

안드로이드 조수연쌤

프린트

빠진 부분

7 RecyclerView

11 Service

12 FirebaseCloudMessaging

14 CustomWidget

1.개발환경설정

JDK의 종류

1. OpenJDK

오픈소스 프로젝트이다

2. Oracle JavaSE

공개버전은 사용해도 되나 2019년 이후에는 기술지원을 받기 위해서는 비용을 지불해야 함

2.Layout

Activity

Activity의 코드는 *.java 파일에서 작성해야 하며 Activity에는 반드시 하나의 layout을 지정해 사용해야 한다.

- 폴더 내에 사용하지 않는 xml파일은 있어도 프로젝트 파일 내에 에러가 있으면 컴파일이 안됨.
- 파일 이름에는 영어 소문자와 '_'만 가능하다.

Activity파일에서 layout을 사용하는 방법

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

Layout

Layout은 트리 구조로 만들어지고 최상위는 반드시 Layout이어야 하지만 내부에는 위젯 또는 또 다른 레이아웃을 포함할 수 있다.

- 추가 라이브러리를 추가하여 레이아웃을 사용할 수 있다.
- 레이아웃의 깊이가 깊어질 경우 성능이 떨어진다

Relative Layout / Constraint Layout 등을 이용하여 깊이를 줄이는 것이 좋다.

레이아웃의 기본 구조

Left vs Start

초기 버전의 안드로이드에서는 `Left/Right/Top/Bottom` 을 사용했지만 최근에 `Start/End` 로 업데이트 되었다.

- 낮은 버전의 minimum SDK로 프로젝트를 생성할 경우 `Left` 와 `Start` 속성을 모두 적어 줘야 정상동작한다.
- 높은 버전은 `Start` 나 `Left` 하나만 적어도 동작한다.

Layout의 width/height

위젯 또는 레이아웃의 크기이고 반드시 지정해 줘야 하는 속성이다.

- `match_parent` : 부모의 사이즈.
- `wrap_content` : 위젯의 크기에 맞게, 내용물의 사이즈 만큼만.

DP

안드로이드에는 기기가 많아서 화면의 사이즈도 다양하다.

pixel을 사용하면 해상도가 바뀌면 사용자가 보는 사이즈가 바뀌게 되어

안드로이드에서 **dp**라는 개념을 도입하였다.

dp로 사이즈를 설정하면 안드로이드 OS에서 다양한 핸드폰들의 Widget이 비슷한 사이즈가 나오도록 출력해 준다.

Padding

자신의 경계에서 내부 요소 사이의 여백

Margin

다른 위젯과 자신 사이의 여백

Padding과 Margin은 상/하/좌/우 각각 지정 가능하다.

Gravity

내부 요소의 정렬

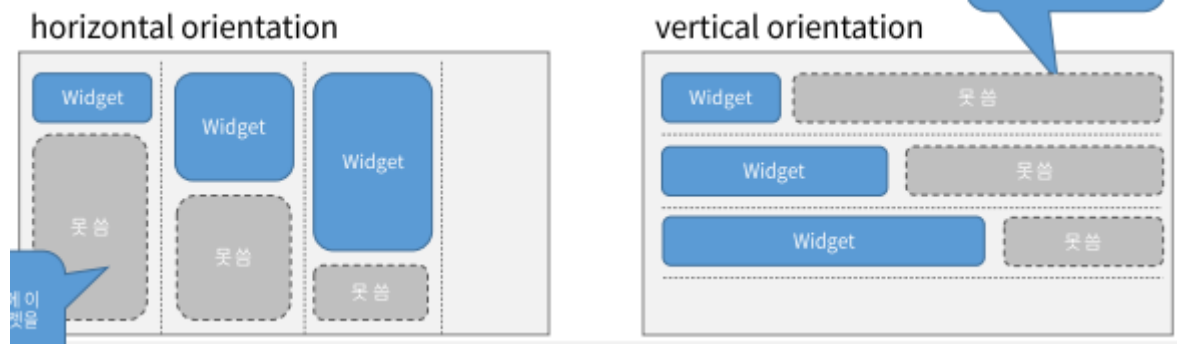
Layout Gravity

Layout 내에서 자신의 위치를 정의

일부 Layout에서만 사용

대표적인 Layout

- Linear Layout한 방향으로 위젯을 배치 **orientation** 속성을 반드시 지정해야 한다.



위젯이 있기 때문에 '못 씬'공간에는 다른 위젯을 둘 수 없다.

Linear Layout 사용 예

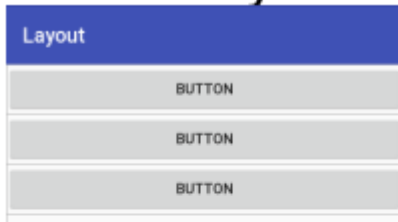
```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
</LinearLayout>

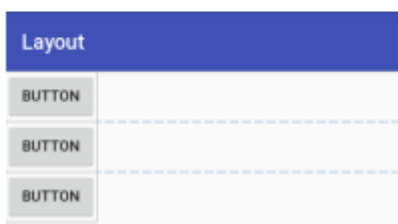
```

