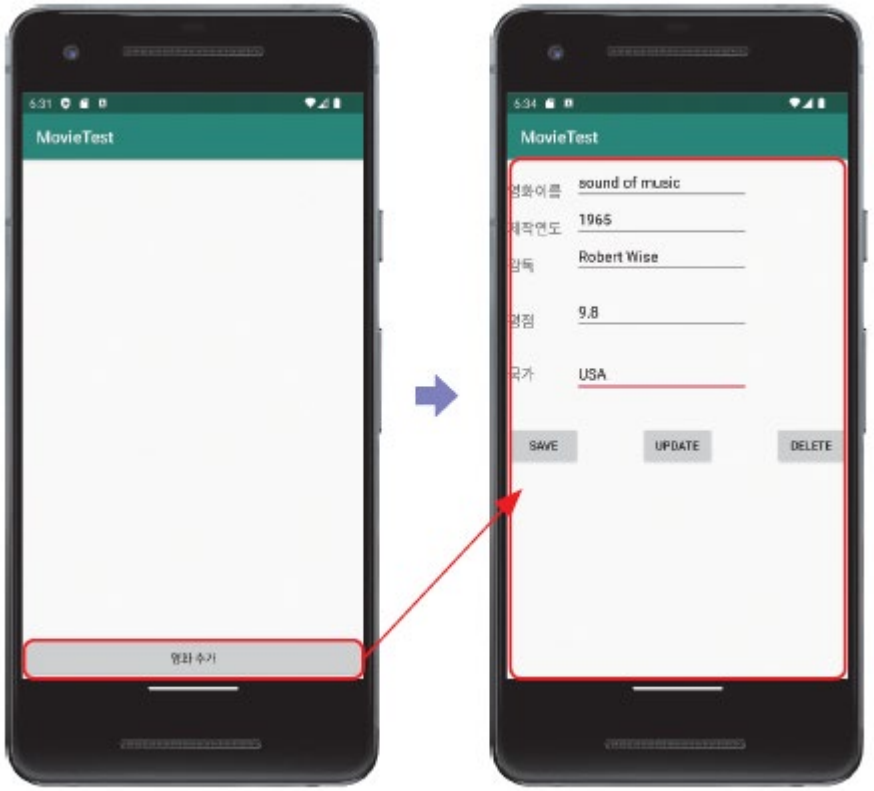
android.R.layout.simple_list_item_2라는 표준 레이아웃을 사용하는 커서 어댑터 객체를 생성한다. 이 표준 레이아웃은 2개의 텍스트 뷰를 가지고 있으며, 첫 번째 텍스트 뷰(android.R.id.text1)는 "name" 필드를 표시하고 두 번째 텍스트 뷰(android.R.id.text2)는 "tel" 필드를 표시하도록 연결한다.

실행 결과

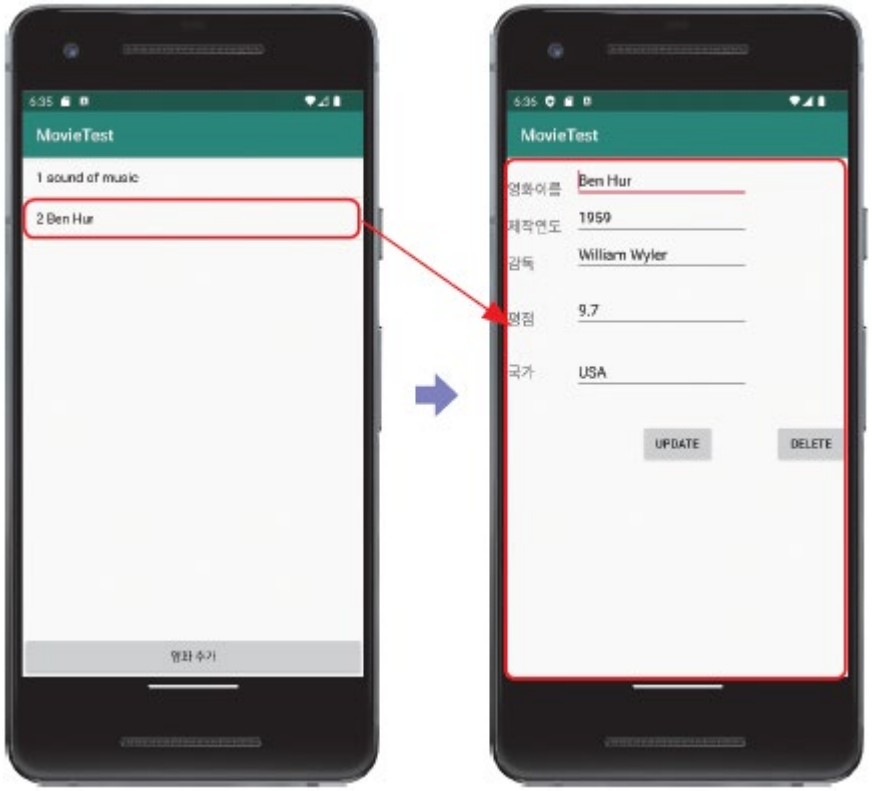

6:28
DatabaseTest02
김철수 0
010-1234-1000
김철수 1
010-1234-1001
김철수 2
010-1234-1002
김철수 3
010-1234-1003
김철수 4
010-1234-1004
김철수 5
010-1234-1005
김철수 6
010-1234-1006
김철수 7
010-1234-1007
김철수 8
010-1234-1008
김철수 9
6:29
DatabaseTest02
김철수 11
010-1234-10011
김철수 12
010-1234-10012
김철수 13
010-1234-10013
김철수 14
010-1234-10014
김철수 15
010-1234-10015
김철수 16
010-1234-10016
김철수 17
010-1234-10017
김철수 18
010-1234-10018
김철수 19
010-1234-10019
김철수 20

Coding Challenge:
영화 데이터베이스 만들기

- 이번 장에서 학습한 데이터베이스 기능을 이용하여서 다음과 같은 기능을 하는 애플리케이션을 작성하여 보자. 사용자가 영화 정보를 입력하면 이것을 데이터베이스에 저장하고 리스트 뷰를 사용하여서 화면에 나열한다.



Coding Challenge:
영화 데이터베이스 만들기



Q & A