Button의 `layout_width` 가 `match_parent` 일 때와 `wrap_content` 일때의 차이

Linear Layout



`width="match_parent"`



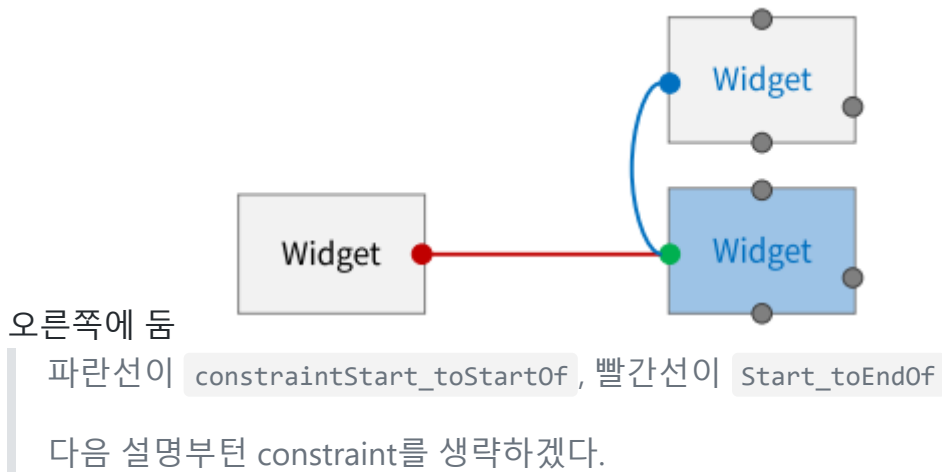
`width="wrap_content"`

- Relative Layout

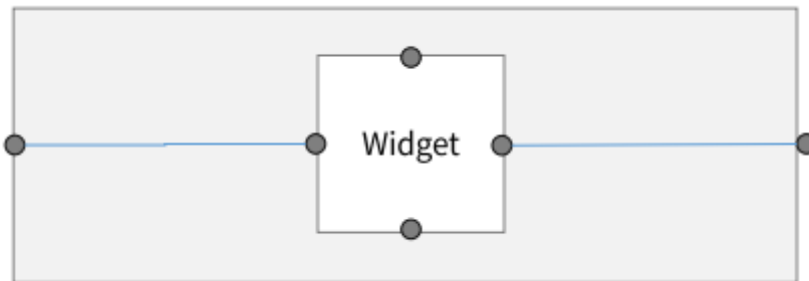
부모 또는 다른 위젯을 기준 삼아 상대적인 위치를 지정하여 배치

- Constraint Layout 각 위젯에 위치에 대한 제약을 지정하여 배치하는 레이아웃 ※ xml namespace로 `android:` 대신 `app:` 을 사용함 `constraintStart_toStartOf` : 자신의 왼쪽과

다른 위젯의 왼쪽을 동일하게 정렬 `constraintStart_toEndOf` : 자신의 왼쪽을 다른 위젯의



각각의 constraint는 중력(스프링)처럼 작동하여 `start_toStartOf` 와 `End_toEndOf` 를 동시에 `parent` 로 설정하면 중간으로 이동함.



- bias: 좌/우 동시에 제약이 걸리면 (위 상황처럼) bias를 조절하면 당겨지는 힘의 비율을 조절할 수 있다.

```
app:layout_constraintHorizontal_bias="0.3"
```

이렇게 하면 위젯이 약간 왼쪽로 이동한다.

3. Widget

TextView

문자열을 출력하는 위젯

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"/>
```

TextView 주요 속성

- textSize: 텍스트 크기
- textColor: 텍스트 색

- `textAlignment`: 텍스트 정렬 방식.
- `visibility`: 보이는 여부
- `clickable`: 클릭 이벤트를 설정하는지 여부

Button

`TextView`를 상속받은 위젯

클릭 가능하며 클릭 리스너를 등록하면 사용자의 이벤트를 처리 할 수 있다.

Listener

사용자가 버튼을 누른 것을 어떻게 알까?

- Polling: 일정 간격을 두고 주기적으로 버튼의 상태를 확인
 - 주로 하드웨어에서 사용
- Listener: 특정 동작이나 메시지, 이벤트에 대한 콜백 등록
 - 주로 소프트웨어에서 사용

Button 주요 속성

- `text`: 버튼에 적힐 label

안드로이드에서 Listener를 등록하는 방법은

1. Activity가 interface를 직접 구현

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    {
        public void onCreate... {
            ...
            // 버튼의 리스너로 자기 자신을 지정
            button.setOnClickListener(this);
        }
        @Override
        public void onClick(View view) {
        }
    }
}
```

2. interface의 익명 객체를 만들어서 사용

```
// ...
// 일반 자바 문법
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
    }
});
// 람다식
button.setOnClickListener((view) => textView.setText("hello"))
```

인자값이 하나일 때는 괄호 생략 가능. body가 한줄일 경우 중괄호 생략 가능 (Javascript와 같다)

익명 객체(람다식)를 사용하기 위해 해야할 자바 버전 설정

```
android {
    // 나머지
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_11
        targetCompatibility JavaVersion.VERSION_11
    }
}
```

- 장점: 로그 출력, 메시지 출력 등의 단순 코드인 경우 간단하게 구현 가능
- 단점: 해당 객체를 다른 위젯이 사용할 수 없다.

3. interface 변수를 사용하는 방법

```
// 클래스 내
private View.OnClickListener buttonListener = new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        textView.setText("Listener!");
    }
};
// 생성자 내
button.setOnClickListener(buttonListener);
```

- 장점: 다른 위젯에서도 리스너로 등록 가능
- 단점: Activity에 속한 메소드가 아니기 때문에 다른 위젯이나 메소드 접근에 제약이 있을 수 있다. (이문제는 익명 객체도 있다.)

4. 메소드를 레이아웃에서 지정하는 방법

```
public void customListener(View v) { // 함수명은 자유롭게 가능
}
```

```
<Button
...
android:onClick="customListener" <!--함수명과 같게-->
```

- 장점: 버튼 별로 메소드를 추가하고 지정하기 쉽다
 - 단점: 리스너를 확인하기 위해 xml파일까지 검토해야 한다.
- 클릭 이외의 이벤트에는 사용하기 어렵다.

EditText

사용자로부터 문자열을 입력 받을 수 있는 위젯

EditText 주요 속성

- inputType: textPassword 등 여러 속성이 있음
- ems

기존에 width가 wrap_content인 경우 글자를 하나도 입력하지 않으면 width가 거의 없이 출력이 됨.

그 때, 최소한의 너비를 확보하기 위해서 사용 (ems값 기준으로 너비를 계산함)

Toast

```
Toast.makeText(Context, "출력할 문자열", Toast.LENGTH_LONG).show();
```

##

4. Widget2

ToggleButton

Checked/Unchecked 두 가지 상태를 가지며 클릭할 때마다 상태가 바뀌는 버튼

```
public void onCheckedChange(CompoundButton btn, boolean isChecked)
```

이벤트가 존재한다. 사용은 기존 리스너처럼 사용.

CheckBox

UI의 출력 형태만 다를 뿐 내부적인 동작은 Toggle Button과 동일하다

Toggle Button과 같은 리스너를 사용한다.

RadioButton

UI에 출력되는 모습은 CheckBox와 비슷하지만 실제 동작은 Button과 유사하다.

클릭하면 선택되지만 다시 클릭한다고 해서 선택이 해제되지 않는다.

Button과 같은 `onClickListener`를 사용함

RadioGroup의 자식으로 배치해야 동시 선택이 안된다

```
<RadioGroup ...>
  <RadioButton ...
    android:text="One"/>
  <RadioButton ...
    android:text="Two"/>
  <RadioButton ...
    android:text="Three"/>
</RadioGroup>
```

이렇게 사용해야 1또는 2로 선택이 된다.

Progress Bar



진척도를 나타내는 위젯

Bar타입과 원형 타입이 있다.

- Bar: 전체 단계 중 진척 단계를 표현

- 원형: 동그라미 애니메이션을 하는 위젯

Bar 타입

```
<ProgressBar
    android:id="@+id/progressBar"
    style="?android:attr/progressBarStyleHorizontal" <!--이부분-->

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

원형 타입

```
<ProgressBar
    android:id="@+id/progressBar"
    style="?android:attr/progressBarStyle" <!--이부분-->
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Bar 타입의 메소드

- `incrementProgressBy` : 현재 값에 해당 수를 더함
- `setProgress` : 주어진 값으로 변경

SeekBar

Android SeekBar



사용자가 드래그하거나 클릭하여 포인팅 지점을 설정 가능한 위젯

연속된 값을 사용하는 타입 (default)와 불연속 값을 사용하는 타입 (discrete)이 있다.

기본 Seek Bar

```
<SeekBar
    ...
    android:max="10" <!-- seekbar max 설정 -->
/>
```

불연속 타입 (discrete)



```
<SeekBar
...
style="@style/Widget.AppCompat.SeekBar.Discrete"
/>
```

이벤트는 `OnSeekBarChangeListener` 이다.

```
@Override
public void onProgressChanged(SearchBar seekBar, int i, boolean b) {
    if(b){ // b는 사용자에게 의해서 값이 바뀌었는지 여부를 나타내는 값이다.

        progressBar2.setProgress(i);    // i는 값
    }
}
```

ImageView

화면에 그림을 출력하는 위젯

출력을 원하는 원본 그림을 속성 값으로 줘야 함

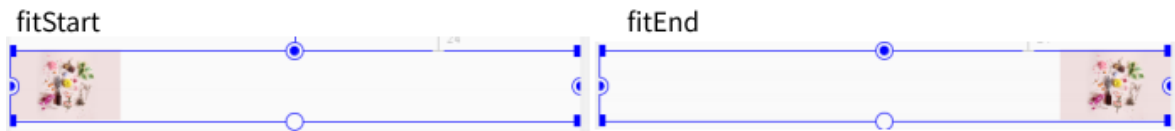
png, jpg를 출력할 수 있으며 SVG파일을 불러와서 사용한 xml파일도 출력 가능하다.

1. res/drawable 폴더에 이미지를 불러옴
2. layout 에서 불러와서 사용 (예시)

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/seekBar2"
    app:srcCompat="@drawable/coffee_icon_64" />
```

- 작은 사이즈의 이미지는 `layout_width/height` 를 수정하여 사이즈 조절
- 큰 사이즈의 이미지는 기본적으로 작게 줄여서 출력이 됨.

`android:scaleType` 으로 이미지를 설정 (ppt 내용)



center : 그림을 원본 사이즈로 만든 상태에서 한 가운데 점 기준으로 widget 사이즈만큼 잘라서 출력.
그림이 작을 경우 widget 영역에 공백이 보임



fitXY : 가로세로 비율을 무시하고 widget 사이즈로 그림을 맞춰 그림



SVG 불러오기는 스킵.

5. Activity Intent

키워드

- Activity: UI가 있는 단일 화면, 하나의 Activity는 하나의 기능을 가짐

하나의 앱에 있어도 각 Activity는 독립적이며, 내 앱이 허용할 경우 다른 앱에서 내 앱의 Activity 중 하나만 골라서 실행 가능.

- Service

백그라운드에서 실행하며 UI를 제공하지 않음

사용하는 경우들

- 오래 걸리는 작업을 실행
- Activity가 꺼진 후에도 수행해야 할 작업

- BroadCast Receiver

시스템이 보내는 broadcast를 수신한다. (즉 시스템이 보내는 메시지를 받음)

예를 들어 배터리 잔량이 부족함, 블루투스 기능이 꺼짐 등이 있다.

- Content Prvider

내 앱이 가진 Content(파일 또는 SQLite에 저장된 데이터)를 다른 앱이 조회하거나 사용할 수 있도록 제공 해 준다. 주로 주소록, 사진 음악 앱에서 제공함.

LifyCycle

1. onCreate()

Activity가 최초로 실행될 때 한 번 호출됨

레이아웃 지정과 딱 한번 초기화할 코드 작성

2. onStart()

Activity가 화면에 출력되는 시점

3. onResume()

사용자와 상호작용이 가능하게 될 때

4. onPause()

Activity가 Foreground(제일 위)에 있지 않게 될 때.

화면 위에 다른 컴포넌트가 출력되어 사용자의 입력을 받지 못하게 됐을 때.

5. onStop()

Activity가 화면에 더 이상 나오지 않게 될 때

6. onDestroy()

Activity가 종료되거나 완전히 새로 로딩되어야 할 때 (화면 회전 등)

Intent

안드로이드에서는 Activity를 사용자가 직접 생성할 수 없다.

```
MyActivity myActivity = new Activity(); // 사용 안됨
```

이렇게 사용할 수가 없다.

안드로이드에게 Activity생성을 요청하여 요청이 처리되면 Activity가 생성된다.

이러한 요청을 "Intent"객체를 이용해서 작성한다.

Activity, Service, Broadcast Receiver 등 모든 안드로이드 컴포넌트가 동일하게 Intent를 사용해서 생성되어야함

장점: 다른 앱의 Activity도 동일하게 호출 가능

- 명시적 Intent: 내 App내의 Activity를 class명과 함께 요청하는 것.
- 묵시적 Intent: 내가 필요한 동작을 지정하여 요청하는 것.

사용자가 앱을 선택하여 실행해야 할 경우

ex) 링크 공유하기, 전화하기

Activity를 Manifest에 추가

AndroidManifest.xml 수정

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="com.andrstudy.activities">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="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>
        <!--이부분 추가-->
        <activity android:name=".SecondActivity"></activity>
    </application>

</manifest>
```

Intent 실행

```
Intent i = new Intent(this, SecondActivity.class);
startActivity(i);
```

Activity Stack

Activity를 호출하면 Stack에 쌓임 (즉 현재 페이지가 바뀌는 것이 아니라 액티비티가 위에 쌓임)

이후 사용자가 뒤로가기 키를 누르거나 액티비티 함수에서 `finish()` 를 호출하면 Stack에서 나옴.

Intent에 데이터 전달

- 단방향 전달

- 보내는 쪽

```
Intent i = new Intent(this, SecondActivity.class);
i.putExtra("message", "Hello!!");
startActivity(i);
```

- 받는 쪽

```
String message = getIntent().getStringExtra("message");
if(message != null) {
    // 받은 message 사용
}
```

- 양방향 전달

- Request Number 부여

```
// MainActivity.java
private static final int REQ_THIRD = 123;
```

- MainActivity에서 호출

```
// MainActivity.java
Intent i = new Intent(this, SecondActivity.class);

startActivityForResult(i, REQ_THIRD);
```

- MainActivity에서 결과를 받았을 때 메소드 생성

```
// MainActivity.java
@Override
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {

    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode == REQ_THIRD){
        String resultStr = "Result Code: " + resultCode;
        if(data != null ){
            String result = data.getStringExtra("result");
            if(result != null)
                resultStr += (" " + result);

        }

        Toast.makeText(this, resultStr, Toast.LENGTH_SHORT)
            .show();

    }
}
```

- SecondActivity에서 데이터 전송

```
// SecondActivity.java
Intent i = new Intent();
i.putExtra("result", "OK");
setResult(Activity.RESULT_OK, i);
finish();
```

묵시적 Intent 사용

- 문자 보내기 예시

```
Intent i = new Intent(Intent.ACTION_SENDTO);
i.setData(Uri.parse("smsto:010-4524-5468"));
i.putExtra("sms_body", "Hello");
startActivity(Intent.createChooser(i, "Select one"));
```

자세한 것은 생략

내 앱이 Intent 받기

내 앱이 공유하기에 관심이 있다는 사실을 Android에게 알림


```

<!-- AndroidManifest.xml -->
<activity android:name=".ThirdActivity"
    android:parentActivityName=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>

```

그 뒤 `ThirdActivity` 에서 Intent를 받았을 때 행동 처리

```

// ThirdActivity.java onCreate메소드 안
Intent i = getIntent();
if(i.getType() != null && i.getType().equals("text/plain")){
    setResult(Activity.RESULT_OK);
    String text = i.getStringExtra(Intent.EXTRA_TEXT);
    if(text != null){
        TextView tv = findViewById(R.id.textView);
        tv.setText(text);
    }
}

```

6. Data

Shared Preferences

key, value 형태의 간단한 데이터를 저장할 때 사용

특징

- Activity가 종료 된 이후에서 데이터를 저장할 수 있다.
- 정해진 위치에 파일이 생성되며 Key-value 형태로 저장된다.
- Key값을 이용한 값의 저장 및 불러오기만 가능

값 불러오기

```

// MainActivity.java
private SharedPreferences preferences;
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    preferences = getSharedPreferences("user", MODE_PRIVATE);
    String user = preferences.getString("user", null);
}

```

값 저장하기

값 불러오기

```
SharedPreferences.Editor editor = preferences.edit();
editor.putString("userName", userName);
// 아래 방법 중 하나를 선택
editor.apply();    // 비동기 방식
editor.commit();   // 동기 방식
```

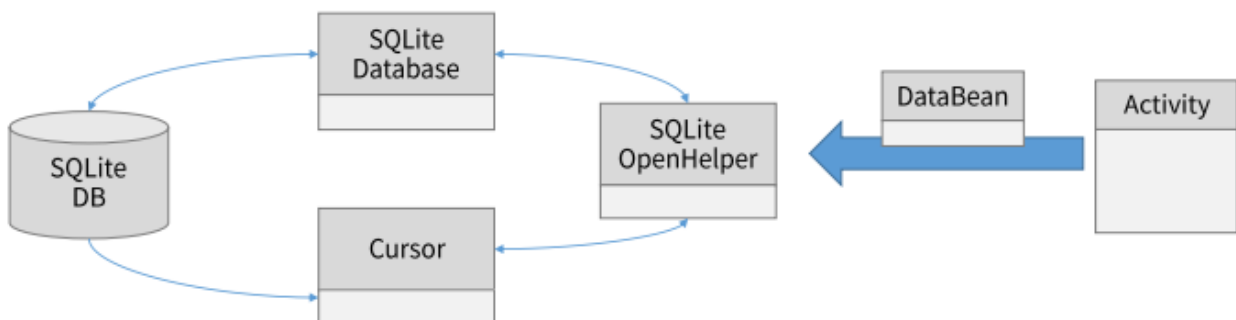
SQLite

안드로이드에서 기본 제공하는 DBMS

특징

- 앱을 삭제할 경우에 데이터가 사라짐.
- 앱에서 DB를 만들고 버전 관리를 할 수 있다.

구조



생성

2. UserBean 클래스 생성

```

public class UserBean {
    private int sequenceNumber;
    private String name;
    public int getSequenceNumber() {
        return sequenceNumber;
    }
    public void setSequenceNumber(int sequenceNumber) {
        this.sequenceNumber = sequenceNumber;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}

```

3. History Helper 클래스 생성

```

public class HistoryDBHelper extends SQLiteOpenHelper {
    // version을 제외한 다른 인자값에 @Nullable이 있다. 여기서 선택
    public HistoryDBHelper(Context context, String name,
        SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }
    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
    }
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
    }
}

```

4. History Helper 함수 구현

```

// HistoryDBHelper.java 클래스 내
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    String sql = "create table history ( sequenceNumber integer primary key
        autoincrement, name text)";
    sqLiteDatabase.execSQL(sql);
}

```

```

@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
    String sql = "drop table history";
    sqLiteDatabase.execSQL(sql);
    onCreate(sqLiteDatabase);
}

public long insert(UserBean user){
    SQLiteDatabase db = getWritableDatabase();
    ContentValues value = new ContentValues();
    value.put("name", user.getName());
    return db.insert("history", null, value);
}

public ArrayList getAll(){
    SQLiteDatabase db = getReadableDatabase();
    Cursor cursor = db.query("history", null, null, null, null, null, null);
    ArrayList result = new ArrayList<>();
    while(cursor.moveToNext()){
        UserBean user = new UserBean();
        user.setSequenceNumber(
            cursor.getInt(cursor.getColumnIndex("sequenceNumber"))
        );
        user.setName(
            cursor.getString(cursor.getColumnIndex("name"))
        );
        result.add(user);
    }
    return result;
}

public long delete(UserBean bean){
    SQLiteDatabase db = getWritableDatabase();
    String sequence = String.valueOf(bean.getSequenceNumber());
    return db.delete("history",
        "sequenceNumber=?", new String[] {sequence});
}

public long delete(){
    SQLiteDatabase db = getWritableDatabase();
    return db.delete("history", null, null);
}

```

5. MainActivity 수정

```

private HistoryDBHelper dbHelper;
@Override
onCreate... {
    ...
    dbHelper = new HistoryDBHelper(this, "userdb", null, 1);
}

```

6. 유저 select 호출 및 출력

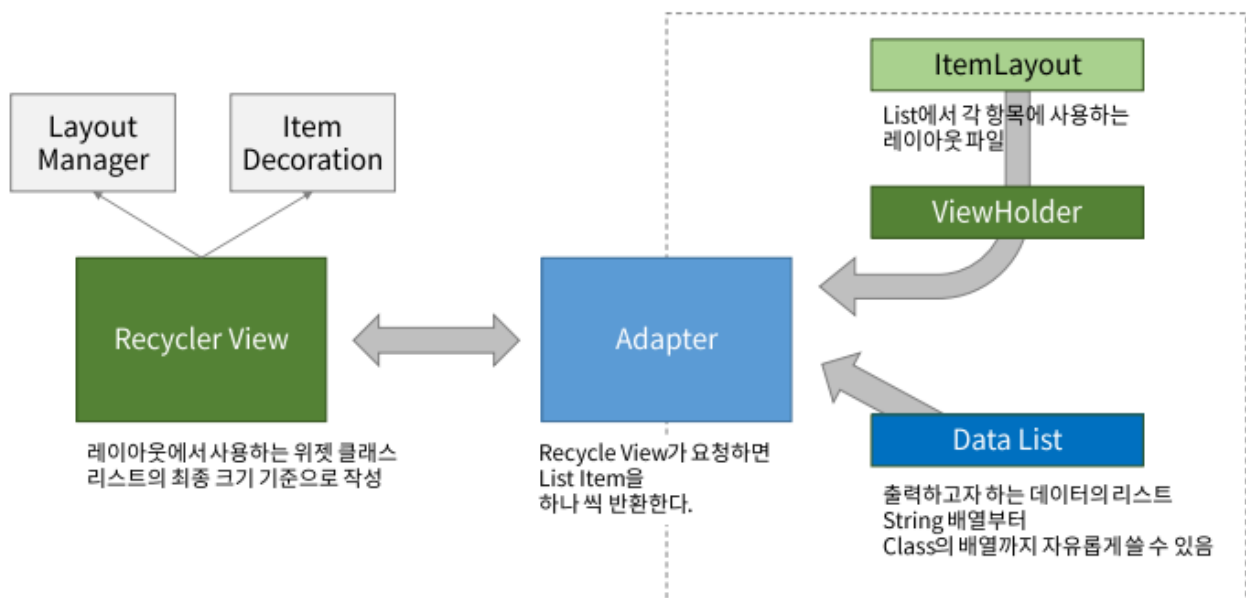
```
private void showUsers(){
    ArrayList users = dbHelper.getAll();
    for(UserBean u: users)
        Log.i("MAIN", "["+u.getSequenceNumber()+"]" + u.getName());
}
```

7. 유저 추가

```
UserBean user = new UserBean();
user.setName(userName);
dbHelper.insert(user);
```

7. RecyclerView

구조



나머지 스킵

8. Permission

각 어플리케이션은 고유의 ID와 영역을 가지고 실행된다.

특징

- 내 영역 밖에 데이터에 마음대로 접근할 수 없음.
- 내 영역에 다른 Application도 들어올 수 없음

내 영역 밖에 데이터나 리소스가 필요하면 권한을 신청해야 함

Permission 요청하기

1. Manifest.xml에 필요한 권한 추가 (정상 권한, 위험 권한 둘 다 해야함)

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

2. 정상 권한(다른 앱의 실행을 방해하거나 사생활을 침해하지 않는 동작)은 그냥 사용
3. 위험 권한은 사용자의 허가를 얻어 사용

정상 권한의 예: 인터넷, 블루투스, 와이파이 상태 등

위험 권한: 캘린더, 통화 이력 등

정상 권한은 바로 API 사용이 가능하다.

위험 권한에서 Permission 사용

- permission 요청

```
// MainActivity.java
public void onSendSMS(View v){
    if( Build.VERSION.SDK_INT >= 23){
        int permission = ContextCompat.checkSelfPermission(this,
            Manifest.permission.SEND_SMS);

        if(permission != PackageManager.PERMISSION_GRANTED){
            requestPermissions(new String[] {
                Manifest.permission.SEND_SMS}, REQ_SEND_SMS);
            return;
        }
    }

    sendSMS();
}
```

- permission 결과 받기

```
// MainActivity.java
@Override
public void onRequestPermissionsResult(int requestCode,
    @NonNull String[] permissions, @NonNull int[] grantResults) {
    if(requestCode != REQ_SEND_SMS) return;
    if(permissions[0].equals(Manifest.permission.SEND_SMS) &&
        grantResults[0] == PackageManager.PERMISSION_GRANTED)
        sendSMS();
    else
        Toast.makeText(this, "문자 전송 권한이 없습니다.",
            Toast.LENGTH_SHORT).show();
}
```

Broadcast Receiver

배터리가 부족하다, 폰이 부팅되었다 등 시스템 이벤트를 수신하는 기능이다.

블루투스 ON/OFF 감지 코드

```
// MainActivity.java
private BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {

        Log.d("Receiver", "onReceive");
        int state =
            intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, -1);

        switch(state){
            case BluetoothAdapter.STATE_ON:
                Log.d("Receiver", "Bluetooth ON");
                break;
            case BluetoothAdapter.STATE_OFF:
                Log.d("Receiver", "Bluetooth OFF");
                break;
        }
    }
};
```

```
// MainActivity.java
@Override
protected void onStart(){
    super.onStart();
    IntentFilter filter = new
        IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
    registerReceiver(receiver, filter);
}
@Override
protected void onStop(){
    super.onStop();
    unregisterReceiver(receiver);
}
```

9. StyleNSelector

Style과 Theme

- Style: View에 적용하는 모양과 형식.
- Theme: 액티비티나 Application에 적용하는 스타일.

테마 수정

res/values/style.xml

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme"
        parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item
            name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

res/values/style.xml에 커스텀 style 추가

```
<resources>
    ...
    <style name="BigText" parent="Widget.AppCompat.Button">
        <item name="android:textSize">24sp</item>
    </style>
</resources>
```


사용

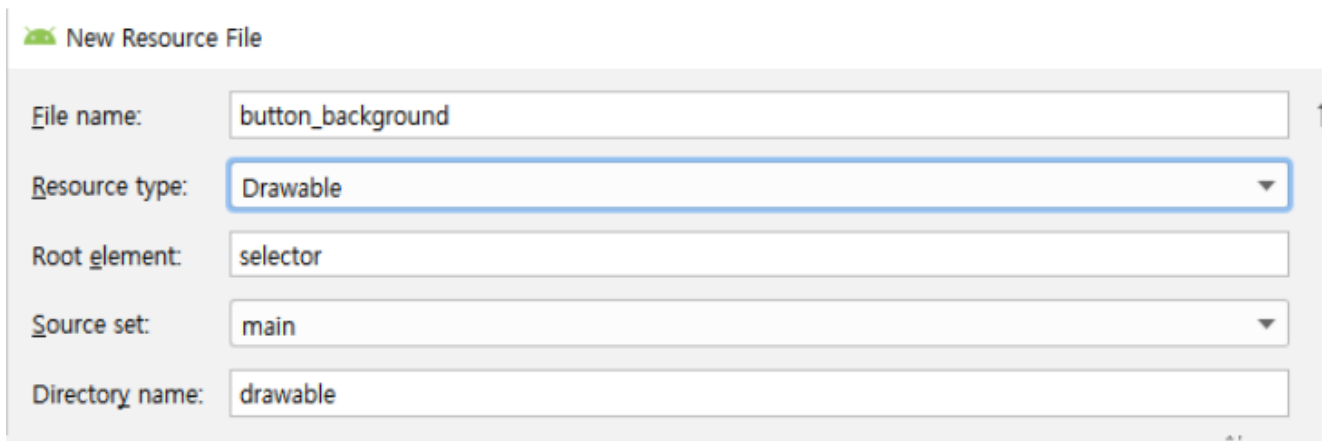
```
<Button
    android:id="@+id/button"
    style="@style/BigText"
    ...
```

Widget에 새 디자인 정의

widget은 여러 상태를 가진다.

위젯에 상태에 맞는 다양한 디자인을 적용하기 위해 필요한 것이 Selector이다.

1. colors.xml에 lightGray와 darkGray 추가 (스킵)
2. res폴더에서 layout file 생성. Root element는 `selector` 로 설정



New Resource File

File name:

Resource type:

Root element:

Source set:

Directory name:

3. layout 파일 작성

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="false"
        android:drawable="@color/lightGray"></item>
    <item android:state_pressed="true"
        android:drawable="@color/colorAccent"></item>
</selector>
```

state_pressed로 버튼의 상태를 정의한다.

4. selector 작성

color폴더에 파일을 추가해야 한다.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:state_pressed="false"
        android:color="@color/colorAccent"></item>
  <item android:state_pressed="true"
        android:color="@color/lightGray"></item>
</selector>
```

5. 위에서 만든 BigText를 수정

```
<style name="BigText" parent="Widget.AppCompat.Button">
  <item name="android:textSize">24sp</item>
  <item name="android:padding">24dp</item>
  <item name="android:background">@drawable/button_background</item>

  <item
    name="android:textColor">@color/button_text_color</item>
</style>
```

10. WorkWithServer

Server와의 통신

- HTTP: 안드로이드 앱에서 서버로 HTTP Request를 보낼 수 있다. 서버에 응답 시간이 필요하기 때문에 Thread로 처리해야 한다.

안드로이드에서 메인 스레드에서만 UI의 업데이트를 허용하기 때문에 **AsyncTask**를 사용해야 한다.

AsyncTask

AsyncTask란 별도의 Thread를 돌리면서도 UI를 갱신할 수 있도록 제공하는 클래스이다.

실습

1. AndroidManifest.xml에 인터넷 권한 추가 (위에 참고)
2. Task 클래스 추가

```

class Task extends AsyncTask<URL, Integer, String>{
    private WeakReference<MainActivity> activityReference;
    public Task(MainActivity activity){
        activityReference = new WeakReference<>(activity);
    }
    @Override
    protected void onProgressUpdate(Integer... values) {}
    @Override
    protected void onPreExecute() {super.onPreExecute();}
    @Override
    protected String doInBackground(URL... params) {
        return null;
    }
    @Override
    protected void onPostExecute(String s) {
        //super.onPostExecute(s);
        MainActivity activity = activityReference.get();
        if (activity == null || activity.isFinishing()) return;
    }
}

```

만약 서버가 고정IP를 가지지 못하는 경우 절대 localhost를 사용하면 안됨.

localhost로 요청하면 핸드폰 자기자신을 찾는다.

3. MainActivity에서 onCreate() 수정

```

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

    textView = findViewById(R.id.textView);
    try {
        URL url = new URL("http://ip:port/context");
        new Task(this).execute(url);
    } catch (MalformedURLException e) { }
}
@Override
protected void onPostExecute(String s) {
    //super.onPostExecute(s);
    MainActivity activity = activityReference.get();
    if (activity == null || activity.isFinishing()) return;
    textView.setText(s);
}

```

4. Task 클래스에서 함수 구현

```

@Override
protected void onProgressUpdate(Integer... values) {
    super.onProgressUpdate(values);
    if(values.length > 0)
        Log.i("http", String.valueOf(values[0]));
}

@Override
protected void onPreExecute() {
    super.onPreExecute();
}

protected String doInBackground(URL... params) {
    int i=0;
    String result = new String();
    if(params == null || params.length < 1) return null;
    try {
        publishProgress(i++);
        HttpURLConnection connection =
            (HttpURLConnection)params[0].openConnection();
        connection.setRequestMethod("GET"); // get방식
        //connection.setDoOutput(true); // 쓰기모드 - POST로 강제 설정됨
        connection.setDoInput(true); // 읽기모드
        connection.setUseCaches(false);
        connection.setDefaultUseCaches(false);
        publishProgress(i++);
        InputStream is = connection.getInputStream();
        StringBuilder builder = new StringBuilder(); //문자열을 담기 위한 객체
        BufferedReader reader = new BufferedReader(new
            InputStreamReader(is, "UTF-8")); //문자열 셋팅

        String line;
        while ((line = reader.readLine()) != null) {
            builder.append(line+ "\n");
            publishProgress(i++);
        }
        result = builder.toString(); Log.i("http", "result=" + result);
        publishProgress(i++);
    } catch(IOException me){ me.printStackTrace(); }

    return result;
}

```

11. Service

Android의 Component중 하나로 화면 없이 동작하는 코드이다.

보통 음악 플레이어, 채팅(알림) 등에 사용된다.

AndroidManifest.xml에 기술되어야 한다.

메소드

- `startService()` / `stopService()` : service를 시작하거나 종료하는 메소드.

메인 쓰레드 안에서 동작한다. (싱글톤으로 동작)

Activity와 Service는 독립적으로 동작하여서 Broadcast로 통신 가능

- `bindService()` / `unbineService()` : Service를 구동하는데, 모든 bind가 해제돼야 서비스가 종료된다. service와 차이점은 activity에서 service를 받아서 직접 데이터를 주고 받을 수 있다.

실습

1. AndroidManifest.xml에 만든 서비스 추가

```
<service
    android:name=".SimpleService"
    android:enabled="true"
    android:exported="true"></service>
```

2. Service 생성

```
public void startService(View v){
    Intent intent = new Intent(this, SimpleService.class);
    startService(intent);
}

public void stopService(View v){
    Intent intent = new Intent(this, SimpleService.class);
    stopService(intent);
}
```

12. FirebaseCloudMessaging

생략

13. CustomImageView

그림판을 만들어 본다.

키워드

- Canvas: 그림을 그릴 대상, 그림판
- Paint: 그림을 어떻게 그릴 것인가에 대한 정보를 가짐. 색, 선, 굵기

- onDraw(): 위젯을 새로 그려야 할 시점에 호출됨

파라미터에서 Canvas가 전달되며 그림을 그리면 화면에 출력된다.

- invalidate(): 개발자가 정한 시점에 화면을 갱신해야 할 경우 onDraw()를 호출하는 것이 아니라 invalidate()를 호출한다. 그러면 필요한 정보들이 준비된 다음 onDraw()가 호출된다.

구현

1. 전체 화면 앱으로 설정

```
<style name="AppTheme"
    parent="Theme.AppCompat.Light.NoActionBar">
```

2. DrawingView 클래스 생성 (선 색 변경 하지 않은 상태)

```

public class DrawingView extends AppCompatActivity
    implements View.OnTouchListener {

    private Path path;

    public DrawingView(Context context) {
        super(context);
        init();
        setOnTouchListener(this);
    }

    public DrawingView(Context context, AttributeSet attrs)
    {
        super(context, attrs);
    }

    public DrawingView(Context context, AttributeSet attrs, int
        defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }

    private void init(){
        paint = new Paint();
        paint.setColor(Color.BLACK);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeCap(Paint.Cap.ROUND);
        paint.setStrokeWidth(10);
        paint.setAntiAlias(true);
        path = new Path();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawPath(path, paint);
    }

    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        int action = motionEvent.getAction();
        switch(action){
            case MotionEvent.ACTION_DOWN:
                path.lineTo(motionEvent.getX(),
                    motionEvent.getY());
                break;
            case MotionEvent.ACTION_MOVE:
                path.moveTo(motionEvent.getX(),
                    motionEvent.getY());
                break;
            case MotionEvent.ACTION_UP: break;
        }

        invalidate();
        return false;
    }
}

```

3. 레이아웃(activity_main)에 DrawingView 추가

```
<com.andrstudy.drawingview.DrawingView
    android:id="@+id/drawingView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clickable="true" <!-- 이부분 중요 -->

    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
```